UCL
Université
catholique
de Louvain

Université Catholique de Louvain
Faculté des Sciences Appliquées
Laboratoire de Microélectronique
UCL Crypto Group

Block Ciphers: Security Proofs, Cryptanalysis, Design, and Fault Attacks

Gilles-François Piret

Thèse soutenue en vue de l'obtention du grade de
Docteur en Sciences Appliquées

Composition du jury:
Pr. Jean-Jacques Quisquater (UCL Crypto Group) - *Promoteur*
Pr. Philippe Delsarte (UCL/INGI)
Pr. Alphonse Magnus (UCL/ANMA)
Pr. Jacques Patarin (Université de Versailles, France)
Pr. Bart Preneel (KUL, COSIC)
Pr. Vincent Rijmen (TU Graz, IAIK, Austria)
Pr. Luc Vandendorpe (UCL/TELE) - *Président*

Janvier 2005

Abstract.

Block ciphers are widely used building blocks for secure communication systems; their purpose is to ensure confidentiality of the data exchanged through such systems, while achieving high performance. In this context, a variety of aspects must be taken into account. Primarily, they must be secure. The security of a block cipher is usually assessed by testing its resistance against known attacks. However as attacks may exist that are currently unknown, generic security proofs are also tried to be obtained. On the other hand, another attack methodology is also worth considering. Contrary to the others, it aims at the implementation of the algorithm rather than the cipher itself. It is known as *side-channel analysis*. Finally, performance of a block cipher in terms of throughput is very important as well. More than any other cryptographic primitive, block ciphers allow a tradeoff to be made between security and performance.

In this thesis, contributions are given regarding these various topics. In the first part of the thesis, we deal with two particular types of attacks, namely the square attack and key schedule cryptanalysis. We also consider security proofs in the so-called Luby-Rackoff model, which deals with adversaries having unbounded computation capabilities. More precisely, we are interested in the Misty structure, when the round functions are assumed to be involutions.

The second part of the thesis is devoted to design and implementation aspects. First, we present a fault attack on substitution-permutation networks, which requires as few as two faulty ciphertexts to retrieve the key. We also study the security of $DeKaRT$, which is an algorithm intended to protect smart cards against probing attacks. Finally we present the design of `ICEBERG`, a block cipher deliberately oriented towards good performance in hardware, and give an adequate analysis of its security.

Acknowledgements

It is my pleasure to thank all the people who helped me throughout the achievement of this thesis.

In first place, I would like to thank my advisor Pr Jean-Jacques Quisquater for introducing me to cryptography, for giving me the opportunity to carry out this research in his laboratory, for his trust and his kind support.

I am grateful to Pr Jacques Patarin and Pr Vincent Rijmen for having accepted to be members of my thesis committee. I would also like to thank Pr Philippe Delsarte, Pr Alphonse Magnus, Pr Bart Preneel for serving in the jury, and Pr Luc Vandendorpe who accepted to preside the committee. Their presence was an honor for me.

I am also grateful to the European Commission and to the Walloon Region, for their financial support throughout several research projects. In other respects, I also thank the system administration team at the microelectronics laboratory.

I would like to thank all members of UCL Crypto Group, past and present, for the nice working (and more...) atmosphere. Special thanks to my coauthors and to all those with whom I had fruitful technical discussions.

I am also grateful to all the persons who accepted to have a critical look at preliminary versions of this thesis, in alphabetical order: Philippe Bulens, Mathieu Ciet, Christophe Giraud, François Koeune, François Macé, Marine Minier, François-Xavier Standaert. Special thanks to Sylvie Baudine who corrected my English mistakes in this thesis as well as in my papers, where I too often forgot to thank her.

There is a life beside cryptography. It is why I would finally like to thank all the persons with whom I spent time when trying to escape from my PhD work...

Contents

Notations

$\oplus$	:	the exclusive-or operation or addition modulo 2.
$\mathbb{Z}_2^n$	:	the group of which the elements are n-bit words, with group operation $\oplus$.
$\overline{x}$	:	for $x \in \mathbb{Z}_2^n$, $\overline{x}$ denotes the one's complement of x, that is, $x \oplus \overline{x} = 2^n - 1$.
$x \bullet y$	:	for $x, y \in \mathbb{Z}_2^n$, $x \bullet y$ is the usual scalar product of x and y.
$\mathbf{F}_{n,p}$	:	the set of all functions from $\mathbb{Z}_2^n$ to $\mathbb{Z}_2^p$.
$\mathbf{P}_n$	:	the set of all permutations on $\mathbb{Z}_2^n$.
$\mathrm{GF}(2^p)$	:	the Galois field with 2^p elements.
`0x`	:	the value which follows is in hexadecimal notation.
$x \ll t$	:	for $x \in \mathbb{Z}_2^n$, x left-shifted by t bits, $t \leq n$. The t leftmost bits are discarded and the t rightmost bits are filled with 0.
$x \lll t$	:	for $x \in \mathbb{Z}_2^n$, x left-rotated by t bits, $t \leq n$.
$x \gg t$	:	for $x \in \mathbb{Z}_2^n$, x right-shifted by t bits, $t \leq n$. The t rightmost bits are discarded and the t leftmost bits are filled with 0.
$x \ggg t$	:	for $x \in \mathbb{Z}_2^n$, x right-rotated by t bits, $t \leq n$.
$\complement S$	:	the complement of the subset S.
$\lvert X \rvert$	:	depending on the context, either the absolute value of the integer X, or the cardinality of the set X.
M^T	:	the transpose of matrix M.
$n!$	:	for a positive integer n, $n! = n \cdot (n-1) \cdot ... \cdot 1$. By convention $0! = 1$.
$f \cdot g$ or $g \circ f$	:	the composition $g(f)$ of two functions f and g.
$\mathcal{P}[A]$	:	the probability of event A.
$\overline{A}$	:	the complementary event of A.

List of Abbreviations

AES	: Advanced Encryption Standard
ASIC	: Application Specific Integrated Circuit
CLB	: Configurable Logic Block
CRT	: Chinese Remainder Theorem
DES	: Data Encryption Standard
DSA	: Digital Signature Algorithm
ECB	: Electronic Code Book
ECDSA	: Elliptic Curves DSA
E/D	: Encryption/Decryption
EEPROM	: Electrically Erasable Programmable Read-Only Memory
FIPS	: Federal Information Processing Standard
FPGA	: Field Programmable Gate Array
Gbps	: Gigabits per second
IDEA	: International Data Encryption Algorithm
I/O	: Input/Output
LFSR	: Linear Feedback Shift Register
LUT	: Look-Up Table
MB	: MegaByte
NBS	: National Bureau of Standards
NESSIE	: New European Schemes for Signatures, Integrity, and Encryption
NIST	: National Institute of Standards and Technology
NSA	: National Security Agency
RAM	: Random Access Memory
RAMB	: Random Access Memory Block
RSA	: Rivest-Shamir-Adleman cryptosystem
S-boxes	: Substitution boxes
SPN	: Substitution-Permutation Network
XL	: eXtended Linearization
XOR	: eXclusive OR

Introduction

The expansion of the digital society has implied a big increase of the amount of data exchanged through long distance channels. These data must be protected in several ways: when two parties are communicating, they want to ensure that the data they share are not eavesdropped (*confidentiality*), that they are not modified by a third-party (*data integrity*), and that the sender is actually who he claims he is (*entity authentication*). *Cryptology* is the science that addresses these various concerns.

In this thesis we are interested in the way *confidentiality* can be reached. This is performed by *encryption*: the sender of a message (or *plaintext*) enciphers it by applying to it an algorithm that depends on a small piece of secret information, called *key*. The piece of enciphered information that is obtained (called *ciphertext*) is sent to the receiver. The latter applies another algorithm parameterized also by a key in order to recover the plaintext. In the context of this thesis the keys used for encryption and decryption are the same: we speak of *symmetric* encryption.

This model requires for the sender and the receiver to be able to share a key. For this purpose we assume they are able to access a secure channel which cannot be eavesdropped. But for some reason (because using it is too costly, or because it is only temporary) it is not convenient to exchange a big amount of data on this channel. It is why the secure channel is used only to agree on a key. Then an insecure channel, which can be eavesdropped, is used. Figure 1 summarizes this framework.

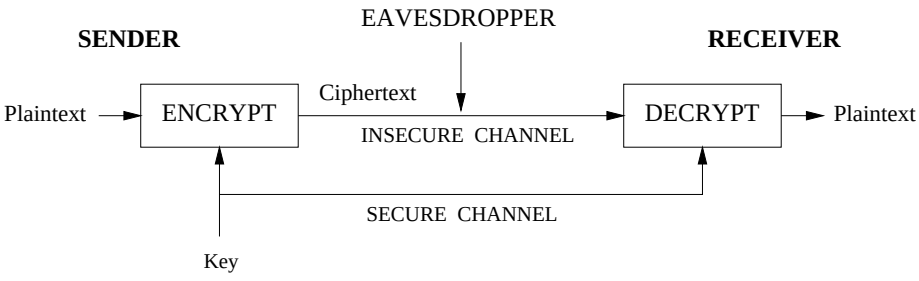


FIGURE 1. Symmetric encryption: framework.

In this thesis we focus on an important class of symmetric encryption algorithms: *block ciphers* (the other class being the one of *stream ciphers*). A variety of aspects must be taken into account when studying block ciphers:

- Primarily, they must be secure, in the sense that an attacker cannot get any information about the plaintexts by observing the ciphertexts. More than that, she should not be able to find the key faster than by exhaustively searching the key space, even knowing several plaintext-ciphertext pairs (possibly a lot of them).
  Exotic attack contexts are sometimes considered. As a matter of fact, the attacker can be assumed to have access to plaintext-ciphertext pairs encrypted under two unknown keys with a given known relationship between them; we speak of *related-key attacks*.
  Assessing security of a block cipher is usually done by proving its resistance to all known attacks. However it is not fully satisfactory, as attacks may exist that are currently unknown. It is why generic *security proofs* on block ciphers are tried to be obtained. But nowadays such security proofs only exist under very specific assumptions.
- Another concern is related to the context in which a block cipher is applied. Unprotected implementations of block ciphers offer opportunities to *side channel attacks*: by observing the behavior of a computing device, such as a smart card, through various channels (such as power consumption, or time of computation), an attacker can be able to deduce the secret key. Among them, *fault attacks* try to tamper with the correct execution of a cryptographic algorithm and use the faulty results obtained in order to retrieve the key.
- Finally, performance of a block cipher in terms of throughput is very important as well. An important factor that influences performance is how well the cipher fits with the platform on which it is used (home PC, smart card, dedicated hardware...); some ciphers are more adapted to some platforms. Moreover, when designing a block cipher it is always possible to improve its performance by lowering the security margin, and vice-versa. Therefore designing a good block cipher is a real security-performance tradeoff.

In this thesis, contributions regarding these various aspects are given:

- Chapter I provides the basics of block cipher analysis and design. We present the basic components of a block cipher, and

two widely used structures: the Feistel and Substitution-Permutation networks. We define the notion of distinguisher, and describe two well-known attacks: linear and differential cryptanalysis. Finally, the history and overall structure of the DES and AES block ciphers are given.

- Chapter II deals with security proofs in the Luby-Rackoff model [108], which assumes the attacker has unbounded computation capabilities. More precisely, we prove security bounds for the structure underlying to the Misty [114] block cipher, when the round functions are involutions without fixed point.
- Chapter III analyzes a particular type of attack: the *square attack*. It is applied to reduced-round versions of both algorithms Skipjack and SAFER++ [112]. Possible extensions of the square attack are also discussed.
- Chapter IV describes *related-key attacks*, and makes a survey of existing attacks in the field of *key schedule cryptanalysis*.
- Chapter V deals with *fault attacks*. It presents a generic fault attack against substitution-permutation networks, and applies it to the standard AES.
- Chapter VI analyzes the security of $DeKaRT$, which is a *scrambling function*; scrambling functions are encryption functions with much weaker security requirements than block ciphers, but much better efficiency as well. They are used on smart cards to counter some side-channel attacks.
- Chapter VII presents `ICEBERG`, a block cipher we designed such as to obtain excellent performances when implemented in hardware; it also offers good opportunities to defeat side-channel attacks. Although we are not specialists in hardware and hardware implementations, in a care of completeness we also describe implementation results for `ICEBERG` and compare them to those obtained for other recent block ciphers.

CHAPTER I

Preliminaries

Abstract. In this chapter we discuss the fundamentals of block cipher design and analysis. We present the basic components of an iterated block cipher and describe the Substitution-Permutation Network and Feistel Network design paradigms. We then explain the notion of distinguisher and its use for key retrieval. Finally we briefly describe the two best-known attacks on modern block ciphers, linear and differential cryptanalysis, and design principles to counter them.
This chapter is not intended to be a review on block ciphers, but rather to provide the non-specialist reader with the basics necessary to (hopefully) understand the other chapters of this thesis.

I.1. What is a Block Cipher?

A block cipher is a pair of functions

$$\begin{aligned} E &: \mathbb{Z}_2^k \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n \\ \text{and} \quad D &: \mathbb{Z}_2^k \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n. \end{aligned} \tag{1}$$

It is intended to provide confidentiality of data circulating over an insecure channel. E is the *encryption* function, that turns a n-bit plaintext onto a n-bit ciphertext under control of a k-bit key K; D is the *decryption* function, that turns a n-bit ciphertext onto a n-bit plaintext under control of a k-bit key K. n is the *block size*; its usual value is 64 or 128 bits. k is the *key size*; nowadays the most classical value for k is 128. For each key K, define $E_K : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ by $E_K(P) = E(K, P)$, and $D_K(P) = D(K, P)$. E_K must be invertible for all K, with $E_K^{-1} = D_K$. Furthermore, both E and D should be easily computable. Note that, as their name suggests, block ciphers encrypt data partitioned in blocks. On the contrary, *stream ciphers* encrypt data one bit at a time[1].

Most block ciphers are *iterated* block ciphers. It means that they consist in the repetition of the same function F (sometimes up to minor differences, mainly in the first and last rounds) a certain number of times R (typically from 6 to 32). However each application of F is parameterized by a different *round key*. Round keys are derived from the master key K using a *key scheduling algorithm*; we obtain the *expanded key*

[1]With the notable exception of RC4, which generates random *bytes*.

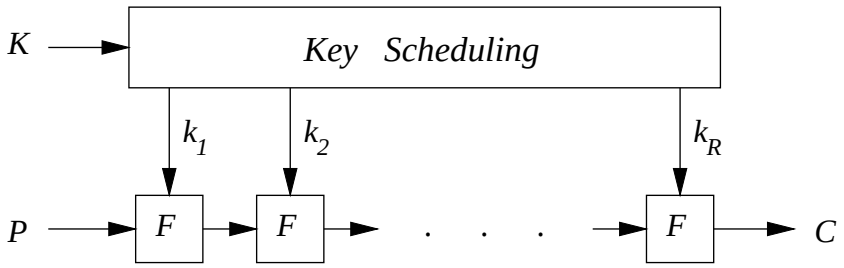


FIGURE 1. Overall structure of an iterated block cipher.

$(k_1, k_2, ..., k_R)$. Usually the key schedule derives the round keys iteratively: first k_1, then k_2, ... The sequence of operations that permits to obtain a new round key k_i is sometimes called *key round*. Figure 1 pictures the overall structure of an iterated block cipher.

The construction of the F function relies on the principles of *confusion* and *diffusion* edicted by C. Shannon in his seminal work [157]. Informally, confusion consists in making the relation between input bits, output bits, and key bits, as complex and tricky as possible; diffusion aims at spreading out the bits of the message. Basically, three different types of components are used:

- Key mixing layers incorporate key material. It is usually done using a simple addition, such as exclusive or ($\oplus$) or addition mod 2^m for some m.
- Non-linear layers achieve confusion. They usually consist in the parallel application of highly nonlinear functions called *S-boxes*, and operating on a small number of bits, typically from 4 to 8[2]. By a $p \times q$ S-box we mean a function S from $\mathbb{Z}_2^p$ to $\mathbb{Z}_2^q$.
- Diffusion layers, which are usually linear.

Note that the components of some iterated block ciphers do not exactly comply with this description, although the three basic principles (key mixing, confusion and diffusion) remain present. We think of the IDEA [105, 106] algorithm for example.

A trivial way to find the unknown key of a block cipher is an exhaustive search: provided the attacker knows a few plaintext-ciphertext pairs (P_i, C_i) encrypted with the unknown key K, she tries to encrypt P_1 under every possible key K', until she obtains C_1 as the ciphertext. Then $K' = K$ with a "high" probability[3]; she can check using the other

[2]The size of these S-boxes is limited by the fact that they are usually implemented in software using *table lookups*. As storing a $m \times n$ S-box requires $n \cdot 2^m$ bits of memory, the S-boxes must be kept small.

[3]The exact probability depends on the ratio between the sizes of the block and the key space. If they are equal, the probability is indeed high.

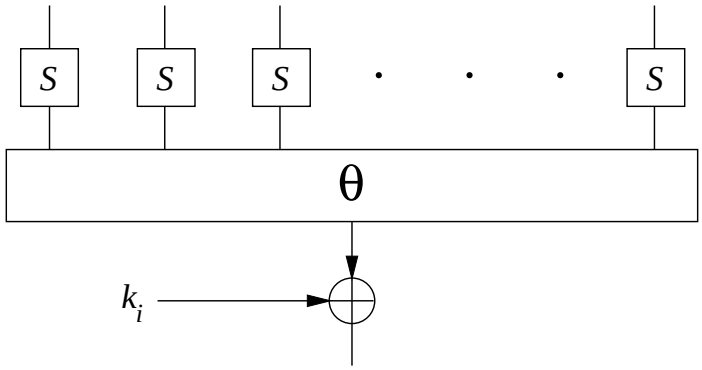


FIGURE 2. One round of a Substitution-Permutation network.

plaintext-ciphertext pairs. This attack permits her to find the key with an average of 2^{k-1} trial encryptions.

A consequence of this attack is that it gives an upper bound on the work we can expect for an attack to work: if it is greater or equal than the one required by exhaustive search, it is useless. On the other hand, a block cipher is academically considered as broken as soon as an attack faster than exhaustive search is found against it.

I.2. Substitution-Permutation Networks and Feistel Ciphers

Substitution-Permutation Networks (SPNs) are block ciphers with a very simple structure. Nevertheless they can be secure and efficient, as testified by the large number of good ciphers built using this paradigm. The round of a SPN is made out of three layers:

- The key addition layer $\sigma[k]$, usually performed using exclusive or ($\oplus$): $\sigma[k](a) = a \oplus k$.
- The diffusion layer θ, which is linear with respect to $\oplus$ (or more generally with respect to the group operation used for key addition)[4].
- The non-linear layer γ, which is made out of the parallel application of S-boxes (not necessarily all identical, although it is often the case).

Therefore the round of a SPN is described as

$$\rho[k] = \sigma[k] \circ \theta \circ \gamma. \tag{2}$$

However it makes no sense to begin encryption with a component other than key addition (as it could be undone). Therefore a key addition

[4]Strictly speaking, the designation *substitution-permutation network* implies that the diffusion layer is a bit permutation. However, it becomes more and more used to refer also to ciphers with a more complex diffusion layer. So do we.

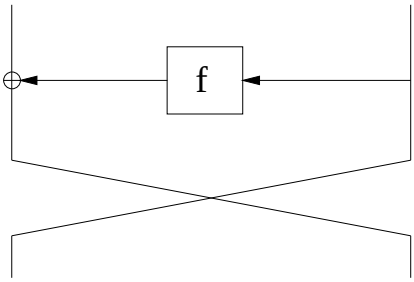


FIGURE 3. One round of a Feistel network.

layer is added before the first round. Also, as $\sigma[k] \circ \theta = \theta \circ \sigma[\theta^{-1}(k)]$, the last θ layer is not cryptographically useful (it can be undone as well). Thus a whole R-round substitution-permutation network is described as

$$\sigma[k_R] \circ \gamma \circ (\bigcirc_{r=1}^{R-1} \sigma[k_r] \circ \theta \circ \gamma) \circ \sigma[k_0], \tag{3}$$

where $k_0, k_1, ..., k_R$ are round keys.

The main advantage of substitution-permutation networks is that their simple structure makes them easy to analyze, which minimizes the probability for a design flaw to remain hidden because of a too intricate structure. Examples of block ciphers complying with this structure are the AES [47] (see section I.6), Serpent [2], Noekeon [45], `ICEBERG` [161] (see Chapter VII),...

Another structure is widely used: the ***Feistel Network***. It is due to H. Feistel [54]. The data X entering a Feistel round is divided into two equal parts: $X = \langle L, R \rangle$. The round is defined as

$$\phi(f)(\langle L, R \rangle) = \langle R, L \oplus f(R) \rangle \tag{4}$$

where f is the *round function*. It is made out of diffusion and confusion layers, and a key mixing layer.

The two advantages of using a Feistel Network are the following:

- The function f does not need to be bijective[5]. It allows more freedom in its design.
- More importantly, the inverse of $\phi(f)$ is $s \circ \phi(f) \circ s$, where s denotes the swap of both parts: $s(\langle L, R \rangle) = \langle R, L \rangle$. Therefore decryption distinguishes from encryption by the order of the round keys and by initial and final swaps only, which is a big advantage for implementation in hardware and in constrained environments.

[5]Although the fact that it is not "close" enough to surjective could lead to attacks, as demonstrated by V. Rijmen et al. [147].

Examples of Feistel ciphers are the former standard DES [164] (see section I.6) or LOKI [31, 30]. Several other ciphers use the Feistel structure in their design, but are not "pure" Feistel ciphers; examples are Twofish [154], Misty [114], or Camellia [5].

I.3. Distinguishers and Their Use to Retrieve the Key

A block cipher can be viewed as a family of permutations, indexed by a key. The strongest property we can expect from it is to be indistinguishable from a completely random permutation, when the key is selected uniformly at random. We define a ***distinguisher*** $\mathcal{A}$ for a given block cipher (E_K, D_K) as an algorithm which is given black-box access to a certain permutation F. It means that the only way it can access F is by querying it to obtain m plaintext-ciphertext pairs (P_i, C_i) such that $F(P_i) = C_i$. Based on $\{(P_i, C_i)\}_{i=1,\ldots,m}$, it decides whether the permutation which produced these pairs was most probably selected by random drawing[6] from the set $\mathbf{P}_n$ of all permutations of $\mathbb{Z}_2^n$, or by randomly[6] choosing a key $\overline{K}$ and selecting $(E_{\overline{K}}, D_{\overline{K}})$. It outputs 1 in the first case, 0 in the second.

Let $s \in_R S$ denote the fact that the random variable s is drawn from set S with respect to the uniform probability distribution. Let $\mathcal{A}^F$ denote the output of $\mathcal{A}$ when querying permutation F (even for fixed F, $\mathcal{A}^F$ can be a random variable if $\mathcal{A}$ is not deterministic). Then the ***advantage*** $\mathrm{Adv}_{\mathcal{A}}$ algorithm $\mathcal{A}$ has in distinguishing (E_K, D_K) from a random permutation is defined as

$$\mathrm{Adv}_{\mathcal{A}} = |\mathcal{P}[\mathcal{A}^F = 1 | F \in_R \mathbf{P}_n] - \mathcal{P}[\mathcal{A}^F = 1 | F = E_K, K \in_R \mathbb{Z}_2^k]|. \quad (5)$$

Informally speaking, a distinguisher is efficient if it achieves a non-negligible advantage for a "low enough" number m of queries and a "small enough" number of computations. The type of the attack depends on the way pairs (P_i, C_i) are selected:

- *Known plaintext attacks*: The distinguisher has no control on the plaintexts P_i that are encrypted to ciphertexts C_i. A priori, they are assumed to be uniformly distributed.
- *Ciphertext-only attacks*: The distinguisher only gets a certain number of ciphertexts C_i, without the corresponding plaintexts. It only knows the probability distribution of these plaintexts (for an attack to work, it has to be non-uniform).
- *Chosen plaintext attack*: The distinguisher chooses the pool $\{P_i\}_{i=1,\ldots,m}$ of plaintexts that are encrypted. All plaintexts are chosen simultaneously before receiving the corresponding ciphertexts $\{C_i\}_{i=1,\ldots,m}$.

[6] with respect to the uniform probability distribution

- *Adaptive chosen plaintext attacks*: The distinguisher chooses the pool $\{P_i\}_{i=1,...,m}$ of plaintexts that are encrypted. Moreover, it is allowed to adapt the encryption queries it makes as a function of the ciphertexts already obtained.
- *(Adaptive) chosen ciphertext attacks*: They are similar to the two previous attacks, but the distinguisher is allowed to make decryption queries instead of encryption queries.
- *(Adaptive) chosen plaintext and ciphertext attacks*: In this case, both encryption and decryption queries are allowed.

Many distinguishers are *statistical* distinguishers. It means that they compute some statistic using the plaintext-ciphertext pairs $\{(P_i, C_i)\}_{i=1,...,m}$. Then they decide whether the value obtained is more likely to correspond to a random permutation or to the cipher considered, using *hypothesis testing*. The advantage of such distinguishers is that, besides the final decision, they give a "score" (the value of the statistic) indicating the reliability of the decision. It is useful when a distinguisher is used to retrieve key material, as we will now explain.

As a matter of fact, besides the theoretical interest of finding an efficient distinguisher for a given block cipher, it often permits the retrieval of key material. Consider a distinguisher $\mathcal{A}$ against a given block cipher of which the first and last rounds have been removed. Denote X_1 the state after the first round and X_{R-1} the state before the last round. Usually knowledge of only a few bits of X_1 and X_{R-1} is necessary for the distinguisher to work; let us denote them by $X_1^{(1)}, X_1^{(2)}, ..., X_1^{(a)}$ and $X_{R-1}^{(1)}, X_{R-1}^{(2)}, ..., X_{R-1}^{(b)}$. Therefore an attacker can use $\mathcal{A}$ to retrieve some bits of the first and last round keys as follows:

(1) She obtains a certain number of plaintext-ciphertext pairs $\{(P_i, C_i)\}_{i=1,...,m}$.
(2) She guesses the key bits of k_1 and k_R that are necessary to compute $X_1^{(1)}, X_1^{(2)}, ..., X_1^{(a)}$ and $X_{R-1}^{(1)}, X_{R-1}^{(2)}, ..., X_{R-1}^{(b)}$. Then for each guess, she uses the distinguisher to accept or reject the key guess. In the case where a statistical distinguisher is used, a ranking of the key candidates, from the most likely to the less likely, is possible[7].

Once bits of the round keys are found (or assumed to be found), the other key bits can be retrieved by repeating the attack targeting other key bits, or by exhaustive search.

Sometimes key guesses are made on more than one round at each end of the algorithm; or they are made at one end of the algorithm only. When key guesses are made on l rounds on the whole, we speak of a *lR-attack*.

[7]The work of P. Junod formally deals with this problem [84, 83].

In the next two sections we present two well-known attacks based on the notion of distinguisher. It will make the concepts presented in this section clearer.

I.4. Linear Cryptanalysis

The discovery of linear cryptanalysis is usually attributed to M. Matsui [113]. However the first appearances of the idea can be found in the paper of A. Tardy-Corfdir and H. Gilbert [62]. In order to build a linear distinguisher on a block cipher, the cryptanalyst tries to identify a relationship between some plaintext bits, some ciphertext bits, and some key bits:

$$\mu_P \bullet P \oplus \mu_C \bullet C = \mu_K \bullet K \tag{6}$$

where μ_P, μ_C, μ_K are masks and $\bullet$ denotes the scalar product over $\mathbb{Z}_2^a$.

This equation must hold with a probability p far enough from $1/2$ in order to allow the attacker to distinguish between the block cipher and a random permutation. Therefore we define the ***bias*** of a linear relationship Λ as the distance of p with $1/2$:

$$\varepsilon_\Lambda = |p - 1/2|. \tag{7}$$

We also define the ***correlation*** between two Boolean functions:

Definition 1. Let $f, g : \mathbb{Z}_2^p \to \mathbb{Z}_2$. The ***correlation*** between f and g is

$$c(f, g) = 2^{1-p} \cdot |\{x | f(x) = g(x)\}| - 1.$$

The problem is to identify relations holding with a high enough bias. For this purpose, the cryptanalyst first computes the bias of all non-trivial linear relations at the S-box level: for a $p \times q$ S-box, she computes the bias for all pairs of masks (α, β), with $\alpha \in \mathbb{Z}_2^p \backslash \{0\}$ and $\beta \in \mathbb{Z}_2^q \backslash \{0\}$:

$$\varepsilon_{\alpha,\beta} = \frac{|c(l_\alpha, l_\beta \circ S)|}{2}, \tag{8}$$

where for $\alpha \in \mathbb{Z}_2^n$, l_α is defined as

$$l_\alpha : \mathbb{Z}_2^n \to \mathbb{Z}_2 : l_\alpha(x) = \alpha \bullet x. \tag{9}$$

Linear relations with a high bias are naturally particularly interesting. The attacker then tries to combine linear approximations of these S-boxes through several rounds, such that the bits of the intermediate states cancel by summation. If t linear relations of respective biases $\varepsilon_1, \varepsilon_2, ..., \varepsilon_t$ are summed, the bias ε of the linear relation obtained can be computed using the ***piling-up lemma***:

THEOREM 2 (Piling-up Lemma). *Let $(x_i)_{i=1,...,t}$ be independent random variables of which the value is 0 with probability p_i and 1 with probability $1-p_i$. Then the probability that $x_1 \oplus x_2 \oplus ... \oplus x_t = 0$ is*

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

Therefore if we assume the t linear relations to be independent, their sum's bias is

$$\varepsilon = 2^{t-1} \cdot \prod_{i=1}^{t} \varepsilon_i. \tag{10}$$

The distinguishing algorithm is very simple:

(1) Obtain a certain number of plaintext-ciphertext pairs (P_i, C_i).
(2) Estimate the bias of $\mu_P \bullet P \oplus \mu_C \bullet C$ by evaluating it for all (P_i, C_i). Let ε' be the estimated bias. If ε' is close to ε, conclude that pairs (P_i, C_i) have been generated by the block cipher. If it is close to 0, conclude that a random permutation has been used. Determining the threshold is a statistical *hypothesis testing* problem.

In order for the distinguisher to be efficient, the number m of pairs (P_i, C_i) must be $\Theta(\varepsilon^{-2})$. Note that it is a *known*-plaintext attack, as any pair (P_i, C_i) can be exploited (however chosen plaintext variants can improve the attack's complexity, see [97]).

This distinguisher can be used to retrieve key bits using a 2R-attack, as described in section I.3: the attacker finds a good linear equation Λ holding for the whole cipher except the first and last rounds:

$$\left(\bigoplus_{i=1}^{a} X_1^{(i)} \right) [k_1^{(1)}, ..., k_1^{(\alpha)}] \oplus \left(\bigoplus_{i=1}^{b} X_{R-1}^{(i)} \right) [k_R^{(1)}, ..., k_R^{(\beta)}] = \bigoplus_{i=1}^{c} k^{(i)} \tag{11}$$

where $k_1^{(1)}, ..., k_1^{(\alpha)}$ are the round key bits needed to evaluate $\bigoplus_{i=1}^{a} X_1^{(i)}$, and $k_R^{(1)}, ..., k_R^{(\beta)}$ the ones needed to evaluate $\bigoplus_{i=1}^{b} X_{R-1}^{(i)}$. $k^{(1)}, ..., k^{(c)}$ are other round key bits used somewhere between round 2 and $R-1$.

Then she obtains a set of pairs $\{(P_i, C_i)\}_{i=1,...,m}$. She guesses bits $k_1^{(1)}, ..., k_1^{(\alpha)}, k_R^{(1)}, ..., k_R^{(\beta)}$ and estimates the bias of the left side of (11) for these m pairs. The most likely candidate for $(k_1^{(1)}, ..., k_1^{(\alpha)}, k_R^{(1)}, ..., k_R^{(\beta)})$ is the one with the highest estimated bias; the second most likely is the one with the second highest estimated bias, and so on. Note that the value of $\bigoplus_{i=1}^{c} k^{(i)}$ is also obtained. Then for each candidate, from the most to the less likely, the remaining key bits are exhaustively searched, until the right key is found.

Note that it is desirable for the number $a + b$ of bits in the input and output masks to be small enough, so that the number $\alpha + \beta$ of round key bits on which the guess is made is also small enough.

From the designer's point of view, one strategy to prevent efficient linear cryptanalysis is to upper-bound the bias for all linear relations of a S-box. We define the ***λ-parameter*** of a $p \times q$ S-box S as

$$\lambda_S = \max_{\substack{\alpha \in \mathbb{Z}_2^p \setminus \{0\} \\ \beta \in \mathbb{Z}_2^q \setminus \{0\}}} |c(l_\alpha, l_\beta \circ S)|. \tag{12}$$

Thus roughly speaking, the λ-parameter is equal to two times the maximal bias of the S-box.

Also, the more S-boxes the attacker must approximate to obtain a linear relation over the whole cipher, the more difficult it will be to find a good linear relation. Therefore maximizing the ***linear branch number*** [146, 46] of a linear transform θ is also a good strategy for the designer to prevent linear attacks.

DEFINITION 3. The ***block Hamming weight*** $W_i(x)$ of a bit string x with respect to a partition of x in blocks of i bits is its number of non-zero blocks.

DEFINITION 4. The ***linear branch number*** of a linear transform θ is defined as

$$\mathcal{B}_l(\theta) = \min_{\substack{\alpha \neq 0, \beta \\ \alpha \cdot x = \beta \cdot \theta(x)}} W_i(\alpha) + W_i(\beta).$$

The length of the blocks considered in this context is equal to the size of the S-boxes. If we describe θ using a matrix M: $\theta(x) = Mx$, then the linear branch number can equivalently be expressed as

$$\mathcal{B}_l(\theta) = \min_{\alpha \neq 0} W_i(\alpha) + W_i(M^T \alpha).$$

I.5. Differential Cryptanalysis

Differential cryptanalysis has been introduced by E. Biham and A. Shamir in 1991 [19]. It is based on identifying a plaintext difference Δ_I which results in a ciphertext difference Δ_O with some "high" probability p.

By "high" we mean: "much higher than $1/2^n$". As a matter of fact, $1/2^n$ is the mean probability that some differential (Δ_I, Δ_O) holds for a random permutation of $\mathbf{P}_n$.

Similarly to the case of linear cryptanalysis, such a differential is found by first evaluating the probability of all possible differentials for a single S-box, and then trying to join these differences together in order to finally obtain a differential (Δ_I, Δ_O) for the whole cipher. A table giving

the probability associated to all pairs of input and output differences of an S-box is called ***XOR distribution table***. Moreover we can define a parameter measuring the resistance of S-boxes to differential cryptanalysis, as we did for linear cryptanalysis; the ***δ-parameter*** of a $q \times r$ S-box S is the probability of the best differential through this S-box. More formally

$$\delta_S = 2^{-q} \cdot \max_{\substack{0 \neq a \in \mathbb{Z}_2^q \\ b \in \mathbb{Z}_2^r}} |\{x \in \mathbb{Z}_2^q | S[x \oplus a] \oplus S[x] = b\}|. \tag{13}$$

Moreover the ***differential branch number*** [146, 46] of a linear transform plays in this context the same role as the linear branch number does against linear cryptanalysis:

DEFINITION 5. The ***differential branch number*** of a linear transform θ is defined as

$$\mathcal{B}_d(\theta) = \min_{x \neq 0} \; W_i(x) + W_i(\theta(x)).$$

The distinguishing algorithm is the following:

(1) Encrypt m pairs $(P_i; P_i^*)$ of plaintexts with $P_i \oplus P_i^* = \Delta_I$.
(2) Among the corresponding pairs $(C_i; C_i^*)$ of ciphertexts, count the number N of them for which $C_i \oplus C_i^* = \Delta_O$. Then use hypothesis testing to obtain a decision: if N/m is close to p, conclude that the function that outputted the ciphertexts was an instance of the block cipher. If it is close to 2^{-n}, conclude that it was a random permutation.

The data complexity of the algorithm is $\Theta(p^{-1})$. As pairs with a given difference must be chosen, it is a chosen plaintext attack. However if enough known plaintexts are available, the attacker can find enough pairs with the required difference among them. As in the case of linear cryptanalysis, a differential distinguisher can be used in the context of a lR-attack.

I.6. The DES and the AES

The two best-known block ciphers are the DES (*Data Encryption Standard*) and the AES (*Advanced Encryption Standard*).

A preliminary version of DES was submitted by IBM in 1974 in response to a public call of the U.S. National Bureau of Standards[8]. After a review by the NSA[9] resulting in a few changes, it was adopted as a U.S. Federal Information Processing Standard in 1977 (FIPS 46 [164]).

[8]NBS, now NIST, National Institute of Science and Technology.
[9]the U.S. National Security Agency.

The DES algorithm is a 16-round Feistel Network operating on blocks of 64 bits. Its key size is 56 bits. Some attacks exist against DES which are not really practical, such as linear cryptanalysis (see section I.4), which requires 2^{39} known plaintexts to be successful [113, 82]. However the major security concern regarding DES is its small key size. Nowadays a key space of size 2^{56} can be searched exhaustively[10]. It is why current implementations of DES use it in triple encryption mode. It means that three independently-keyed DES are applied one after the other, the second one being a decryption:

$$C = \mathrm{DES}_{K_3}(\mathrm{DES}^{-1}_{K_2}(\mathrm{DES}_{K_1}(P))). \tag{14}$$

We speak of ***3-key Triple-DES***. Sometimes K_3 is equal to K_1; we then speak of ***2-key Triple-DES***.

Because of DES's small key size, the NIST published a request for candidates to become the next encryption standard in 1997. The candidates should support block size of 128 bits and key sizes of 128, 192 and 256 bits. Moreover, there should not be any shortcut attack against them allowing a retrieval of the key faster than exhaustive search. 15 submissions were originally accepted. In 1999, NIST selected five of them as finalists: Mars [32], Rijndael [47], RC6 [149], Serpent [2], and Twofish [154]. In October 2000, NIST decided after three years of a public review process to select the Rijndael algorithm, developed by the Belgian researchers J. Daemen and V. Rijmen, as the new AES (FIPS 197).

Rijndael is a substitution-permutation network. Its 128-bit key 128-bit block version has 10 rounds. The currently best known attacks against Rijndael deal with reduced-round versions (7 or 8 rounds) of it. They are due to N. Ferguson et al. [55] and H. Gilbert and M. Minier [61]. Nowadays there is no attack on the full-round Rijndael faster than exhaustive search.

As references to DES and AES will be made in several chapters, we give their description in Appendix B.

I.7. Conclusion

We have presented the basic concepts in block cipher analysis and design. Note that several attacks, as well as several variants to linear and differential cryptanalysis, have not been covered.

In the first part of this thesis, a few topics will be discussed in more details. Chapter II deals with distinguishers having unlimited computation capabilities. Chapter III discusses the *square attack*, while Chapter IV extensively discusses attacks on the key schedule of block ciphers.

[10] In fact only 2^{55} trial encryptions are needed, due to complementation properties of DES.

The second part of the thesis will first deal with physical attacks on computing devices through Chapters V (faults attacks) and VI (probing attacks); finally in Chapter VII we present the design of a block cipher deliberately optimized for hardware efficiency.

Part 1

Security Proofs and Cryptanalysis

CHAPTER II

Security Proofs in the Luby-Rackoff Model

- Introducing Random Involutions -

Abstract. In this chapter we deal with security proofs of block ciphers in a very specific context: the Luby-Rackoff model. After a brief description of the model, of the proof techniques in this model, and of previous results, we focus on the Misty structure. Two schemes are considered: the *L-scheme*, which is widely used in the Misty [114] block cipher, and the *R-scheme*, which is its inverse. We show that the previously known security bounds [121] on these schemes remain essentially the same when the inner random permutations are replaced by random *involutions* (i.e. permutations c such that $c(c(x)) = x$) without fixed points ($\nexists x : c(x) = x$).
These results have originally been published in [142]. It is the first paper to consider involutions in the context of the Luby-Rackoff model.

II.1. Introduction

Proving the security of block ciphers has been a long-standing problem, and it is not solved yet. Usually, the "proof" is limited to showing or arguing that the cipher is resistant against the well-known attacks (linear and differential cryptanalysis, square attack (see Chapter III), slide attack (see Chapter IV),...). The seminal paper of M. Luby and C. Rackoff [108] is an attempt to build some kind of generic proof on the security of constructions[1]. However as we will see it has severe limitations.

The Luby-Rackoff model deals with distinguishers whose aim is to differentiate whether a given function has been randomly drawn in accordance with the uniform probability distribution, or in accordance with another (given) distribution. As a matter of fact, indistinguishability from a random permutation family is one of the strongest security notions one can expect from a block cipher. The specificity of the distinguishers considered in the Luby-Rackoff model is that they are assumed to have

[1] Although we do not deal with them in this thesis, there are other ways to obtain partial security proofs on block ciphers. Knudsen and Nyberg [127] analyzed resistance against differential cryptanalysis. Thereafter Vaudenay introduced the theory of decorrelation [167, 168] which deals with security proofs in a wider setting.

unbounded computation capabilities. In this context, indistinguishability against adversaries allowed to make adaptively chosen encryption queries is called *pseudorandomness*. When adaptively chosen decryption queries are allowed as well, we speak of *superpseudorandomness*. Equivalence between these notions and other security notions for symmetric ciphers such as *semantic security*, *left-or-right indistinguishability*, or *find-then-guess indistinguishability* [10] has been proven in [49, 138, 68].

The random functions considered in the framework of the Luby-Rackoff model are usually obtained from actual block cipher structures (such as the Feistel Network), by replacing the round functions by completely random functions or permutations. Thus the validity of these security proofs is relatively limited, as it does not deal with actual block cipher round functions. On the other hand, these proofs ensure that the structure used to construct a cipher is not flawed from the beginning. Furthermore, trying to prove a given security level for a structure sometimes results in finding an attack against it!

In this chapter, we deal with two schemes which originate in the Misty block cipher [114]. Security proofs for these schemes have already been given in [121, 77, 76]. The originality of our work is to consider the case where the round functions are *involutions without fixed points*. The motivation of this hypothesis originates in the fact that involutions are more and more used in the construction of block ciphers, for implementation reasons; examples are the block ciphers KHAZAD [8], Anubis [7], Noekeon [45], or `ICEBERG` [161](see Chapter VII). Furthermore, the fact that involutions are used could potentially lead to attacks, as suggested by A. Biryukov [21]. We will show that the security bounds on the Misty schemes proved in [121] remain essentially the same when the inner permutations are replaced by involutions without fixed points.

II.2. Basic Concepts and Notations

Formally, a ***pseudorandom distinguisher*** is defined as follows:

Definition 6. Let $N_1, N_2 > 1$. A ***pseudorandom distinguisher*** is a deterministic algorithm $\mathcal{A}$ with unbounded (but finite) computation capabilities, which given a function $F : \mathbb{Z}_2^{N_1} \to \mathbb{Z}_2^{N_2}$ can query it by asking values $x \in \mathbb{Z}_2^{N_1}$ of which it obtains the image $y = F(x)$. Depending on the answers $y \in \mathbb{Z}_2^{N_2}$ it obtains, $\mathcal{A}$ outputs either 0 or 1.

It follows from the definition that pseudorandom distinguishers are allowed to make *adaptively chosen* encryption queries. In practice we will always deal with functions of same domain and range. Thus $N = N_1 = N_2$. As a shortcut, we speak of a *random function* to designate a function randomly drawn in accordance with a fixed probability distribution. We also define a ***perfect*** *random function* (resp. permutation)

to be a random function (resp. permutation) drawn in accordance with the *uniform* distribution.

The following notations will be used in the remaining of this chapter:

- I_n denotes the $\mathbb{Z}_2^n$ set.
- $\mathbf{F}_{n,p}$ is the set of all functions from I_n to I_p.
- $\mathbf{P}_n$ is the set of all permutations on I_n.
- The transition probability $\mathcal{P}[\mathbf{x} \overset{F}{\mapsto} \mathbf{y}]$, where F is a random function in $\mathbf{F}_{n,p}$, $\mathbf{x} \in I_n^m$,$\mathbf{y} \in I_p^m$, is the probability that $\forall i \in \{1, ..., m\} : F(x_i) = y_i$. With an abuse of notation, this event is sometimes written $F(\mathbf{x}) = \mathbf{y}$.
- $\mathcal{A}^f$ denotes the output of $\mathcal{A}$ when querying function f.
- m is the number of queries made by $\mathcal{A}$.
- F^* (resp. C^*) denotes a perfect random function F (resp. permutation C). In some contexts, we also use $F \in_R \mathbf{F}_{n,p}$ to denote a perfect random function.

We define the ***advantage*** a distinguisher $\mathcal{A}$ has in distinguishing a random function F from a perfect random function F^*:

Definition 7. Let F be a random function, and F^* be a *perfect* random function. The ***advantage*** a pseudorandom distinguisher $\mathcal{A}$ has in distinguishing F from F^* is

$$\mathrm{Adv}_{\mathcal{A}}(F, F^*) := |\mathcal{P}[\mathcal{A}^{F^*} = 1] - \mathcal{P}[\mathcal{A}^F = 1]|.$$

In the following we will note $p^* = \mathcal{P}[\mathcal{A}^{F^*} = 1]$ and $p := \mathcal{P}[\mathcal{A}^F = 1]$.

Assume we flip a coin to obtain a random bit b, and provide $\mathcal{A}$ with a random function F if $b = 0$, and with a perfect random function F^* if $b = 1$. If we denote by b' the output of $\mathcal{A}$ we have:

$$\begin{aligned} \mathrm{Adv}_{\mathcal{A}}(F, F^*) =& |\mathcal{P}[\mathcal{A}^{F^*} = 1] - \mathcal{P}[\mathcal{A}^F = 1]| \\ =& |\mathcal{P}[\mathcal{A}^{F^*} = 1] - 1 + \mathcal{P}[\mathcal{A}^F = 0]| \\ =& |2 \cdot \mathcal{P}[b = b'] - 1| \end{aligned} \tag{15}$$

which is a much more intuitive view of the advantage.

Pseudorandom distinguishers as defined above are allowed to make encryption queries only. ***Superpseudorandom distinguishers*** are allowed to make decryption queries as well:

Definition 8. Let $N > 1$. A ***superpseudorandom distinguisher*** is a deterministic algorithm $\mathcal{A}$ with unbounded (but finite) computation capabilities, which can query a given permutation $C \in \mathbf{P}_N$ by providing it with values $x \in I_N$ of which it obtains to its choosing either the image $y = C(x)$, or the inverse image $y = C^{-1}(x)$. Depending on the answers $y \in I_N$ it obtains, $\mathcal{A}$ outputs either 0 or 1.

The advantage a superpseudorandom distinguisher has in distinguishing a random permutation C from a perfect random permutation C^* is defined similarly to the case of pseudorandom distinguishers.

The non-perfect random functions we want to distinguish from the perfect ones are practically built by embedding perfect random functions $f_1^*, f_2^*, ..., f_r^*$ into a global structure Φ. The domain and range of $f_1^*, ..., f_r^*$ have variable size; it is smaller than the size of the domain and range of $\Phi(f_1^*, ..., f_r^*)$. Such a structure Φ is sometimes called ***function*** (or ***permutation***) ***generator***. An example is the Feistel structure considered in section II.3; to r functions $f_1, f_2, ..., f_r \in \mathbf{F}_{n,n}$ it associates a permutation $F = \phi(f_1, f_2, ..., f_r) \in \mathbf{P}_{2n}$. A proof of security for such a structure in the Luby-Rackoff model consists in upper-bounding the advantage for all possible distinguishers $\mathcal{A}$ as a function of the number of queries m and the block size $2n$.

Consider a function generator Φ. Let N denote its block size. Φ is said ***pseudorandom*** if for all pseudorandom distinguishers $\mathcal{A}$ of which the number of queries m is polynomial in N, the advantage remains "polynomially small" (for N big enough). More formally:

Definition 9. A function generator Φ is ***pseudorandom*** if for all polynomials $P(N), Q(N)$, there is an integer $N_0 \in \mathbb{N}$ such that:
$\forall N \geq N_0$, for all pseudorandom distinguishers $\mathcal{A}$ allowed to make $m \leq Q(N)$ queries,

$$\mathrm{Adv}_{\mathcal{A}}(\Phi(f_1^*, ..., f_r^*), F^*) \leq \frac{1}{P(N)}.$$

Superpseudorandom permutations generators are defined similarly with respect to superpseudorandom distinguishers.

At this point, one could wonder why we restrict ourselves to *deterministic* distinguishers, while we could also consider probabilistic algorithms using a random tape to generate a value $\omega \in \Omega$ (without loss of generality, the probability distribution on Ω is assumed to be uniform), of which the choices x submitted to the encryption oracle and the final output depend. Let $\mathcal{A}$ be such a probabilistic distinguisher. Then probabilities p and p^* are computed not only on the distributions $(\mathcal{P}[F = \overline{F}])_{\overline{F} \in \mathbf{F}_{n,p}}$ and $(\mathcal{P}[F^* = \overline{F}])_{\overline{F} \in \mathbf{F}_{n,p}}$ anymore, but also on the probability distribution of $\omega \in \Omega$. Note that fixing ω to a chosen value $\bar{\omega}$ defines a deterministic distinguisher, let $\mathcal{A}_{\bar{\omega}}$. We also define $p_{\bar{\omega}} = \mathcal{P}[\mathcal{A}_{\bar{\omega}}^F = 1]$ and $p_{\bar{\omega}}^* = \mathcal{P}[\mathcal{A}_{\bar{\omega}}^{F^*} = 1]$.

It is now easy to show that there exists a $\bar{\omega}$ such that $\mathcal{A}_{\bar{\omega}}$ achieves an advantage $|p_{\bar{\omega}} - p_{\bar{\omega}}^*|$ equal or better than the one of $\mathcal{A}$. *Ex absurdo*, let us assume it is not the case, i.e. $\forall \bar{\omega} \in \Omega : |p_{\bar{\omega}} - p_{\bar{\omega}}^*| < |p - p^*|$. Then we have:

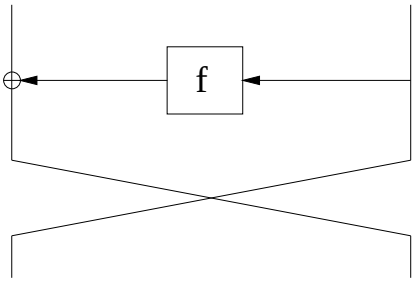


FIGURE 1. One round of a Feistel structure.

$$\begin{aligned} |p - p^*| &= \frac{|\sum_{\bar{\omega}\in\Omega} p_{\bar{\omega}} - \sum_{\bar{\omega}\in\Omega} p^*_{\bar{\omega}}|}{|\Omega|} \\ &= \frac{|\sum_{\bar{\omega}\in\Omega}(p_{\bar{\omega}} - p^*_{\bar{\omega}})|}{|\Omega|} \\ &\leq \frac{\sum_{\bar{\omega}\in\Omega}|p_{\bar{\omega}} - p^*_{\bar{\omega}}|}{|\Omega|} < |p - p^*| \end{aligned}$$

which is a contradiction.

The conclusion is that we gain nothing by considering probabilistic distinguishers: deterministic ones perform as well.

II.3. An Introductory Example: the 3-round Feistel Structure

In this section we give a demonstration of the pseudorandom character of the 3-round Feistel structure. The initial demonstration of this result has been given in [108], but a clearer demonstration appeared in [115]. The demonstration we give here is very similar to this last. We choose to give it in this thesis in order to give the reader an intuitive view of the fundament of this type of result. As a matter of fact, in section II.4 efficient tools will be given to prove them, but they are at the price of loosing a part of the intuition.

II.3.1. Description of the Feistel Structure

Remember that a 1-round Feistel is a $2n$-bit permutation ϕ taking a n-bit function f as a round function and such that

$$\phi(f)(\langle L, R\rangle) = \langle R, L \oplus f(R)\rangle.$$

It is depicted in Figure 1. An r-round Feistel structure is simply the composition of r 1-round Feistel, transforming r n-bit functions $f_1, ..., f_r$ into a $2n$-bit permutation:

$$\phi(f_1, f_2, ..., f_r) = \phi(f_r) \circ ... \circ \phi(f_1).$$

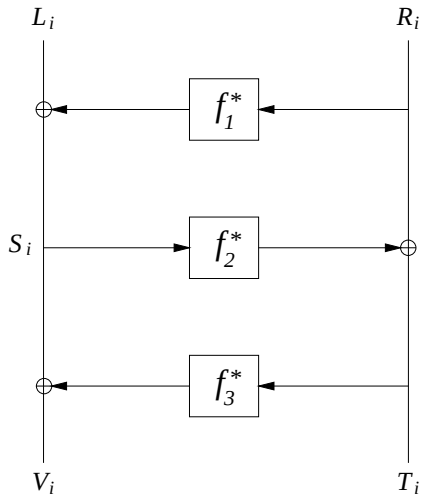


FIGURE 2. 3 rounds of a Feistel structure.

II.3.2. The 3-round Feistel Structure is Pseudorandom

We prove the following theorem:

THEOREM 10. *Let $f_1^*, f_2^*, f_3^* \in \mathbf{F}_{n,n}$ be independent perfect random functions. Let $F^* \in \mathbf{F}_{2n,2n}$ be a perfect random function. For any pseudorandom distinguisher $\mathcal{A}$ allowed to make m adaptive encryption queries,we have*

$$Adv_{\mathcal{A}}(\phi(f_1^*, f_2^*, f_3^*), F^*) \leq \frac{m(m-1)}{2^n}.$$

PROOF. Consider a pseudorandom distinguisher $\mathcal{A}$ allowed to make m adaptive encryption queries. Without loss of generality, we assume $\mathcal{A}$ never makes two times the same query. As a matter of fact, it will learn nothing more by repeating a query.

Let $\langle L_i, R_i \rangle$ denote the i^{th} query of $\mathcal{A}$. Let $\langle V_i, T_i \rangle$ be the corresponding output. Moreover we define the intermediate state $S_i = L_i \oplus f_1^*(R_i)$ (see Figure 2). As S_i, V_i, T_i depend on the output of random functions, they are random variables. For $i = 2, ..., m$, $\langle L_i, R_i \rangle$ is a random variable as well, as it is a function of the previously obtained outputs $\langle V_j, T_j \rangle$ for $j = 1, ..., i-1$ (remember that $\mathcal{A}$ is an adaptive distinguisher). Let $\mathcal{E}_S$ denote the event that $S_1, ..., S_m$ are all distinct, and $\mathcal{E}_T$ denote the event

that $T_1, ..., T_m$ are all distinct. Then we have:

$$\begin{aligned}\mathcal{P}[\mathcal{E}_S \wedge \mathcal{E}_T] &= 1 - \mathcal{P}[\overline{\mathcal{E}_S} \vee \overline{\mathcal{E}_T}] \geq 1 - \mathcal{P}[\overline{\mathcal{E}_S}] - \mathcal{P}[\overline{\mathcal{E}_T}] \\ &\geq 1 - \mathcal{P}[\bigvee_{i<j} S_i = S_j] - \mathcal{P}[\bigvee_{i<j} T_i = T_j] \\ &\geq 1 - \sum_{i<j} \mathcal{P}[S_i = S_j] - \sum_{i<j} \mathcal{P}[T_i = T_j]\end{aligned} \tag{16}$$

For given $i < j$:

- Either $R_i = R_j$, which implies $L_i \neq L_j$ (as the distinguisher is assumed not to repeat the same query), and thus $S_i \neq S_j$.
- Or $R_i \neq R_j$. Then $f_1^*(R_i)$ and $f_1^*(R_j)$ are random and independent, thus $\mathcal{P}[S_i = S_j] = \mathcal{P}[L_i \oplus f_1^*(R_i) = L_j \oplus f_1^*(R_j)] = 2^{-n}$.

We can apply the same reasoning to the T_i. Therefore

$$\mathcal{P}[\mathcal{E}_S \wedge \mathcal{E}_T] \geq 1 - \frac{m(m-1)}{2^n}. \tag{17}$$

When $\mathcal{E}_S$ and $\mathcal{E}_T$ both occur, then $V_1 = S_1 \oplus f_3^*(T_1)$, ..., $V_m = S_m \oplus f_3^*(T_m)$ and $T_1 = R_1 \oplus f_2^*(S_1)$, ..., $T_m = R_m \oplus f_3^*(S_m)$ are completely random, because f_1^* and f_2^* are random functions. Thus they are indistinguishable from the output of a random function.

Therefore the distinguishing probability is upper-bounded by $\frac{m(m-1)}{2^n}$, as announced. □

The basic principle of the proof is to prove that $\phi(f_1^*, f_2^*, f_3^*)$ behaves randomly "almost always". The "almost" permits for a distinguisher to achieve a non-zero advantage. The philosophy underlying to the coefficient H method suggested by Patarin in order to prove pseudorandomness (see section II.4) is not different.

II.3.3. Other Results on the Feistel Structure

The Feistel structure is the most widely studied in the Luby-Rackoff model. On the one hand, its security bounds were tried to be improved [130, 132, 133, 134, 135]; a good summary of the best current results can be found in [135]. On the other hand, slightly modified constructions were examined. For example, constructions were some of the round functions are identical [131], or are replaced by hash functions [109, 125], were considered. Feistel networks were exclusive or is replaced by another group operation are also considered in [136].

II.4. The Coefficient H Method

It seems natural that the distinguishing probability of the best distinguisher is bound to the probability $\mathcal{P}[\mathbf{x} \overset{F}{\mapsto} \mathbf{y}]$. As a matter of fact, if $\forall \mathbf{x}, \mathbf{y} \in I_N^m : \mathcal{P}[\mathbf{x} \overset{F}{\mapsto} \mathbf{y}] = 1/2^{Nm}$ as it is the case for a perfectly random function, F cannot be distinguished from such a function. If now probability distribution $\mathcal{P}[\mathbf{x} \overset{F}{\mapsto} \mathbf{y}]$ is "close enough" to the uniform distribution, it should be possible to draw some conclusion on the best distinguishing probability. Several results were stated by J. Patarin regarding this problem [130, 131]. They are known as the "coefficient H method", because J. Patarin denoted by $H(\mathbf{x}, \mathbf{y})$ the number of functions F such that $F(\mathbf{x}) = \mathbf{y}$; up to a constant factor, this number is equal to the transition probability $\mathcal{P}[\mathbf{x} \overset{F}{\mapsto} \mathbf{y}]$.

In this section we describe J. Patarin's main theorems on the subject. Although it is already done in other works [130, 120], we give a proof of the first one, as we think it helps the reader to have a better understanding. Demonstration of the other theorems is similar.

Before going to the theorems themselves, we restate a few observations about distinguishers:

- The distinguishers considered being adaptive and deterministic, the vector $\mathbf{x} \in I_N^m$ of queries is a function of the vector $\mathbf{y} \in I_N^m$ of answers (only). We can thus write it as $\mathbf{x}(\mathbf{y})$. Note that in the case of a superpseudorandom distinguisher, the direction of the queries (encryption or decryption) is determined by $\mathbf{y}$ as well.
- Similarly, the output of a distinguisher $\mathcal{A}$ is also a function of $\mathbf{y} \in I_N^m$ only.
- Without loss of generality, we may assume that the distinguisher never makes two times the same query. As a matter of fact, a distinguisher would learn nothing more by repeating a query. We denote by $\mathcal{X}$ the subset of I_N^m such that $\forall \mathbf{x} \in \mathcal{X} : \forall i \neq j : x_i \neq x_j$.

A first theorem is the following. We give it in its full generality, i.e. when the domain and range of the functions considered may be different.

THEOREM 11 (Patarin). *Let $F \in \mathbf{F}_{N_1,N_2}$ be a random function; let $F^* \in \mathbf{F}_{N_1,N_2}$ be a perfect random function. Let m be an integer. If there exists a subset $\mathcal{Y}$ of $I_{N_2}^m$ and two positive real numbers ϵ_1 and ϵ_2 such that*

(1) $|\mathcal{Y}| \geq (1 - \epsilon_1) \cdot |I_{N_2}|^m$

(2) $\forall \mathbf{x} \in \mathcal{X} \subset I_{N_1}^m \quad \forall \mathbf{y} \in \mathcal{Y} : \mathcal{P}[\mathbf{x} \overset{F}{\mapsto} \mathbf{y}] \geq (1 - \epsilon_2) \cdot \frac{1}{|I_{N_2}|^m}$

Then for any distinguisher $\mathcal{A}$ using m encryption queries

$$Adv_{\mathcal{A}}(F, F^*) \leq \epsilon_1 + \epsilon_2.$$

PROOF. Consider the subset $\mathcal{S}$ of elements of $I_{N_2}^m$ for which the distinguisher answers 1. Then one can write:

$$\begin{cases} p^* = \sum_{\mathbf{y} \in \mathcal{S}} \mathcal{P}[\mathbf{x}(\mathbf{y}) \overset{F^*}{\mapsto} \mathbf{y}] \\ p = \sum_{\mathbf{y} \in \mathcal{S}} \mathcal{P}[\mathbf{x}(\mathbf{y}) \overset{F}{\mapsto} \mathbf{y}] \end{cases} \tag{18}$$

Then

$$\begin{aligned} p^* - p = &\sum_{\mathbf{y} \in \mathcal{S} \cap \mathcal{Y}} \mathcal{P}[\mathbf{x}(\mathbf{y}) \overset{F^*}{\mapsto} \mathbf{y}] + \sum_{\mathbf{y} \in \mathcal{S} \cap \mathcal{Y}} (-\mathcal{P}[\mathbf{x}(\mathbf{y}) \overset{F}{\mapsto} \mathbf{y}]) \\ &+ \sum_{\substack{\mathbf{y} \in \mathcal{S} \\ \mathbf{y} \notin \mathcal{Y}}} \mathcal{P}[\mathbf{x}(\mathbf{y}) \overset{F^*}{\mapsto} \mathbf{y}] + \sum_{\substack{\mathbf{y} \in \mathcal{S} \\ \mathbf{y} \notin \mathcal{Y}}} (-\mathcal{P}[\mathbf{x}(\mathbf{y}) \overset{F}{\mapsto} \mathbf{y}]). \end{aligned} \tag{19}$$

$\mathcal{P}[\mathbf{x}(\mathbf{y}) \overset{F^*}{\mapsto} \mathbf{y}]$ is trivially equal to $1/|I_{N_2}|^m$. Also, hypothesis (2) implies that $\forall \mathbf{y} \in \mathcal{Y} : -\mathcal{P}[\mathbf{x}(\mathbf{y}) \overset{F}{\mapsto} \mathbf{y}] \leq (\epsilon_2 - 1) \cdot 1/|I_{N_2}|^m$. Finally, the fourth term is less than 0. Then equation (19) implies

$$\begin{aligned} p^* - p &\leq \sum_{\mathbf{y} \in \mathcal{S} \cap \mathcal{Y}} \epsilon_2/|I_{N_2}|^m + \sum_{\substack{\mathbf{y} \in \mathcal{S} \\ \mathbf{y} \notin \mathcal{Y}}} 1/|I_{N_2}|^m \\ &\leq \epsilon_2 + |\complement\mathcal{Y}| \cdot 1/|I_{N_2}|^m, \end{aligned} \tag{20}$$

where $\complement\mathcal{Y}$ denotes the complementary of subset $\mathcal{Y}$.

From hypothesis (1) we have

$$|\complement\mathcal{Y}| = |I_{N_2}|^m - |\mathcal{Y}| \leq |I_{N_2}|^m + (\epsilon_1 - 1)|I_{N_2}|^m = \epsilon_1 |I_{N_2}|^m. \tag{21}$$

Thus finally: $p - p^* \leq \epsilon_1 + \epsilon_2$.

Consider now the inverse distinguisher, i.e. the one that answers 1 if and only if the original one answers 0. Associated probabilities are $p'^* = 1 - p^*$ and $p' = 1 - p$. Applying to it the same reasoning as above, we obtain $p' - p'^* \leq \epsilon_1 + \epsilon_2$, which is equivalent to $p - p^* \leq \epsilon_1 + \epsilon_2$.

We conclude that $|p - p^*| = \mathrm{Adv}_{\mathcal{A}}(F, F^*) \leq \epsilon_1 + \epsilon_2$, as announced. □

Theorem 11 has dealt with random functions. The following theorem deals with *permutations*. It will be useful as well to prove our results regarding the Misty structure.

THEOREM 12 (Patarin). *Let $C \in \mathbf{P}_N$ be a random permutation; let $C^* \in \mathbf{P}_N$ be a perfect random permutation. Let m be an integer, and $\epsilon > 0$. If for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$: $\mathcal{P}[\mathbf{x} \overset{f}{\mapsto} \mathbf{y}] \geq (1-\epsilon) \cdot \frac{1}{|I_N|^m}$, then for any distinguisher $\mathcal{A}$ allowed to make m encryption or decryption queries*

$$Adv_{\mathcal{A}}(C, C^*) \leq \epsilon + \frac{m(m-1)}{2 \cdot 2^N}.$$

II.5. The Misty Scheme

II.5.1. Description of the Structures

In this section we will consider two different structures. The first one is the ***L-Scheme***; it is used at various levels in the design of the Misty [114] and Kasumi [1] block ciphers. The ***R-Scheme*** is (almost) its inverse (we follow the terminology used by H. Gilbert and M. Minier [121]).

We define a 1-round L-scheme as a $2n$-bit permutation ψ_L taking a n-bit permutation c as a round function and such that

$$\psi_L(c)(\langle L, R \rangle) = \langle R, c(L) \oplus R \rangle.$$

It is depicted in Figure 3. An r-round L-scheme is simply the composition of r 1-round L-schemes, transforming r permutations $c_1, ..., c_r \in \mathbf{P}_n$ into a $2n$-bit permutation:

$$\psi_L(c_1, c_2, ..., c_r) = \psi_L(c_r) \circ ... \circ \psi_L(c_1).$$

A 1-round R-scheme transforms a n-bit permutation c into a $2n$-bit permutation $\psi_R(c)$ too. It is defined as (see Figure 3)

$$\psi_R(c)(\langle L, R \rangle) = \langle c(L) \oplus R, c(L) \rangle.$$

The composition of r 1-round R-schemes is a r-round R-scheme

$$\psi_R(c_1, c_2, ..., c_r) = \psi_R(c_r) \circ ... \circ \psi_R(c_1).$$

In this chapter we sometimes consider variants of the ψ_L and ψ_R schemes, where the last XOR operation is omitted, as well as the last swap. We call them ψ'_L and ψ'_R.

REMARK 1. Cryptographically speaking, ψ'_L and ψ'_R are equivalent respectively to ψ_L and ψ_R.

REMARK 2. $\psi'_L(c_1, c_2, ..., c_r)$ and $\psi'_R(c_r^{-1}, c_{r-1}^{-1}, ..., c_1^{-1})$ are inverses of each other. It implies that their security against superpseudorandom distinguishers is the same, as these distinguishers are allowed to encryption as well as decryption queries.

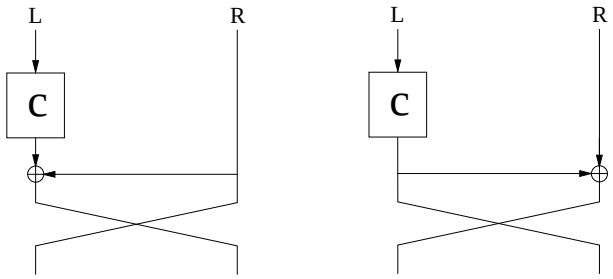


FIGURE 3. 1-round L-scheme at left, 1-round R-scheme at right.

II.5.2. Previous Results

II.5.2.1. *Sakurai-Zheng (1997).* The first results on pseudorandomness of the Misty schemes are due to K. Sakurai and Y. Zheng [153], who presented several negative results (i.e. non-pseudorandomness and non-superpseudorandomness) on the L-scheme:

- $\{\psi_L(f,g,h)|f,g,h \in_R \mathbf{P}_n\}$ is not pseudorandom.
- $\{\psi_L(f_1,f_2,f_3,f_4)|f_1,f_2,f_3,f_4 \in_R \mathbf{P}_n\}$ is not superpseudorandom.
- $\{\psi_L(f,f^2,f,f)|f \in_R \mathbf{P}_n\}$ is not pseudorandom.
- $\forall i,j \geq 0 : \{\psi_L(f^{i+j},f^j,f^i,f^i)|f \in_R \mathbf{P}_n\}$ is not pseudorandom.
- $\forall i,j \geq 0 : \{\psi_L(g,f^{i+j},f^j,f^i,f^i)|f \in_R \mathbf{P}_n\}$ is not pseudorandom.

These results can be proved by mounting simple attacks.

II.5.2.2. *Gilbert-Minier (2001).* Four years later H. Gilbert and M. Minier [121] presented the first positive results:

THEOREM 13 (Pseudorandomness of 4-round L-scheme).
Let n be an integer, $c_1,c_2,c_3,c_4 \in_R \mathbf{P}_n$ and $F^* \in_R \mathbf{F}_{2n,2n}$. Let $F = \psi_L(c_1,c_2,c_3,c_4) \in \mathbf{P}_{2n}$. Then for any adaptive pseudorandom distinguisher $\mathcal{A}$ allowed to make m queries we have

$$Adv^m_{\mathcal{A}}(F,F^*) \leq \frac{7}{2} \cdot \frac{m^2}{2^n}.$$

THEOREM 14 (Superpseudorandomness of 5-round L- and R-schemes).

Let n be an integer, $c_1,c_2,c_3,c_4,c_5 \in_R \mathbf{P}_n$ and $C^* \in_R \mathbf{P}_{2n}$. Let $C = \psi_L(c_1,c_2,c_3,c_4,c_5) \in \mathbf{P}_{2n}$ (resp. $C = \psi_R(c_1,c_2,c_3,c_4,c_5) \in \mathbf{P}_{2n}$). Then for any adaptive superpseudorandom distinguisher $\mathcal{A}$ allowed to make m queries we have

$$Adv^m_{\mathcal{A}}(C,C^*) \leq \frac{9}{2} \cdot \frac{m^2}{2^n}.$$

THEOREM 15 (Pseudorandomness of 3-round R-scheme).
Let n be an integer, $c_1,c_2,c_3 \in_R \mathbf{P}_n$ and $F^* \in_R \mathbf{F}_{2n,2n}$. Let $F =$

$\psi_R(c_1, c_2, c_3) \in \mathbf{P}_{2n}$. Then for any adaptive pseudorandom distinguisher $\mathcal{A}$ allowed to make m queries we have

$$Adv^m_{\mathcal{A}}(F, F^*) \leq \frac{3m^2}{2^n}.$$

On the other hand, Gilbert and Minier mounted an attack on a 4-round scheme, which requires only 2 encryption and 2 decryption queries. Thus such scheme is not superpseudorandom.

II.5.2.3. *Iwata et al. (2001-2002).* In addition to the proof of superpseudorandomness for the 5-round Misty structure, T. Iwata et al. consider in [77] the case where some of the round functions are *uniform ϵ-XOR universal permutations*. They also assume the distinguisher has oracle access to some of the round functions (following the model of Z. Ramzan and L. Reyzin [145]).

Definition 16. Let H_n be a keyed permutation family over $\mathbb{Z}_2^n$.

- H_n is ***uniform*** if $\forall x, y \in \mathbb{Z}_2^n, |\{h \in H_n : h(x) = y\}| = \frac{|H_n|}{2^n}$.
- H_n is ***ϵ-XOR universal*** if

$$\forall x \neq x' \in \mathbb{Z}_2^n, y \in \mathbb{Z}_2^n : |\{h \in H_n : h(x) \oplus h(x') = y\}| \leq \epsilon |H_n|.$$

Furthermore in [76] they show that the second round function of a 5-round Misty need not be cryptographic at all to achieve superpseudorandomness, it can be a public an constant function g satisfying the condition $\forall x \neq x' : g(x) \oplus x \neq g(x') \oplus x'$.

II.5.3. A Few Additional Notations

In the remaining of the section devoted to the Misty structure, we use the following notations:

- $I := I_n^m$ (remember that m is the number of plaintext-ciphertext pairs considered).
- For $\mathbf{x}, \mathbf{y} \in I$: $\mathbf{x} \sim \mathbf{y}$ informally means that $\mathbf{x}$ and $\mathbf{y}$ could be the inputs and outputs of a permutation. More formally: $\mathbf{x} \sim \mathbf{y} \Leftrightarrow \forall i, j \in \{1, 2, ..., m\} : X_i = X_j \Leftrightarrow Y_i = Y_j$.
- $I^{\neq} := \{\mathbf{x} \in I | \nexists i \neq j \in \{1, 2, ..., m\} : X_i = X_j\}$.
- $I^{=} := I \backslash I^{\neq}$.

II.5.4. The Gilbert-Minier Bounds on 4 rounds are Tight

In this section, we prove that the bounds given in theorems 13 and 15 are tight, by showing very simple attacks when $m \simeq 2^{n/2}$.

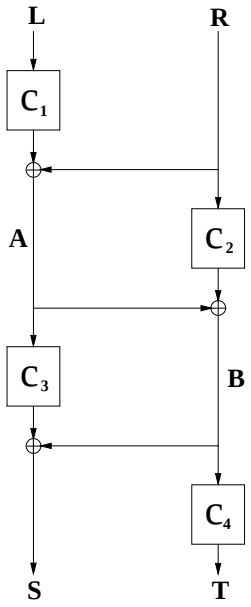


FIGURE 4. The 4-round L-scheme.

II.5.4.1. *The 4-Round L-Scheme.* The 4-round L-scheme (see Figure 4) has the following remarkable property:

PROPERTY 1. Consider a m-element input vector $\langle \mathbf{L}, \mathbf{R} \rangle$ to a function $\psi'_L(c_1, c_2, c_3, c_4)$, where $\mathbf{L} \in I^{\neq}$ and $\mathbf{R}$ is constant (i.e. $R_1 = R_2 = ... = R_m$). Then the corresponding output vector $\langle \mathbf{S}, \mathbf{T} \rangle$ is such that $\mathbf{T} \in I^{\neq}$.

PROOF. As $\mathbf{L} \in I^{\neq}$ and $\mathbf{R}$ is constant, we have $\mathbf{A} \in I^{\neq}$. Then $\mathbf{B} = \mathbf{A} \oplus c_2(\mathbf{R}) \in I^{\neq}$, and finally $\mathbf{T} \in I^{\neq}$. □

This property is exploited by the following distinguishing algorithm $\mathcal{A}$:

(1) Pick one value $R \in I_n$, and m different values $L_i \in I_n (i = 1, ..., m)$.
(2) Submit the m values $\langle L_i, R \rangle$ to the encryption oracle F. m outputs $\langle S_i, T_i \rangle$ are obtained.
(3) Look whether all values T_i are different. Output "1" if and only if it is the case.

If $F = \psi'_L(c_1^*, c_2^*, c_3^*, c_4^*)$, then $\mathcal{P}[\mathcal{A}^{\psi'_L(c_1^*, c_2^*, c_3^*, c_4^*)} = 1] = 1$. If F is a perfect random function F^*, then $\mathcal{P}[\mathcal{A}^{F^*} = 1] < 1 - 0.3\frac{m(m-1)}{2^n}$, as shown by Theorem 17.

THEOREM 17. *Let $(T_i)_{i=1,...,m} \in I_n^m$ be a random m-uple chosen in accordance with the uniform probability distribution. Then the probability that all T_i are different is smaller than $1 - 0.3\frac{m(m-1)}{2^n}$.*

PROOF. Let D_j be the event that there is no collision amongst $T_1, T_2, ..., T_j$. Then $\mathcal{P}[D_{j+1}|D_j] = 1 - \frac{j}{2^n}$. We can bound $\mathcal{P}[D_m]$ by:

$$\mathcal{P}[D_m] = \prod_{j=1}^{m-1} \mathcal{P}[D_{j+1}|D_j] = \prod_{j=1}^{m-1} \left(1 - \frac{j}{2^n}\right) \leq \prod_{j=1}^{m-1} e^{-j/2^n} = e^{-\sum_{j=1}^{m-1} j/2^n}$$

$$= e^{\frac{-m(m-1)}{2^{n+1}}} \leq 1 - \frac{(1-e^{-1})}{2} \cdot \frac{m(m-1)}{2^n} < 1 - 0.3\frac{m(m-1)}{2^n}.$$

where we applied the fact that for any x with $0 \leq x \leq 1$ we have $(1-e^{-1})x \leq 1 - e^{-x} \leq x$. □

Therefore

$$\mathrm{Adv}_{\mathcal{A}}(\psi'_L(c_1^*, c_2^*, c_3^*, c_4^*), F^*) > 0.3\frac{m(m-1)}{2^n}. \tag{22}$$

The conclusion is that the 4-round L-scheme can be distinguished from a perfect random function with a non negligible advantage as soon as the number of queries m is $\Theta(2^{n/2})$. Thus the bound given by H. Gilbert and M. Minier [121] is tight.

II.5.4.2. *The 4-Round R-Scheme.* This scheme satisfies a property similar to the one of the 4-round L-scheme:

PROPERTY 2. Consider a m-element input vector $\langle \mathbf{L}, \mathbf{R} \rangle$ to a function $\psi'_R(c_1, c_2, c_3, c_4)$, where $\mathbf{L}$ is constant (i.e. $L_1 = L_2 = ... = L_m$) and $\mathbf{R} \in I^{\neq}$. Then the corresponding output vector $\langle \mathbf{S}, \mathbf{T} \rangle$ is such that $\mathbf{S} \in I^{\neq}$.

Using this property, a distinguishing algorithm $\mathcal{A}$ can be built, such that $\mathrm{Adv}_{\mathcal{A}}(\psi'_R(c_1^*, c_2^*, c_3^*, c_4^*), F^*) > 0.3\frac{m\cdot(m-1)}{2^n}$. Thus once again the bound claimed by H. Gilbert and M. Minier [121] is tight.

II.5.5. Some Combinatorial Facts

Before proving new results on the Misty scheme, we give two useful properties about involutions.

PROPERTY 3. The number of involutions c from I_n to I_n without fixed points (i.e. $\forall x : c(x) \neq x$) is

$$\frac{2^n!}{2^{2^{n-1}} \cdot (2^{n-1})!}.$$

PROPERTY 4. The number of involutions c from I_n to I_n without fixed points, and such that a pairs (x, y) with $c(x) = y$ and $c(y) = x$ are fixed is

$$\frac{(2^n - 2a)!}{2^{2^{n-1}-a} \cdot (2^{n-1} - a)!}.$$

II.5.6. Pseudorandomness of 4R L-Scheme with Involutions

We now consider a 4-round L-scheme were the four permutations have been replaced by involutions without fixed points. In section II.5.7 we will prove this lemma:

LEMMA 1. *Let $m, n > 0$. Let $\langle \mathbf{L}, \mathbf{R} \rangle \in \mathcal{X} \subset I_{2n}^m, \langle \mathbf{S}, \mathbf{T} \rangle \in I \times I^{\neq}$. Then the probability for a 4-uple (c_1, c_2, c_3, c_4) of **involutions without fixed points** to satisfy $\psi'_L(c_1, ..., c_4)(\langle \mathbf{L}, \mathbf{R} \rangle) = \langle \mathbf{S}, \mathbf{T} \rangle$ is lower bounded by*

$$\left(1 - \frac{63m^2}{5 \cdot 2^n}\right) \cdot \frac{1}{2^{2nm}}.$$

It allows us to prove the following theorem:

THEOREM 18. *Let $c_1^*, ..., c_4^*$ be independent perfect random **involutions without fixed points** on I_n. Let $C := \psi_L(c_1^*, ..., c_4^*)$. Let $F^* \in \mathbf{F}_{2n,2n}$ be a perfect random function. Then for any pseudorandom distinguisher $\mathcal{A}$ allowed to make m queries, we have*

$$Adv_{\mathcal{A}}(C, F^*) < \frac{131m^2}{10 \cdot 2^n}.$$

Thus $\psi_L(c_1^*, ..., c_4^*)$ is pseudorandom, and secure as long as $m \ll 2^{n/2}$.

PROOF. It is an immediate application of theorem 11. The constraint $\mathbf{T} \in I^{\neq}$ in lemma 1 implies a non-zero ϵ_1. More precisely, ϵ_1 is equal to the probability for a (perfect) random $\mathbf{T} \in I$ to belong to $I^{=}$. It can be shown to be smaller than $\frac{m^2}{2 \cdot 2^n}$:

$$\mathcal{P}[\mathbf{T} \in I^{=}] = \mathcal{P}[\bigvee_{i<j} T_i = T_j] \leq \sum_{i<j} \mathcal{P}[T_i = T_j] \leq \frac{m^2}{2 \cdot 2^n}.$$

Lemma 1 gives the corresponding ϵ_2. □

The proof of lemma 1 will require the following lemma:

LEMMA 2. *Let $a \in]0, 1[$ and $b > 0$. Then $(1-a)^b > 1 - \frac{ba}{1-a}$.*

PROOF. $(1-a)^b = \exp(b \ln(1-a)) = \exp\left(-\sum_{k=1}^{\infty} \frac{ba^k}{k}\right)$

$$> \exp\left(-\sum_{k=1}^{\infty} ba^k\right) = \exp\left(\frac{-ba}{1-a}\right) > 1 - \frac{ba}{1-a}.$$

□

II.5.7. Proof of Lemma 1

For a given $(\mathbf{L}, \mathbf{R}, \mathbf{S}, \mathbf{T})$, we define λ and ρ as the number of independent equalities of the form $L_i = L_j$ and $R_i = R_j$ $(i \neq j)$, respectively.

We define two intermediate states during the computation of $\psi'_L(c_1, ..., c_4)$, namely (see Figure 4):

$$\mathbf{A} := c_1(\mathbf{L}) \oplus \mathbf{R}$$
$$\mathbf{B} := c_2(\mathbf{R}) \oplus \mathbf{A}$$

Let $\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle}$ be the probability that a random 4-uple (c_1, c_2, c_3, c_4) is such that $\psi'_L(c_1, c_2, c_3, c_4)(\langle \mathbf{L}, \mathbf{R}\rangle) = \langle \mathbf{S}, \mathbf{T}\rangle$. Then

$$\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle} = \sum_{\mathbf{A},\mathbf{B}\in I} \mathcal{P}[(c_1(\mathbf{L}) \oplus \mathbf{R} = \mathbf{A}) \wedge (c_2(\mathbf{R}) \oplus \mathbf{A} = \mathbf{B}) \wedge (c_3(\mathbf{A}) \oplus \mathbf{B} = \mathbf{S}) \wedge (c_4(\mathbf{B}) = \mathbf{T})]. \tag{23}$$

We consider the following conditions $(\mathcal{C})$ on $(\mathbf{A}, \mathbf{B})$:

- $(\mathcal{C}1)$ $\mathbf{A} \oplus \mathbf{R} \sim \mathbf{L}$ and $\nexists i, j$ s.t. $L_i = A_j \oplus R_j$.
- $(\mathcal{C}2)$ $\mathbf{A} \oplus \mathbf{B} \sim \mathbf{R}$ and $\nexists i, j$ s.t. $R_i = A_j \oplus B_j$.
- $(\mathcal{C}3)$ $\mathbf{B} \oplus \mathbf{S} \in I^{\neq}$ and $\nexists i, j$ s.t. $A_i = B_j \oplus S_j$.
- $(\mathcal{C}4)$ $\nexists i, j$ s.t. $B_i = T_j$.

Then equation (23) implies

$$\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle} \geq \sum_{\substack{\mathbf{A},\mathbf{B}\in I^{\neq} \\ (\mathbf{A},\mathbf{B}) \text{ satisfies } (\mathcal{C})}} \mathcal{P}[(c_1(\mathbf{L}) \oplus \mathbf{R} = \mathbf{A})] \cdot \mathcal{P}[c_2(\mathbf{R}) \oplus \mathbf{A} = \mathbf{B}] \cdot \mathcal{P}[c_3(\mathbf{A}) \oplus \mathbf{B} = \mathbf{S}] \cdot \mathcal{P}[c_4(\mathbf{B}) = \mathbf{T}]. \tag{24}$$

The number of $\mathbf{A}$ such that $(\mathcal{C}1)$ is satisfied is $\frac{(2^n - m + \lambda)!}{(2^n - 2m + 2\lambda)!}$. For a (perfect) random such $\mathbf{A}$ we have

$$\mathcal{P}[\mathbf{A} \in I^{\neq} | (\mathcal{C}1)] = 1 - \mathcal{P}[\bigvee_{i<j} A_i = A_j | (\mathcal{C}1)] \geq 1 - \sum_{i<j} \mathcal{P}[A_i = A_j | (\mathcal{C}1)]. \tag{25}$$

Consider given $1 \leq i < j \leq m$, and assume $L_i \neq L_j$ and $R_i \neq R_j$. As there are $(2^n - m + \lambda)(2^n - m + \lambda - 1)$ possible values for (A_i, A_j) satisfying $(\mathcal{C}_1)$, among which $2^n - m + \lambda$ satisfy $A_i = A_j$, we get

$$\mathcal{P}[A_i = A_j | (\mathcal{C}_1)] = \frac{2^n - m + \lambda}{(2^n - m + \lambda)(2^n - m + \lambda - 1)} \leq \frac{2}{2^n}. \tag{26}$$

If $L_i = L_j$ or $R_i = R_j$, it is easy to see that $\mathcal{P}[A_i = A_j | (\mathcal{C}1)] = 0$.

Then we have

$$\mathcal{P}[\mathbf{A} \in I^{\neq} | (\mathcal{C}1)] \geq 1 - \frac{m(m-1)}{2} \cdot \frac{2}{2^n} \geq 1 - \frac{m^2}{2^n}. \tag{27}$$

Similarly, the number of $\mathbf{B}$ such that $(\mathcal{C}2)$ is satisfied is $\frac{(2^n-m+\rho)!}{(2^n-2m+2\rho)!}$, and $\mathcal{P}[\mathbf{B} \in I^{\neq}|(\mathcal{C}2)] \geq 1 - \frac{m^2}{2^n}$. Finally for a (perfect) random $(\mathbf{A}, \mathbf{B})$ we compute:

$$\begin{aligned}
&\mathcal{P}[\mathbf{B} \text{ satisfies } (\mathcal{C}3) \wedge \mathbf{B} \text{ satisfies } (\mathcal{C}4) \wedge \mathbf{A} \in I^{\neq} \wedge \mathbf{B} \in I^{\neq}|(\mathcal{C}1) \wedge (\mathcal{C}2)] \\
&\geq 1 - \mathcal{P}[\bigvee_{i<j} B_i \oplus S_i = B_j \oplus S_j|(\mathcal{C}1) \wedge (\mathcal{C}2)] \\
&- \mathcal{P}[\bigvee_{i,j} A_i = B_j \oplus S_j|(\mathcal{C}1) \wedge (\mathcal{C}2)] - \mathcal{P}[\bigvee_{i,j} B_i = T_j|(\mathcal{C}1) \wedge (\mathcal{C}2)] - 2 \cdot \frac{m^2}{2^n} \\
&\geq 1 - \frac{m(m-1)}{2} \cdot \frac{2}{2^n} - \frac{4m^2}{2^n} - 2 \cdot \frac{m^2}{2^n} \geq 1 - \frac{7m^2}{2^n}.
\end{aligned}$$

Thus the number of $(\mathbf{A}, \mathbf{B}) \in I^{\neq}$ satisfying $(\mathcal{C})$ can be lower bounded by

$$\frac{(2^n - m + \lambda)!}{(2^n - 2m + 2\lambda)!} \cdot \frac{(2^n - m + \rho)!}{(2^n - 2m + 2\rho)!} \cdot (1 - \frac{7m^2}{2^n}). \tag{28}$$

Under these conditions on $(\mathbf{A}, \mathbf{B})$ we can evaluate

$$\mathcal{P}[(c_1(\mathbf{L}) \oplus \mathbf{R} = \mathbf{A})] \cdot \mathcal{P}[c_2(\mathbf{R}) \oplus \mathbf{A} = \mathbf{B}] \cdot \mathcal{P}[c_3(\mathbf{A}) \oplus \mathbf{B} = \mathbf{S}] \cdot \mathcal{P}[c_4(\mathbf{B}) = \mathbf{T}]$$

and we obtain

$$\frac{\frac{(2^n-2m+2\lambda)!}{2^{2^{n-1}-m+\lambda} \cdot (2^{n-1}-m+\lambda)!} \cdot \frac{(2^n-2m+2\rho)!}{2^{2^{n-1}-m+\rho} \cdot (2^{n-1}-m+\rho)!} \cdot \left[\frac{(2^n-2m)!}{2^{2^{n-1}-m} \cdot (2^{n-1}-m)!}\right]^2}{\left[\frac{2^n!}{2^{2^{n-1}} \cdot (2^{n-1})!}\right]^4}. \tag{29}$$

After multiplication of (29) by the number of terms (28):

$$\begin{aligned}
&\frac{2^{4m-\lambda-\rho} \cdot (2^n - m + \lambda)! \cdot (2^n - m + \rho)!}{(2^{n-1} - m + \lambda)! \cdot (2^{n-1} - m + \rho)!} \cdot \frac{(2^{n-1})!^4}{(2^n)!^4} \cdot \left[\frac{(2^n - 2m)!}{(2^{n-1} - m)!}\right]^2 \cdot \left(1 - \frac{7m^2}{2^n}\right) \\
&= 2^{4m-\lambda-\rho} \cdot \frac{\prod_{i=0}^{m-\lambda-1} \frac{2^{n-1}-i}{2^n-i} \cdot \prod_{i=0}^{m-\rho-1} \frac{2^{n-1}-i}{2^n-i} \cdot \left(\prod_{i=0}^{m-1} \frac{2^{n-1}-i}{2^n-i}\right)^2}{\left(\prod_{i=m}^{2m-1} 2^n - i\right)^2} \cdot \left(1 - \frac{7m^2}{2^n}\right).
\end{aligned}$$

By lower bounding the products, this expression can be shown to be greater or equal than

$$2^{4m-\lambda-\rho} \cdot \left(\frac{2^{n-1} - m}{2^n - m}\right)^{4m-\lambda-\rho} \cdot \frac{1}{2^{2nm}} \cdot \left(1 - \frac{7m^2}{2^n}\right). \tag{30}$$

It is easy to show that $\frac{2^{n-1}-m}{2^n-m} = \frac{1}{2} - \frac{1}{2} \sum_{k=1}^{\infty} \frac{m^k}{2^{nk}}$. Then (30) is greater or equal than

$$\left(1 - \sum_{k=1}^{\infty} \frac{m^k}{2^{nk}}\right)^{4m} \cdot \frac{1}{2^{2nm}} \cdot \left(1 - \frac{7m^2}{2^n}\right). \tag{31}$$

We evaluate the first factor. First, we have

$$\sum_{k=1}^{\infty} \frac{m^k}{2^{nk}} = \frac{m/2^n}{1 - m/2^n}. \tag{32}$$

Without loss of generality, we may assume $m/2^n \leq m^2/2^n \leq 1/7$. So we have

$$\left(1 - \sum_{k=1}^{\infty} \frac{m^k}{2^{nk}}\right)^{4m} \geq \left(1 - \frac{7m}{6 \cdot 2^n}\right)^{4m}. \tag{33}$$

Applying lemma 2, we obtain

$$\left(1 - \frac{7m}{6 \cdot 2^n}\right)^{4m} > 1 - 4m \left(\frac{\frac{7m}{6 \cdot 2^n}}{1 - \frac{7m}{6 \cdot 2^n}}\right) \geq 1 - 4m \left(\frac{\frac{7m}{6 \cdot 2^n}}{1 - \frac{7}{6} \cdot \frac{1}{7}}\right) = 1 - \frac{28m^2}{5 \cdot 2^n}. \tag{34}$$

Finally, immediate calculations show that (31) is greater or equal than

$$\left(1 - \frac{63m^2}{5 \cdot 2^n}\right) \cdot \frac{1}{2^{2nm}}, \tag{35}$$

which concludes the proof.

II.5.8. Pseudorandomness of 3R R-Scheme with Involutions

We prove a result similar to theorem 18 for a 3-round R-scheme where the three round functions c_1, c_2, c_3 are involutions without fixed points (see Figure 5). In the next section the following lemma will be proved:

LEMMA 3. *Let $m, n > 0$. Let $\langle \mathbf{L}, \mathbf{R} \rangle \in \mathcal{X} \subset I_{2n}^m, \langle \mathbf{S}, \mathbf{T} \rangle \in I^{\neq} \times I^{\neq}$. Then the probability for a 3-uple (c_1, c_2, c_3) of involutions without fixed points to satisfy $\psi'_R(c_1, ..., c_3)(\langle \mathbf{L}, \mathbf{R} \rangle) = \langle \mathbf{S}, \mathbf{T} \rangle$ is lower bounded by*

$$\left(1 - \frac{9m^2}{2^n}\right) \cdot \frac{1}{2^{2nm}}.$$

It implies the following theorem:

THEOREM 19. *Let c_1^*, c_2^*, c_3^* be independent perfect random involutions without fixed points on I_n. Let $C := \psi_R(c_1^*, c_2^*, c_3^*)$. Let $F^* \in \mathbf{F}_{2n,2n}$ be a perfect random function. Then for any pseudorandom distinguisher $\mathcal{A}$ allowed to make m queries, we have*

$$Adv_{\mathcal{A}}(C, F^*) < \frac{10m^2}{2^n}.$$

Thus $\psi_R(c_1^*, c_2^*, c_3^*)$ is pseudorandom, and secure as long as $m \ll 2^{n/2}$.

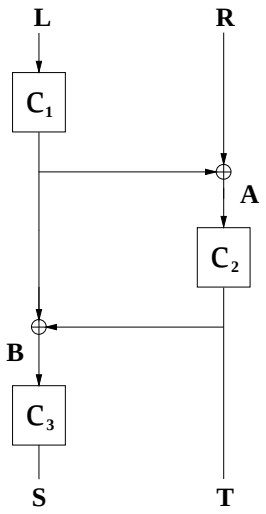


FIGURE 5. The 3-round R-scheme.

PROOF. We apply theorem 11. As $\langle \mathbf{S}, \mathbf{T} \rangle$ is assumed to belong to $I^{\neq} \times I^{\neq}$, ϵ_1 is equal to the probability for a (perfect) random $\langle \mathbf{S}, \mathbf{T} \rangle \in I \times I$ that $\langle \mathbf{S}, \mathbf{T} \rangle \notin I^{\neq} \times I^{\neq}$. We obtain

$$\epsilon_1 = \mathcal{P}[\mathbf{S} \in I^{=} \vee \mathbf{T} \in I^{=}] \leq \mathcal{P}[\bigvee_{i<j} S_i = S_j] + \mathcal{P}[\bigvee_{i<j} T_i = T_j]$$

$$\leq \sum_{i<j} \mathcal{P}[S_i = S_j] + \sum_{i<j} \mathcal{P}[T_i = T_j] \leq \frac{m^2}{2^n}.$$

Lemma 3 gives the corresponding ϵ_2. □

The proof of lemma 3 will require the following lemma:

LEMMA 4. *Let $x, y \in I_n, 0 \neq \Delta \in I_n$. The probability for a random involution without fixed points c to satisfy*

$$c(x) \oplus c(y) = \Delta$$

is at most $4/2^n$.

PROOF. If $x = y$ the probability is trivially 0. Thus in the following we assume $x \neq y$.

$$\mathcal{P}[c(x) \oplus c(y) = \Delta] = \sum_{a \in I_n} \mathcal{P}[c(x) = a \wedge c(y) = a \oplus \Delta]$$

$$= \sum_{\substack{a \in I_n \\ a \notin \{x, y, x \oplus \Delta, y \oplus \Delta\}}} \mathcal{P}[c(x) = a \wedge c(y) = a \oplus \Delta] + p,$$

where $p = \mathcal{P}[c(x) = y] = \frac{1}{2^n - 1}$ if $\Delta = x \oplus y$, else $p = 0$.

Under the assumption that $a \notin \{x, y, x \oplus \Delta, y \oplus \Delta\}$, we use properties 3 and 4 of section II.5.5 to compute:

$$\begin{aligned}\mathcal{P}[c(x) = a \wedge c(y) = a \oplus \Delta] &= 4 \cdot \frac{2^{n-1}!}{(2^{n-1}-2)!} \cdot \frac{(2^n-4)!}{2^n!} \\ &= \frac{2^n - 2}{(2^n-1)(2^n-2)(2^n-3)} \\ &\leq \frac{1}{2^{2n-1}},\end{aligned}$$

provided that $n \geq 4$.

Thus finally $\mathcal{P}[c(x) \oplus c(y) = \Delta] \leq (2^n - 4) \cdot \frac{2}{2^{2n}} + \frac{1}{2^n - 1} \leq \frac{4}{2^n}$. □

II.5.9. Proof of Lemma 3

We define two intermediate states during the computation of $\psi'_R(c_1, c_2, c_3)$, namely (see Figure 5):

$$\begin{aligned}\mathbf{A} &:= c_1(\mathbf{L}) \oplus \mathbf{R} \\ \mathbf{B} &:= c_1(\mathbf{L}) \oplus \mathbf{T}\end{aligned}$$

Let $\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle}$ be the probability that a random 3-uple (c_1, c_2, c_3) is such that $\psi'_R(c_1, c_2, c_3)(\langle \mathbf{L}, \mathbf{R}\rangle) = \langle \mathbf{S}, \mathbf{T}\rangle$. Then

$$\begin{aligned}\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle} = \sum_{\mathbf{A},\mathbf{B} \in I} \mathcal{P}[&(c_1(\mathbf{L}) \oplus \mathbf{R} = \mathbf{A}) \wedge (c_1(\mathbf{L}) \oplus \mathbf{T} = \mathbf{B}) \\ &\wedge (c_2(\mathbf{A}) = \mathbf{T}) \wedge (c_3(\mathbf{B}) = \mathbf{S})].\end{aligned} \tag{36}$$

We consider the following conditions $(\mathcal{C})$ on $(\mathbf{A}, \mathbf{B})$:

- $(\mathcal{C}1)$ $\nexists i, j$ s.t. $L_i = A_j \oplus R_j$.
- $(\mathcal{C}2)$ $\nexists i, j$ s.t. $A_i = T_j$.
- $(\mathcal{C}3)$ $\nexists i, j$ s.t. $B_i = S_j$.

(36) is greater or equal than

$$\begin{aligned}\sum_{\substack{\mathbf{A},\mathbf{B} \in I^{\neq} \\ (\mathbf{A},\mathbf{B}) \text{ satisfies } (\mathcal{C})}} &\mathcal{P}[(c_1(\mathbf{L}) \oplus \mathbf{R} = \mathbf{A}) \wedge (c_1(\mathbf{L}) \oplus \mathbf{T} = \mathbf{B})] \\ &\cdot \mathcal{P}[c_2(\mathbf{A}) = \mathbf{T}] \cdot \mathcal{P}[c_3(\mathbf{B}) = \mathbf{S}].\end{aligned} \tag{37}$$

When $\mathbf{A}, \mathbf{B} \in I^{\neq}$ and $(\mathbf{A}, \mathbf{B})$ satisfies $(\mathcal{C})$, we have

$$\mathcal{P}[c_2(\mathbf{A}) = \mathbf{T}] \cdot \mathcal{P}[c_3(\mathbf{B}) = \mathbf{S}] = \left[\frac{\frac{(2^n - 2m)!}{2^{2^{n-1}-m} \cdot (2^{n-1}-m)!}}{\frac{2^n!}{2^{2^{n-1}} \cdot (2^{n-1})!}} \right]^2. \tag{38}$$

It is easy to show that (38) is greater or equal than

$$2^{2m} \cdot \left(\frac{2^{n-1}-m}{2^n-m}\right)^{2m} \cdot \frac{1}{2^{2nm}}. \tag{39}$$

Thus $\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle}$ is greater or equal than

$$2^{2m}\cdot\left(\frac{2^{n-1}-m}{2^n-m}\right)^{2m}\cdot\frac{1}{2^{2nm}}\cdot \sum_{\substack{\mathbf{A},\mathbf{B}\in I^{\neq} \\ (\mathbf{A},\mathbf{B}) \text{ satisfies } (\mathcal{C})}} \mathcal{P}[(c_1(\mathbf{L})\oplus\mathbf{R}=\mathbf{A})\wedge(c_1(\mathbf{L})\oplus\mathbf{T}=\mathbf{B})], \tag{40}$$

where the sum is equal to

$$\begin{aligned}
&\mathcal{P}[c_1(\mathbf{L})\oplus\mathbf{R}\in I^{\neq} \wedge c_1(\mathbf{L})\oplus\mathbf{T}\in I^{\neq} \wedge \nexists i,j : L_i = c_1(L_j) \\
&\qquad\qquad \wedge \nexists i,j : T_i = c_2(T_j) \wedge \nexists i,j : S_i = c_3(S_j)] \\
&\geq 1 - \mathcal{P}[c_1(\mathbf{L})\oplus\mathbf{R}\in I^{=}] - \mathcal{P}[c_1(\mathbf{L})\oplus\mathbf{T}\in I^{=}] - \mathcal{P}[\exists i,j : L_i = c_1(L_j)] \\
&\qquad\qquad - \mathcal{P}[\exists i,j : T_i = c_2(T_j)] - \mathcal{P}[\exists i,j : S_i = c_3(S_j)] \\
&\geq 1 - \sum_{i<j}\mathcal{P}[c_1(L_i)\oplus R_i = c_1(L_j)\oplus R_j] - \sum_{i<j}\mathcal{P}[c_1(L_i)\oplus T_i = c_1(L_j)\oplus T_j] \\
&\quad - \sum_{i<j}\mathcal{P}[L_i = c_1(L_j)] - \sum_{i<j}\mathcal{P}[T_i = c_2(T_j)] - \sum_{i<j}\mathcal{P}[S_i = c_3(S_j)]
\end{aligned}$$

Using lemma 4 to bound the first two sums (note that if $R_i = R_j$ then by hypothesis $L_i \neq L_j$ which implies $\mathcal{P}[c_1(L_i)\oplus R_i = c_1(L_j)\oplus R_j] = 0$):

$$\begin{aligned}
&\geq 1 - m(m-1)\frac{4}{2^n} - 3\cdot\frac{m(m-1)}{2}\cdot\frac{1}{2^n-1} \\
&\geq 1 - \frac{4m^2}{2^n} - \frac{2m^2}{2^n} = 1 - \frac{6m^2}{2^n}.
\end{aligned}$$

Then

$$\begin{aligned}
\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle} &\geq 2^{2m}\cdot\left(\frac{2^{n-1}-m}{2^n-m}\right)^{2m}\cdot\frac{1}{2^{2nm}}\cdot\left(1-\frac{6m^2}{2^n}\right) \\
&= \left(1-\sum_{k=1}^{\infty}\frac{m^k}{2^{nk}}\right)^{2m}\cdot\frac{1}{2^{2nm}}\cdot\left(1-\frac{6m^2}{2^n}\right).
\end{aligned} \tag{41}$$

Using lemma 2 and assuming without loss of generality $m/2^n \leq 1/6$, we make a development similar to the one of section II.5.7. We get

$$\left(1-\sum_{k=1}^{\infty}\frac{m^k}{2^{nk}}\right)^{2m} > 1 - \frac{3m^2}{2^n}, \tag{42}$$

which implies

$$\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle} > \left(1-\frac{9m^2}{2^n}\right)\cdot\frac{1}{2^{2nm}}. \tag{43}$$

II.5.10. Superpseudorandomness of 5-round Schemes with Involutions

The following lemma is proved in the next section:

LEMMA 5. *Let $m, n > 0$. Let $\langle \mathbf{L}, \mathbf{R} \rangle, \langle \mathbf{S}, \mathbf{T} \rangle \in \mathcal{X} \subset I_{2n}^m$. Then the probability for a 5-uple $(c_1, c_2, c_3, c_4, c_5)$ of involutions without fixed points to satisfy $\psi'_L(c_1, ..., c_5)(\langle \mathbf{L}, \mathbf{R} \rangle) = \langle \mathbf{S}, \mathbf{T} \rangle$ is lower bounded by*

$$\left(1 - \frac{12m^2}{2^n}\right) \cdot \frac{1}{2nm}.$$

Using theorem 12, it implies superpseudorandomness for a 5-round scheme with the inner functions being perfect random involutions without fixed points:

THEOREM 20. *Let $c_1^*, c_2^*, ..., c_5^*$ be independent perfect random involutions without fixed points of I_n. Let $C := \psi_L(c_1^*, c_2^*, ..., c_5^*)$ (resp. $C := \psi_R(c_1^*, c_2^*, ..., c_5^*)$). Let C^* be a perfect random permutation of I_{2n}. Then for any superpseudorandom distinguisher $\mathcal{A}$ allowed to make m queries,*

$$Adv_{\mathcal{A}}(C, C^*) < \frac{12m^2}{2^n} + \frac{m^2}{2 \cdot 2^{2n}}.$$

II.5.11. Proof of Lemma 5

We use the following intermediate states (see Figure 6):

$$\begin{aligned} \mathbf{A} &:= c_1(\mathbf{L}) \oplus \mathbf{R} \\ \mathbf{B} &:= c_2(\mathbf{R}) \oplus \mathbf{A} \\ \mathbf{C} &:= c_3(\mathbf{A}) \oplus \mathbf{B} \end{aligned}$$

Let $\mathcal{P}^{\langle \mathbf{L}, \mathbf{R} \rangle}_{\langle \mathbf{S}, \mathbf{T} \rangle}$ be the probability that a random 5-uple $(c_1, c_2, c_3, c_4, c_5)$ of involutions without fixed points is such that $\psi'_L(c_1, c_2, c_3, c_4, c_5)(\langle \mathbf{L}, \mathbf{R} \rangle) = \langle \mathbf{S}, \mathbf{T} \rangle$. Then

$$\begin{aligned} \mathcal{P}^{\langle \mathbf{L}, \mathbf{R} \rangle}_{\langle \mathbf{S}, \mathbf{T} \rangle} = \sum_{\mathbf{A}, \mathbf{B}, \mathbf{C} \in I} \mathcal{P}[&(c_1(\mathbf{L}) \oplus \mathbf{R} = \mathbf{A}) \wedge (c_2(\mathbf{R}) \oplus \mathbf{A} = \mathbf{B}) \\ &\wedge (c_3(\mathbf{A}) \oplus \mathbf{B} = \mathbf{C}) \wedge (c_4(\mathbf{B}) \oplus \mathbf{C} = \mathbf{T}) \wedge (c_5(\mathbf{C}) = \mathbf{S})]. \end{aligned} \tag{44}$$

We define the following three conditions $(\mathcal{C})$ on $(\mathbf{A}, \mathbf{B}, \mathbf{C})$:

- $(\mathcal{C}1)$ $\nexists i, j : L_i = A_j \oplus R_j$ and $\nexists i, j : R_i = A_j \oplus B_j$.
- $(\mathcal{C}2)$ $\nexists i, j : A_i = B_j \oplus C_j$ and $\nexists i, j : B_i = C_j \oplus T_j$.
- $(\mathcal{C}3)$ $\nexists i, j : C_i = S_j$.

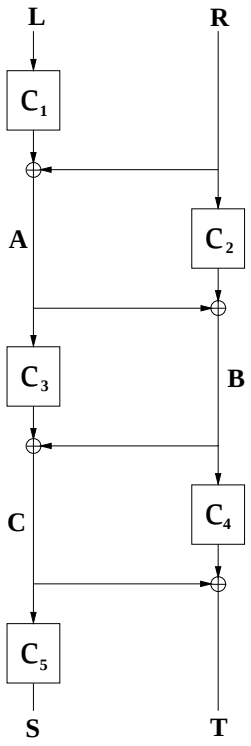


FIGURE 6. The 5-round L-scheme.

Then $\mathcal{P}^{\langle \mathbf{L},\mathbf{R}\rangle}_{\langle \mathbf{S},\mathbf{T}\rangle}$ is greater or equal than

$$\sum_{\substack{\mathbf{A},\mathbf{B}\in I^{\neq}\\ \mathbf{A},\mathbf{B} \text{ satisfy } (\mathcal{C}1)}} \Big(\mathcal{P}[(c_1(\mathbf{L})\oplus\mathbf{R}=\mathbf{A})\wedge(c_2(\mathbf{R})\oplus\mathbf{A}=\mathbf{B})]$$
$$\cdot \sum_{\substack{\mathbf{C}\in I\\ \mathbf{C} \text{ satisfies } (\mathcal{C}2),(\mathcal{C}3)}} \mathcal{P}[c_3(\mathbf{A})\oplus\mathbf{B}=\mathbf{C}]\cdot\mathcal{P}[c_4(\mathbf{B})\oplus\mathbf{C}=\mathbf{T}]\cdot\mathcal{P}[c_5(\mathbf{C})=\mathbf{S}]\Big). \tag{45}$$

We first evaluate the inner sum for given $\mathbf{A},\mathbf{B}\in I^{\neq}$ satisfying $(\mathcal{C}1)$. Adding constraints $\mathbf{C}\sim\mathbf{S}$, $\mathbf{C}\oplus\mathbf{T}\in I^{\neq}$ and $\mathbf{B}\oplus\mathbf{C}\in I^{\neq}$ only removes zero terms from the sum. Thus it is equal to

$$\sum_{\substack{\mathbf{C}\sim\mathbf{S}\\ \mathbf{C}\oplus\mathbf{T}\in I^{\neq},\ \mathbf{B}\oplus\mathbf{C}\in I^{\neq}\\ \mathbf{C} \text{ satisfies } (\mathcal{C}2),(\mathcal{C}3)}} \mathcal{P}[c_3(\mathbf{A})\oplus\mathbf{B}=\mathbf{C}]\cdot\mathcal{P}[c_4(\mathbf{B})\oplus\mathbf{C}=\mathbf{T}]\cdot\mathcal{P}[c_5(\mathbf{C})=\mathbf{S}]. \tag{46}$$

It is easy to see that

$$|\{\mathbf{C}\in I:\mathbf{C}\sim\mathbf{S}\wedge(\mathcal{C}_3)\}| = \frac{(2^n-m+\sigma)!}{(2^n-2m+2\sigma)!}. \tag{47}$$

Moreover we compute

$$\begin{aligned}
&\mathcal{P}[\mathbf{C} \oplus \mathbf{T} \in I^{\neq} \wedge \mathbf{B} \oplus \mathbf{C} \in I^{\neq} \wedge (\mathcal{C}_2) | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)] \\
&\geq 1 - \sum_{i<j} \mathcal{P}[C_i \oplus T_i = C_j \oplus T_j | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)] \\
&- \sum_{i<j} \mathcal{P}[B_i \oplus C_i = B_j \oplus C_j | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)]) \\
&- \sum_{i,j} \mathcal{P}[A_i = B_j \oplus C_j | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)] \\
&- \sum_{i,j} \mathcal{P}[B_i = C_j \oplus T_j | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)].
\end{aligned} \tag{48}$$

We evaluate the first sum. Consider given $1 \leq i < j \leq m$, and assume $S_i \neq S_j$ and $T_i \neq T_j$. Then the value of C_j is fixed by the one of C_i. As there are $(2^n - m + \sigma)(2^n - m + \sigma - 1)$ possible values of (C_i, C_j) satisfying $\mathbf{C} \sim \mathbf{S}$ and $(\mathcal{C}_3)$, we get

$$\mathcal{P}[C_i \oplus T_i = C_j \oplus T_j | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)] = \frac{1}{2^n - m + \sigma - 1} \leq \frac{2}{2^n}. \tag{49}$$

If $S_i = S_j$ or $T_i = T_j$, it is easy to see that the probability is 0, thus the inequality of equation (49) still holds. Therefore

$$\sum_{i<j} \mathcal{P}[C_i \oplus T_i = C_j \oplus T_j | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)] \leq \frac{m(m-1)}{2} \cdot \frac{2}{2^n} \leq \frac{m^2}{2^n}. \tag{50}$$

The second sum can be bounded similarly.

We now consider the third sum. Let $1 \leq i, j \leq m$. As there are $2^n - m + \sigma$ possible values of C_j satisfying $\mathbf{C} \sim \mathbf{S}$ and $(\mathcal{C}_3)$, we obtain

$$\mathcal{P}[C_j = A_i \oplus B_j | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)] = \frac{1}{2^n - m + \sigma} \leq \frac{2}{2^n}. \tag{51}$$

Therefore

$$\sum_{i,j} \mathcal{P}[A_i = B_j \oplus C_j | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)] \leq m^2 \cdot \frac{2}{2^n}. \tag{52}$$

The fourth sum can be bounded similarly.

Putting these inequalities together, we finally get

$$\mathcal{P}[\mathbf{C} \oplus \mathbf{T} \in I^{\neq} \wedge \mathbf{B} \oplus \mathbf{C} \in I^{\neq} \wedge (\mathcal{C}_2) | \mathbf{C} \sim \mathbf{S} \wedge (\mathcal{C}_3)] \geq 1 - \frac{6m^2}{2^n}. \tag{53}$$

The probabilities in (46) are easy to evaluate. Thus using (47) and (53), (46) is lower bounded by

$$\frac{\left[\frac{(2^n-2m)!}{2^{2^{n-1}-m} \cdot (2^{n-1}-m)!}\right]^2 \cdot \left[\frac{(2^n-m+\sigma)!}{2^{2^{n-1}-m+\sigma} \cdot (2^{n-1}-m+\sigma)!}\right]}{\left[\frac{2^n!}{2^{2^{n-1}} \cdot (2^{n-1})!}\right]^3} \cdot (1 - \frac{6m^2}{2^n}), \tag{54}$$

which is greater or equal than

$$2^{3m-\sigma} \cdot \left(\frac{2^{n-1}-m}{2^n-m}\right)^{3m-\sigma} \cdot \frac{1}{2nm} \geq \left(1 - \sum_{k=1}^{\infty} \frac{m^k}{2^{nk}}\right)^{3m} \cdot \frac{1}{2nm}. \quad (55)$$

It remains to evaluate

$$\sum_{\substack{\mathbf{A},\mathbf{B}\in I^{\neq} \\ \mathbf{A},\mathbf{B} \text{ satisfy } (\mathcal{C}1)}} \mathcal{P}[(c_1(\mathbf{L}) \oplus \mathbf{R} = \mathbf{A}) \wedge (c_2(\mathbf{R}) \oplus \mathbf{A} = \mathbf{B})], \quad (56)$$

which is equal to

$$\begin{aligned}
&\mathcal{P}[c_1(\mathbf{L}) \oplus \mathbf{R} \in I^{\neq} \wedge c_1(\mathbf{L}) \oplus c_2(\mathbf{R}) \oplus \mathbf{R} \in I^{\neq} \\
&\quad \wedge \nexists i,j : c_1(L_i) = L_j \wedge \nexists i,j : c_2(R_i) = R_j] \\
&\geq 1 - \sum_{i<j} \mathcal{P}[c_1(L_i) \oplus c_1(L_j) = R_i \oplus R_j] \\
&\quad - \sum_{i<j} \mathcal{P}[c_1(L_i) \oplus c_2(R_i) \oplus R_i = c_1(L_j) \oplus c_2(R_j) \oplus R_j] \\
&\quad - \sum_{i<j} \mathcal{P}[c_1(L_i) = L_j] - \sum_{i<j} \mathcal{P}[c_2(R_i) = R_j].
\end{aligned}$$

Let $1 \leq i < j \leq m$. $\mathcal{P}[c_1(L_i) \oplus c_1(L_j) = R_i \oplus R_j]$ is easy to evaluate. If $R_i \oplus R_j = 0$, then $L_i \neq L_j$ and the probability is 0. If $R_i \oplus R_j \neq 0$, we can apply lemma 4. Thus in any case

$$\mathcal{P}[c_1(L_i) \oplus c_1(L_j) = R_i \oplus R_j] \leq 4/2^n. \quad (57)$$

For shortness, let us denote $Z(R_i, R_j) := c_2(R_i) \oplus c_2(R_j) \oplus R_i \oplus R_j$. The terms of the second sum can be written:

$$\begin{aligned}
& \mathcal{P}[c_1(L_i) \oplus c_1(L_j) = Z(R_i, R_j)] \\
= \; & \mathcal{P}[c_1(L_i) \oplus c_1(L_j) = Z(R_i, R_j) | Z(R_i, R_j) = 0] \cdot \mathcal{P}[Z(R_i, R_j) = 0] \\
& + \mathcal{P}[c_1(L_i) \oplus c_1(L_j) = Z(R_i, R_j) | Z(R_i, R_j) \neq 0] \cdot \mathcal{P}[Z(R_i, R_j) \neq 0] \\
= \; & \mathcal{P}[c_1(L_i) \oplus c_1(L_j) = 0] \cdot \mathcal{P}[Z(R_i, R_j) = 0] \\
& + \mathcal{P}[c_1(L_i) \oplus c_1(L_j) = Z(R_i, R_j) | Z(R_i, R_j) \neq 0] \cdot \mathcal{P}[Z(R_i, R_j) \neq 0].
\end{aligned}$$

If $R_i = R_j$ then $L_i \neq L_j$ and the first term is 0. Else by lemma 4 it is not greater than $4/2^n$. Using lemma 4 again, the second term is also not greater than $4/2^n$. The conclusion is that

$$\mathcal{P}[c_1(L_i) \oplus c_2(R_i) \oplus R_i = c_1(L_j) \oplus c_2(R_j) \oplus R_j] \leq \frac{8}{2^n}. \quad (58)$$

Finally using (57) and (58), (56) is greater or equal than

$$\begin{aligned} & 1 - \frac{m(m-1)}{2} \cdot \frac{4}{2^n} - \frac{m(m-1)}{2} \cdot \frac{8}{2^n} - 2 \cdot \frac{m(m-1)}{2 \cdot (2^n - 1)} \\ \geq \quad & 1 - \frac{8m^2}{2^n}. \end{aligned} \tag{59}$$

Multiplying (55) and (59), we get

$$\mathcal{P}^{\langle \mathbf{L}, \mathbf{R} \rangle}_{\langle \mathbf{S}, \mathbf{T} \rangle} \geq \left(1 - \sum_{k=1}^{\infty} \frac{m^k}{2^{nk}}\right)^{3m} \cdot \frac{1}{2nm} \cdot \left(1 - \frac{8m^2}{2^n}\right). \tag{60}$$

Using lemma 2 again, (60) can be shown to be greater than

$$\left(1 - \frac{4m^2}{2^n}\right) \cdot \frac{1}{2nm} \cdot \left(1 - \frac{8m^2}{2^n}\right) > \left(1 - \frac{12m^2}{2^n}\right) \cdot \frac{1}{2nm}. \tag{61}$$

II.6. Conclusion

In this chapter we showed that replacing the inner permutations of a Misty structure by involutions without fixed points, without changing the number of rounds, does not significantly affect the previously known security bounds.

Several open problems remain: first, one could wonder whether the hypothesis "without fixed points" is important. Intuitively it is clearly not, as taking the inner permutations from a (much) bigger set increases the variety of functions one can generate, and hence the difficulty to distinguish them from perfect random functions. But it is only a conjecture. Moreover the number of involutions without fixed points is asymptotically negligible compared to the number of involutions, which makes us think that our proofs cannot be easily adapted to the case where involutions are used.

Also, it is an open question whether in some cases involutions achieve significantly weaker security bounds than permutations; in other words, it is not clear whether the fact that the inner functions are involutions can in some cases be used in an attack. This problem is also worth considering in structures different from the Misty one.

Basically, the limitations of the Luby-Rackoff model are twofold: the first limitation is due to the fact that it is difficult to do security proofs on real-life round functions. As we have seen, passing from permutations to involutions implied much trickier computations. But the second limitation is much more fundamental. Paradoxically, it owns to the fact that Luby-Rackoff proofs ensure security even against adversaries with unbounded computation capabilities. The consequence is that Luby-Rackoff security results cannot longer be proved as soon as the number of queries allowed to the adversaries permits an attack similar to exhaustive key search (i.e., to consider all possible r-uples of inner functions, and check whether they can generate the plaintext-ciphertext pairs observed).

CHAPTER III

Square Attacks

Abstract. This chapter deals with *square attacks*. First we briefly describe the original attack on SQUARE introduced by J. Daemen, L. Knudsen, and V. Rijmen [44]. We give a few definitions, and formalize the attack.
We then show the similarities between the way truncated differential distinguishers and square distinguishers are built. These similarities are used to derive square distinguishers on 16-round and 20-round Skipjack. We also present a square distinguisher on SAFER++, which allows attacking up to 4 rounds of this cipher.
Finally, we suggest possible extensions of square distinguishers. Among other things, we discuss the feasibility of applying square attacks to bit-oriented ciphers.
We originally published part of the results of this chapter in [139, 141].

III.1. Introduction

The square attack (also named saturation attack, or integral attack) was introduced by J. Daemen, L. Knudsen, and V. Rijmen when they published the algorithm of the same name [44]. It was presented as a dedicated attack. Square attacks were then applied to reduced-round Rijndael [47, 55, 61] and Crypton [107, 50], which is not surprising, as their structure is very close to the one of SQUARE. However four years later, S. Lucks applied the square technique to a non-square-like cipher, namely Twofish [110]. This way he managed to break up to eight rounds of Twofish, which is at present the best known attack against it. Since then, the square attack has been applied to several algorithms with various structures: examples are Hierocrypt [9], Camellia [171], Misty [99].

In this chapter we analyze the way square distinguishers are built; we show the similarities and differences such distinguishers have with truncated and impossible differential distinguishers. Section III.2 describes the original attack on SQUARE, and introduces the basic concepts of the square attack. These concepts are summarized in section III.3. Then section III.5 presents the principles of the truncated and impossible differential cryptanalysis. In section III.6 the link between both attacks

TABLE 1. The state during a SQUARE encryption.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

is discussed. This link is exploited in section III.7 to mount a square attack on Skipjack.

Another application of this type of attacks to SAFER++ is described in section III.9. Finally, possible extensions of the technique are considered in section III.10.

III.2. The Original Square Attack

The SQUARE algorithm was designed by J. Daemen, L. Knudsen and V. Rijmen in 1997 [44]. The state during a SQUARE encryption is usually described by a 4×4 square of bytes; these 16 bytes are denoted by $a_{i,j}$, $0 \leq i, j \leq 3$ (see Figure 1). SQUARE is a SP-Network made up of four different components:

- a mixing linear transform θ comparable to Rijndael's `MixColumns`. It is described by the following equation:

  $$b = \theta(a) \Leftrightarrow b_{i,j} = c_j a_{i,0} \oplus c_{j-1} a_{i,1} \oplus c_{j-2} a_{i,2} \oplus c_{j-3} a_{i,3},$$

  where the multiplication is in $\mathrm{GF}(2^8)$ and the indices of c must be taken modulo 4. Note that θ could be described equivalently using a modular polynomial multiplication.
- an S-boxes layer γ, in which each byte passes through a 8×8 S-box.
- a byte permutation π: $b = \pi(a) \Leftrightarrow b_{i,j} = a_{j,i}$.
- a round key addition $\sigma[k^t]$, which XORs key k^t to the state.

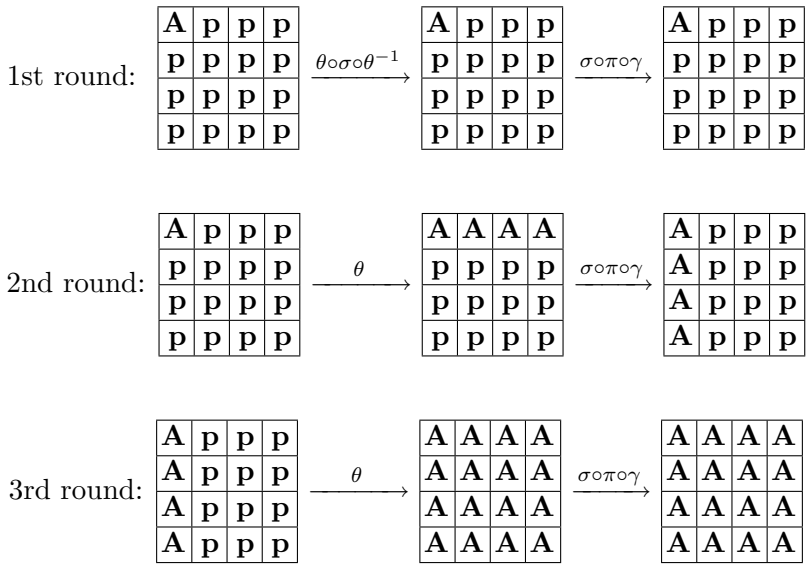
A round is defined as $\rho[k^t] = \sigma[k^t] \circ \pi \circ \gamma \circ \theta$. Then the whole cipher is

$$\mathrm{SQUARE}[K] = \bigcirc_{r=1}^{8} \rho[k^r] \circ \sigma[k^0] \circ \theta^{-1}.$$

We do not describe how $k_0, ..., k_8$ are computed from K, as it is of no relevance for the attack.

A chosen plaintext attack on SQUARE (reduced to up to 6 rounds) is the following. Consider a set Λ of 256 plaintexts $\{P^{(n)}\}_{n=1,...,256} = \{(P_{i,j}^{(n)})_{i,j=0,...,3}\}_{n=1,...,256}$ which are constant in all bytes but one; this

TABLE 2. Propagation of the active/passive character through 3 rounds of SQUARE.



particular byte takes all possible values. Let us assume its position is $(0,0)$. Thus more formally:

$$\begin{cases} p_{i,j}^{(n)} = p_{i,j} & \forall (i,j) \neq (0,0), \forall n \in \{1,2,...256\} \\ p_{0,0}^{(n)} = \Pi(n) & \forall n \in \{1,2,...256\}, \end{cases}$$

where Π is a permutation of $\{1,2,...256\}$.

Bytes which have the same value for all plaintexts are said *passive* or *constant*. Bytes for which all possible values appear the same number of times are said *active*. Bytes which are neither passive nor active are *garbled*.

For the set Λ of plaintexts, we trace the evolution of the active/passive state for the 16 bytes through the encryption process. γ and σ do not change the state of the bytes, and π only changes their position. The behavior of θ is easy to figure as well. The evolution is as pictured in Table 2, where **A** means active and **p** passive. After the θ layer of the fourth round all bytes are garbled. However one can show they still have an interesting property. Let $\{A^{(n)}\}_{n=1,...,256} = \{(a_{i,j}^{(n)})_{i,j=0,...,3}\}_{n=1,...,256}$ denote the state after round 3 corresponding to the set $\{P^{(n)}\}_{n=1,...,256}$ of plaintexts. Let $B^{(n)} = \theta(A^{(n)})$. Then

$$\bigoplus_{n=1}^{256} b_{i,j}^{(n)} = \bigoplus_{n=1}^{256} \bigoplus_{k=0}^{3} c_{j-k} a_{i,k}^{(n)} = \bigoplus_{l=0}^{3} c_l \bigoplus_{n=1}^{256} a_{i,j-l}^{(n)} = \bigoplus_{l=0}^{3} c_l \cdot 0 = 0.$$

In other words, the bytes at the output of θ in the fourth round have sum 0 when taken over the 256 plaintexts. A byte with such property is said *balanced*.

By doing guesses on key k^4 of the fourth round, this distinguisher can be used to retrieve the key of a 4-round SQUARE. More costly key guesses allow attacking up to 6 rounds of SQUARE using the same distinguisher.

III.3. Basic Definitions and Properties

In this section we collect basic definitions and properties related to the square attack.

First, we define a notion of set slightly different from the usual one. Namely, we consider that the elements of a set are always indexed. More formally:

DEFINITION 21. A ***n-bit set*** with e elements is defined as an injective function $\phi : \{1, 2, ..., e\} \to \mathbb{Z}_2^n$.

The notion of "multiset" is often used to describe an n-bit data channel. Roughly speaking, a multiset is a set whose elements may appear several times. Compared to the definition of set, it implies that ϕ is no longer required to be injective:

DEFINITION 22. A ***n-bit multiset*** with e elements is defined as a function $\phi : \{1, 2, ..., e\} \to \mathbb{Z}_2^n$.

In the remaining of this chapter a multiset with e elements will be denoted by $\{\!\{A^{(i)}\}\!\}_{i=1,...,e}$ (note the $\{\!\{.\}\!\}$ notation instead of $\{.\}$ for a set). We define the following properties for a n-bit multiset:

DEFINITION 23. A n-bit multiset with $k \cdot 2^n$ elements is said ***active*** if any value in $\mathbb{Z}_2^n$ appears exactly k times in it.
A n-bit multiset is said ***passive*** if it contains only one fixed value: $|\mathrm{Im}(\phi)| = 1$.
A n-bit multiset is said ***garbled*** if it is neither active nor passive.

We will sometimes allow us to write "active/passive word" to designate an active/passive multiset. The basic principle of a square attack is, beginning with a set of plaintexts of which the words are either active or passive, to trace active and passive words along the encryption process (the rules of propagation for active and passive words will be detailed below). More formally, we define the following terms:

DEFINITION 24. The ***state*** of a word is active, passive, or garbled.
The ***state*** of the data at a certain point of an encryption is the description of the state of every word that is relevant for the attack.
A ***square characteristic*** is the description of the state of the data after each round (or even after each block cipher layer).

It is worth noting that the size of the words considered need not be the one of one only S-box. It can include several ones, as we will see in the attacks on Skipjack and SAFER++. In this context, the notion of ***restriction*** of a multiset makes sense. We say that a m-bit multiset $\{\!\{B^{(i)}\}\!\}_{i=1,...,e}$ is a ***restriction*** of a n-bit multiset $\{\!\{A^{(i)}\}\!\}_{i=1,...,e}$ if $\{\!\{B^{(i)}\}\!\}_{i=1,...,e}$ is obtained by selecting some of the bits of $\{\!\{A^{(i)}\}\!\}_{i=1,...,e}$:

Definition 25. Consider a n-bit multiset

$$\phi : \{1, 2, ..., e\} \rightarrow \mathbb{Z}_2^n : k \rightarrow A^{(k)} = (a_1^{(k)}, a_2^{(k)}, ..., a_n^{(k)}).$$

A m-bit $(m < n)$ multiset

$$\phi : \{1, 2, ..., e\} \rightarrow \mathbb{Z}_2^m : k \rightarrow B^{(k)} = (b_1^{(k)}, b_2^{(k)}, ..., b_m^{(k)})$$

is a ***restriction*** of $\{\!\{A^{(k)}\}\!\}$ if there exists indices $1 \leq i_1 < i_2 < ... < i_m \leq n$ such that $\forall j \in \{1, 2, ..., m\}, \forall k \in \{1, 2, ..., e\} : b_j^{(k)} = a_{i_j}^{(k)}$.

Moreover, the ***complementary*** of a given m-bit restriction $\{\!\{B^{(k)}\}\!\}$ of a n-bit multiset $\{\!\{A^{(k)}\}\!\}$ is defined as the $(n-m)$-bit restriction of $\{\!\{A^{(k)}\}\!\}$ which consists of the bits that are not present in $\{\!\{B^{(k)}\}\!\}$. Restrictions have the following trivial property:

Property 5. All restrictions of an active multiset $\{\!\{A^{(k)}\}\!\}$ are active. All restrictions of a constant multiset $\{\!\{A^{(k)}\}\!\}$ are constant.

This property induces a partial order on the set of all active multisets of a given state. As a consequence, the state can be completely described by enumerating all maximal active multisets.

The following notion can be useful to attack some algorithms. An example is the attack on Twofish [110].

Definition 26. A multiset $\{\!\{A^{(i)}\}\!\}_{i=1,...,k \cdot 2^n}$ is said to be ***balanced*** with respect to some group operation whenever

$$\sum_{i=1}^{k \cdot 2^n} A^{(i)} = 0$$

(where $\sum$ and 0 refer to the considered group operation).

Balanced multisets have the following properties:

Theorem 27. *An active multiset is balanced. A passive multiset is balanced. The sum of two balanced multisets is a balanced multiset.*

III.4. Description of Skipjack

In this section we briefly describe the Skipjack algorithm, which will be dealt with in sections III.5 and III.7.

The algorithm Skipjack has been developed by the NSA and kept secret until 1998. It is a 64-bit iterated block cipher with a 80-bit key, of which the input is divided into four words of 16 bits. It is made out of 32 rounds of two types, called A-rounds and B-rounds; Skipjack applies 8 A-rounds, followed by 8 B-rounds, followed by another 8 A-rounds and finally another 8 B-rounds. In each round one of the four words passes through a bijective keyed transformation G, and at most two words are modified; furthermore a counter, which is incremented at each round, is XORed to one of the words. The G function is a four-round Feistel permutation of which the round function is defined as an 8×8 S-box preceded by addition of 8 key bits.

The key schedule of Skipjack is very simple. Namely, four bytes of the key at a time are used to key each G permutation: the first four bytes are used to key the first G permutation, and each additional G permutation is keyed by the next four bytes cyclically, with a cycle of five rounds.

Figure 1 pictures 16 rounds of Skipjack. The first eight are A-rounds, while the last eight are B-rounds.

III.5. Truncated and Impossible Differential Attacks

In this section, we briefly describe the truncated and impossible differential cryptanalysis. The results described come from [98, 16]. The attention of the reader is also drawn to [96, 67].

III.5.1. Truncated Differentials

In some ciphers, usually those of which all components operate on words of a given length (and never on bits), differential characteristics tend to cluster. This can be exploited by an attacker by considering only a part of the difference near the first and the last round. In practice, it amounts to tracing words having a zero difference through the cipher (however other characteristics are sometimes used, such as equality of the difference in different words).

As an example, it is easy to see that a truncated differential with probability 1 of Skipjack reduced to rounds 5 to 16 (thus including 4 A-rounds and 8 B-rounds) is $(0, a, 0, 0) \rightarrow (c, d, e, 0)$, where $a, c, d, e \neq 0$.

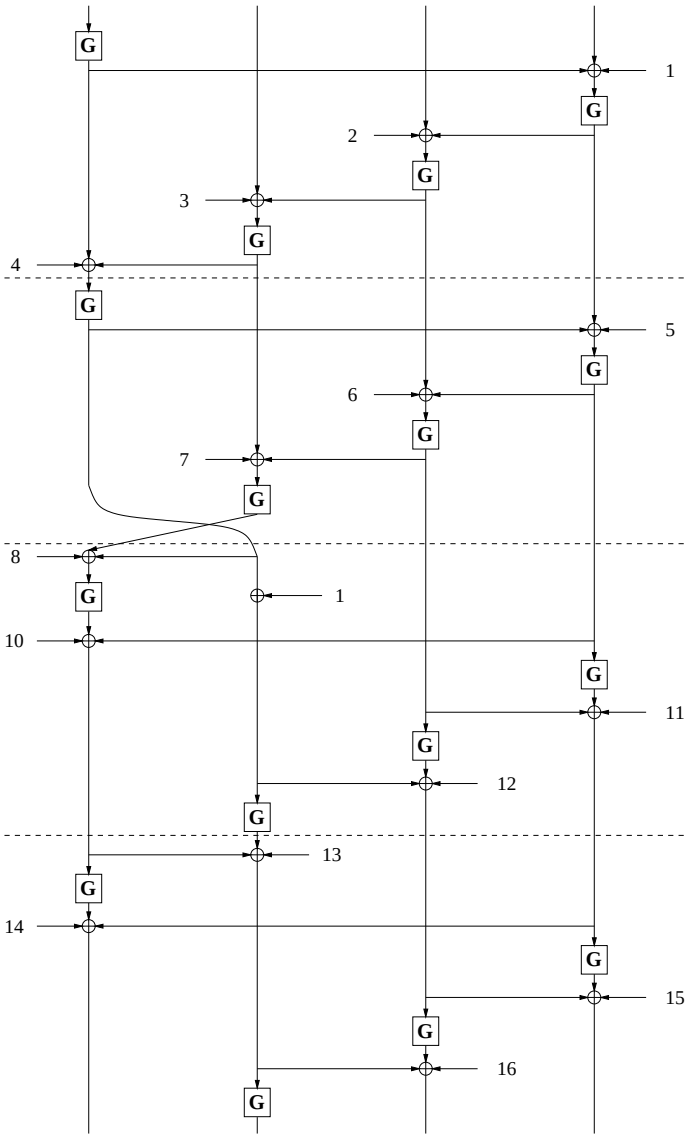


FIGURE 1. 16 rounds of Skipjack.

III.5.2. Impossible Differentials

Impossible differentials use truncated differentials with zero probability. The attacker does key guessing on the outer rounds of the cipher, and can discard guesses suggesting that the impossible differential indeed occurred. Impossible differential characteristics are usually built by concatenating two truncated differentials of probability 1 which are incompatible, which means that the conditions on the difference given

by both truncated differentials at their meeting point cannot be simultaneously fulfilled. It is why this type of attacks is also called *miss-in-the-middle* attack.

Still in the case of Skipjack, it is easy to show that if the difference after round 28 is of the form $(b, 0, 0, 0)$, with $b \neq 0$, then the difference after round 16 is of the form $(f, g, 0, h)$, with $f, g, h \neq 0$.

Consider both truncated differentials:

$$\begin{array}{ll} (0, a, 0, 0) \ {}_{5}\!\rightarrow_{16} (c, d, e, 0) & (a, c, d, e \neq 0) \\ (b, 0, 0, 0) \, {}_{28}\!\rightarrow_{17} (f, g, 0, h) & (b, f, g, h \neq 0) \end{array}$$

It is clear that differences $(c, d, e, 0)$ and $(f, g, 0, h)$ cannot both hold after round 16. We conclude that a differential $(0, a, 0, 0) \rightarrow (b, 0, 0, 0)$ between rounds 5 and 28 of Skipjack has probability 0. In [16] this impossible differential is used to attack up to 31 rounds of Skipjack.

Other examples of application of the impossible differential cryptanalysis are given in [17].

III.6. Square Versus Truncated Differential Characteristics

As we have seen, building a truncated differential characteristic is most of the time all about tracing the zero and non-zero differences. Square attacks, as for them, are about tracing active and passive words. We observe that the rules of propagation are very similar for square attacks and truncated differential attacks. As a matter of fact, it appears that active words behave the same way as non-zero differences, while passive words behave just like zero differences. We summarize these observations in Table 3, where:

- δ_0 (resp. $\Delta_{\neq 0}$) stands for a zero (resp. non-zero) difference.
- Act, Pas, Bal respectively denote an active, passive, and balanced multiset.
- $\oplus$ denotes the group operation of the cipher (usually exclusive or).
- F (resp. f) denotes a bijective (resp. non-bijective) function.

The interesting point with these observations is that it is often possible, given a truncated differential characteristic, to build a corresponding square characteristic. And reciprocally, given a square characteristic, one can often build a truncated differential characteristic. In the next section, we apply this technique to build a square distinguisher for Skipjack, based on the impossible differential distinguisher given in [16].

TABLE 3. Parallelism between square and truncated differential characteristics.

Square attacks	Truncated differential attacks
$Pas \oplus Pas \rightarrow Pas$	$\delta_0 \oplus \delta_0 \rightarrow \delta_0$
$Act \oplus Pas \rightarrow Act$	$\Delta_{\neq 0} \oplus \delta_0 \rightarrow \Delta_{\neq 0}$
$Act \oplus Act \rightarrow Bal$	$\Delta_{\neq 0} \oplus \Delta_{\neq 0} \rightarrow ?$
$Act \xrightarrow{F} Act$	$\Delta_{\neq 0} \xrightarrow{F} \Delta_{\neq 0}$
$Pas \xrightarrow{F} Pas$	$\delta_0 \xrightarrow{F} \delta_0$
$Act \xrightarrow{f} ?$	$\Delta_{\neq 0} \xrightarrow{f} ?$
$Pas \xrightarrow{f} Pas$	$\delta_0 \xrightarrow{f} \delta_0$

III.7. Square Attacks on Skipjack

In this section we first build a square distinguisher on Skipjack directly based on the truncated differential distinguisher described in section III.5.1. We then show how this distinguisher can be improved.

III.7.1. A 16-Round Square Distinguisher

The truncated differential distinguisher described in section III.5.1 is

$$(0, a, 0, 0) \ {}_5\!\rightarrow_{16} (c, d, e, 0).$$

Using observations made in section III.6 we "convert" this distinguisher into a square distinguisher for rounds 5-20 of Skipjack. Thus the initial state (before round 5) is (p, A, p, p). Table 4 shows the state of the different words after each round. We denote the four 16-bit words obtained after round r by the quadruplet $(a_0^r, a_1^r, a_2^r, a_3^r)$; an active (respectively passive, balanced, garbled) word is denoted by A (respectively p, b, .). Observe that the state after round 16 is (A, A, A, p), which was predictable, as it corresponds to the difference $(c, d, e, 0)$ after round 16 in the truncated differential characteristic.

Naturally we come to wonder whether we can still apply a miss-in-the-middle approach to a square-like characteristic. Indeed if the data after round 28 is in state (A, p, p, p), then the data after round 16 is in state (A, A, p, A) (we refer to the impossible differential characteristic of section III.5.2). Because data after round 16 cannot be simultaneously in state (A, A, A, p) and (A, A, p, A), we conclude that (p, A, p, p) before round 5 cannot cause (A, p, p, p) after round 28.

But the problem is the following: what is the probability, for a random permutation, that a batch of 2^{16} plaintexts with state (p, A, p, p) cause their ciphertexts to satisfy (A, p, p, p)? In fact, it is so small that we are almost sure that this event would never occur even if we consider

TABLE 4. 16-round square characteristic for Skipjack.

Round number r	a_0^r	a_1^r	a_2^r	a_3^r
4	p	A	p	p
5	p	A	p	p
6	p	A	p	p
7	p	A	p	p
8	A	p	p	p
9	A	p	p	p
10	A	p	p	p
11	A	p	p	p
12	A	p	p	p
13	A	A	p	p
14	A	A	p	p
15	A	A	p	p
16	A	A	A	p
17	A	A	A	A
18	A	A	b	A
19	A	.	.	A
20	.	.	.	A
21	.	.	.	.

all the 2^{48} possible batches! Building a distinguisher based on the miss-in-the-middle technique applied on square characteristics is therefore impossible.

III.7.2. A 20-Round Square Distinguisher

It has been shown in [74] that the distinguisher described in the previous section can be extended. For this purpose we consider not only the four 16-bit words $(a_0^r, a_1^r, a_2^r, a_3^r)$, but also 32-bit words made up of the concatenation of two of the earlier 16-bit words. There are 6 of them, which are denoted $b_{ij}^r := (a_i^r, a_j^r)$ with $0 \leq i < j \leq 3$. 48-bit words are considered as well; there are 4 of them, which are denoted $c_{ijk}^r := (a_i^r, a_j^r, a_k^r)$ with $0 \leq i < j < k \leq 3$.

Consider a batch of 2^{48} plaintexts, such that $\{\!\{c_{013}^0\}\!\}$ is active, and $\{\!\{a_2^0\}\!\}$ is passive. Then the state evolves through the first four rounds as described in Table 5.

Consider 2^{16} plaintexts such that a_0^4 and a_2^4 have fixed values, respectively $\alpha, \beta \in \mathbb{Z}_2^{16}$. As a_2^4 and a_3^4 are linked by a bijection (because a_2^0 is passive), it implies that for these 2^{16} plaintexts a_3^4 is constant as well. Then the 16-round distinguisher described in the previous section can be applied to these 2^{16} plaintexts. Summing up on all the 2^{32} batches

TABLE 5. First 4 rounds of a 20-round characteristic of Skipjack.

Round number r	a_0^r	a_1^r	a_2^r	a_3^r	c_{012}^r	c_{013}^r
Before 1	A	A	p	A	.	A
1	A	A	p	A	.	A
2	A	A	A	A	A	A
3	A	A	A	A	A	A
4	A	A	A	A	A	A

of 2^{16} plaintexts, we conclude that a_3^{20} is active, from which we obtain the 20-round distinguisher.

In [74] this distinguisher is used to attack up to 27 rounds of Skipjack.

III.8. Truncated Differential vs. Square Distinguishers: Conclusion

We have shown how strong is the link between square and truncated differential characteristics. Although the similarity we showed is factual rather than conceptual, we think it is interesting. Indeed, one conclusion of this discussion is that one should try to convert every truncated differential distinguisher into a square distinguisher, and vice-versa. A good illustration of this was given in section III.7.

However both techniques have advantages and weaknesses. One of the strong points for square attacks is that the sum of two active words still has a remarkable property, called *balance*. Another interesting specificity of its is property 5: any restriction of an active word is itself active. These two specificities of the square attack have no counterpart for truncated differential attacks, and can be useful in order to build the longest characteristic possible.

On the other hand, truncated differentials can be "missed-in-the-middle" in order to build impossible differential distinguishers. It virtually doubles the number of rounds attacked. We have seen that the miss-in-the-middle technique is not applicable to the square attack.

Finally, note that the square attack uses batches of many chosen plaintexts, and is not probabilistic (the distinguishing property is always satisfied by the ciphertexts). On the contrary, truncated differential distinguishers are sometimes probabilistic (although it is not the case in our example on Skipjack), but every single pair with a right plaintext difference is useful.

The conclusion is that both attacks can be useful, depending on the structure of the algorithm. In the case of Skipjack, the impossible differential attack wins the challenge. On the other hand, the best known attacks against the AES [55, 61] or Twofish [110] are fundamentally square attacks.

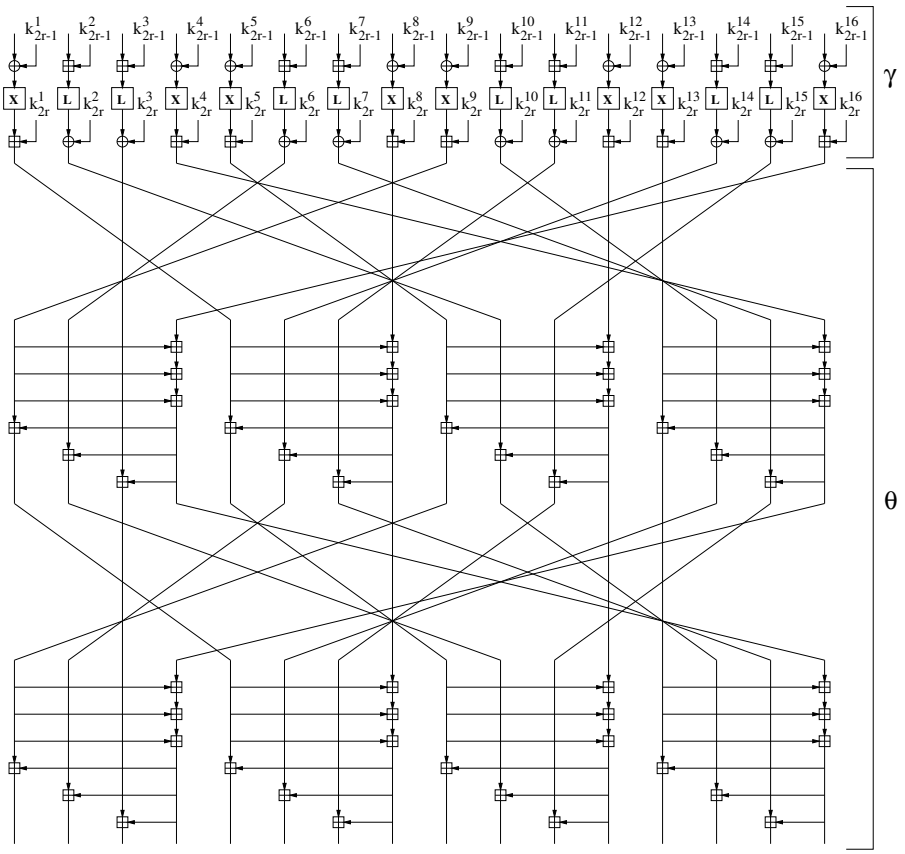


FIGURE 2. One round of SAFER++.

III.9. Square Attacks on SAFER++

In this section we present several square distinguishers on the SAFER++ algorithm. They allow attacking up to 4.5 rounds of SAFER++. We published the first known square distinguisher on SAFER++ in [141]. A few months later, other distinguishers were published by A. Biryukov et al. in [22].

III.9.1. Short Description of SAFER++

The SAFER++ algorithm was designed by J.L. Massey, G.H. Khachatrian, and M.K. Kuregian; it was a finalist of the NESSIE European project [126]. There are three different versions of SAFER++: 128-bit block 128-bit key (SAFER++$_{128}$); 128-bit block 256-bit key (SAFER++$_{256}$); and 64-bit block 128-bit key. We will deal with the first two versions only. They only differ by their key schedule and their number of rounds.

III.9.1.1. *The encryption algorithm:* The algorithm makes frequent use of addition mod 256, denoted by $\boxplus$ or $+$. It also uses exclusive or ($\oplus$). SAFER++$_{128}$ has 7 rounds, while SAFER++$_{256}$ consists of 10 rounds. One round is depicted in Figure 2. It is made out of four layers: successively, one key addition layer, one S-boxes layer (L and X are two 8×8 S-boxes, that are inverses of each other), another key addition layer, and finally the PHT diffusion layer. We denote the composition of the first three layers by γ, and the fourth layer by θ. We sometimes index them by the number r of the round: γ_r, θ_r. Note that both key addition layers alternate the $\boxplus$ and $\oplus$ operations. We denote by $F_{r,p}$ the action of γ restricted to the p^{th} byte at round r:

$$\begin{cases} F_{r,p}(x) = X(x \oplus k_{2r-1}^{(p)}) + k_{2r}^{(p)} & \text{if } p \equiv 0,1 \mod 4 \\ F_{r,p}(x) = L(x + k_{2r-1}^{(p)}) \oplus k_{2r}^{(p)} & \text{if } p \equiv 2,3 \mod 4 \end{cases} \tag{62}$$

where $k_i^{(p)}$ denotes the p^{th} byte of the i^{th} subkey. Note that the 7 (resp. 10) rounds are followed by one final key addition layer. Finally, the state during the encryption will sometimes be denoted as $(a_1, a_2, ..., a_{16})$.

III.9.1.2. *The key schedule of SAFER++$_{128}$:*
Let $K = (k^{(1)}, k^{(2)}, ..., k^{(16)})$ denote the key ($k^{(i)} \in \mathbb{Z}_2^8$). To understand our attack, it is sufficient to know the following properties of the key schedule (the reader interested in a detailed description can refer to [112]):

- A 17^{th} byte is computed as:

$$k^{(0)} = \bigoplus_{i=1}^{16} k^{(i)}. \tag{63}$$

- $k_1^{(p)} = k^{(p)} (p = 1, 2, ..., 16)$.
- $k_i^{(p)}$ can be computed as a function of $k^{(i+p-1 \mod 17)}$.

We use the notation $k_i^{(p)} \approx k_j^{(q)}$ to denote the fact that two subkey bytes are derived from the same master key byte.

III.9.1.3. *About the linear transform θ:* The linear transform can be described by the matrix multiplication $B = A \cdot M$, where $A, B \in \mathbb{Z}_{256}^{16}$ and the computations are done in $\mathbb{Z}_{256}$. The matrix M and its inverse M^{-1} are the following:

$$M = \begin{pmatrix}
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 2 & 2 & 2 & 1 & 1 & 2 & 1 \\
2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 4 & 2 & 2 \\
2 & 2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 \\
4 & 2 & 2 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & 2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 2 & 2 & 4 & 2 & 2 & 1 & 1 & 1 \\
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 \\
1 & 1 & 2 & 1 & 4 & 2 & 2 & 2 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 2 & 4 & 2 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 \\
1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 2 & 2 & 4 & 2 \\
2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 4 & 2 & 2 & 2 \\
2 & 4 & 2 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 \\
2 & 1 & 1 & 1 & 2 & 2 & 4 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}$$

$$M^{-1} = \begin{pmatrix}
0 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -4 & 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \\
0 & 0 & -1 & 16 & -1 & 0 & -4 & 1 & 0 & -4 & 0 & 1 & -4 & -1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & -1 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & -4 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & -1 & 0 & 1 & 1 & 0 \\
-4 & 0 & 0 & 1 & 0 & 0 & 0 & 16 & -1 & -1 & -4 & 1 & 0 & -4 & -1 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & -1 \\
0 & 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 1 & 0 \\
-1 & -4 & 0 & 1 & -4 & -1 & -1 & 1 & 0 & 0 & 0 & 16 & 0 & 0 & -4 & 1 \\
0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -4 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & -4 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -4 \\
0 & -1 & -4 & 1 & 0 & -4 & 0 & 1 & -4 & 0 & -1 & 1 & -1 & 0 & 0 & 16
\end{pmatrix}$$

Whereas the designers of SAFER++ claimed it to have a high diffusion PHT layer, it was proved in [124] that its differential byte branch number was ≤ 7. It is in fact precisely 5. As

$$a = \texttt{0x00008080000000000000000000000000}$$

implies

$$\theta(a) = \texttt{0x80800080000000000000000000000000}$$

and $W(a) = 2$ while $W(\theta(a)) = 3$, we conclude that $\mathcal{B}_d(\theta) \leq 5$.

By considering the different cases $(W(a), W(\theta(a))) = (1, 3)$ or $(2, 2)$ or $(3, 1)$, and showing that neither of them are possible, it is easy to show that the differential byte branch number of θ is indeed 5.

It is often a good idea to try integral attacks against ciphers with such a relatively weak diffusion layer. Indeed, such attacks can exploit weak diffusions, while the cryptographic strength of the S-boxes and key addition used are of no relevance against them.

III.9.2. A Distinguisher on 2 Rounds of SAFER++

Consider a set of plaintexts of the form

$$\{(A, B, a^{(i)}, D; E, F, G, H; I, J, K, L; b^{(i)}, N, O, P)\}_{i\in\{1,\dots,2^{16}\}}, \quad (64)$$

where $\{\!\{(a^{(i)}, b^{(i)})\}\!\}_{i\in\{1,\dots,2^{16}\}}$ is a 16-bit active multiset, the other bytes being constants.

As all functions $F_{r,p}$ are permutations, the action of γ does not change the active/passive properties. Thus the data after it can be described by (64) as well. Without loss of generality, we may assume the constants to be 0. Then at the end of first round, we obtain

$$\begin{aligned}&\{(2a^{(i)} + b^{(i)}, 2a^{(i)} + b^{(i)}, 4a^{(i)} + b^{(i)}, 2a^{(i)} + b^{(i)}; 2a^{(i)} + b^{(i)},\\ &a^{(i)} + 2b^{(i)}, a^{(i)} + b^{(i)}, a^{(i)} + b^{(i)}; a^{(i)} + b^{(i)}, 2a^{(i)} + b^{(i)}, a^{(i)} + 2b^{(i)},\\ &a^{(i)} + b^{(i)}; a^{(i)} + 4b^{(i)}, a^{(i)} + 2b^{(i)}, a^{(i)} + 2b^{(i)}, a^{(i)} + 2b^{(i)})\}_i.\end{aligned} \quad (65)$$

After the γ layer of the second round, the data can be written as

$$\begin{aligned}&\{X^{(i)}\}_i =\\ &\{F_{2,1}(2a^{(i)} + b^{(i)}), F_{2,2}(2a^{(i)} + b^{(i)}), F_{2,3}(4a^{(i)} + b^{(i)}), F_{2,4}(2a^{(i)} + b^{(i)});\\ &F_{2,5}(2a^{(i)} + b^{(i)}), F_{2,6}(a^{(i)} + 2b^{(i)}), F_{2,7}(a^{(i)} + b^{(i)}), F_{2,8}(a^{(i)} + b^{(i)});\\ &F_{2,9}(a^{(i)} + b^{(i)}), F_{2,10}(2a^{(i)} + b^{(i)}), F_{2,11}(a^{(i)} + 2b^{(i)}), F_{2,12}(a^{(i)} + b^{(i)});\\ &F_{2,13}(a^{(i)} + 4b^{(i)}), F_{2,14}(a^{(i)} + 2b^{(i)}), F_{2,15}(a^{(i)} + 2b^{(i)}), F_{2,16}(a^{(i)} + 2b^{(i)})\}_i.\end{aligned} \quad (66)$$

And after the second θ layer, the first byte of the output can be expressed as

$$\begin{aligned}&\{\!\{4F_{2,5}(2a^{(i)} + b^{(i)}) + 2F_{2,2}(2a^{(i)} + b^{(i)}) + 2F_{2,15}(a^{(i)} + 2b^{(i)}) +\\ &2F_{2,12}(a^{(i)} + b^{(i)}) + 2F_{2,14}(a^{(i)} + 2b^{(i)}) + F_{2,1}(2a^{(i)} + b^{(i)}) +\\ &F_{2,11}(a^{(i)} + 2b^{(i)}) + F_{2,8}(a^{(i)} + b^{(i)}) + 2F_{2,3}(4a^{(i)} + b^{(i)}) +\\ &F_{2,9}(a^{(i)} + b^{(i)}) + F_{2,6}(a^{(i)} + 2b^{(i)}) + F_{2,16}(a^{(i)} + 2b^{(i)}) + F_{2,13}(a^{(i)} + 4b^{(i)})\\ &+ F_{2,10}(2a^{(i)} + b^{(i)}) + F_{2,7}(a^{(i)} + b^{(i)}) + F_{2,4}(2a^{(i)} + b^{(i)})\}\!\}_i.\end{aligned} \quad (67)$$

We can summarize the successive states through these 2 rounds by:
(64) $\xrightarrow{\gamma_1}$ (64) $\xrightarrow{\theta_1}$ (65) $\xrightarrow{\gamma_2}$ (66) $\xrightarrow{\theta_2}$ (67).

As all bytes of (66) are active, they satisfy

$$\forall j \in \{1, 2, ..., 16\} : \sum_{i=1}^{2^{16}} X_j^{(i)} \equiv 0 \pmod{256}. \quad (68)$$

Thus so does sum (67); this is the distinguishing criteria. It is satisfied with probability 2^{-8} for a random permutation. As the same reasoning applies to any byte of the output, by checking whether all of them are

balanced we obtain a distinguisher that is satisfied with probability 2^{-128} for a random permutation. It is thus pretty strong.

Note finally that there are other distinguishers similar to the one we described, but corresponding to sets of 2^{16} plaintexts different from (64).

III.9.3. Three Attacks on SAFER++ Using this Distinguisher

III.9.3.1. *An attack on 3 rounds of SAFER++$_{128}$:* In this section we present an attack on 3 rounds of SAFER++, without the final key addition layer however. It means that the cipher we attack is $\gamma_1 \cdot \theta_1 \cdot \gamma_2 \cdot \theta_2 \cdot \gamma_3 \cdot \theta_3$, which is equivalent to $\gamma_1 \cdot \theta_1 \cdot \gamma_2 \cdot \theta_2 \cdot \gamma_3$. We make a straightforward use of the distinguisher we just described; it is applied to the first two rounds.

We consider 2^{16} chosen plaintexts following pattern (64). Then we guess the subkey bytes of k_5 and k_6 one after the other, and use the fact that the data we obtain by decrypting θ_3 and γ_3 must be balanced in order to eliminate a large proportion of bad guesses. More precisely, the attack works this way:

(1) Ask for encryption of 2^{16} plaintexts of the form (64).
(2)
- Go through all values of $(k_5^{(1)}, k_6^{(1)}) \approx (k^{(5)}, k^{(6)})$; for each of them compute the first byte of the data after round 2, corresponding to all 2^{16} ciphertexts. Check balance of these bytes. Out of the 2^{16} candidates, about 2^8 pass the test.
- Guess successively $k^{(7)}$, $k^{(8)}$, ..., $k^{(0)}$, $k^{(1)}$, ..., $k^{(5)}$. Note that $k^{(i)} \approx k_5^{(i-4)} \approx k_6^{(i-5)}$ $(i = 6...16)$ and $k^{(i)} \approx k_5^{(i+13)} \approx k_6^{(i+12)}$ $(i = 0...4)$. Thus each guess allows us to check whether one more byte is balanced. After checking balance, the mean number of candidates for the part of the key already guessed is 2^8. Note also that $k^{(4)} \approx k_6^{(16)}$ is obtained "for free" after all other key bytes were guessed, by equation (63). Thus after checking balance of the last byte we can expect to have a very small number of remaining candidates (about 1 wrong key along with the right one).

(3) If necessary, exhaustive search allows us to distinguish the right key from the few wrong ones.

The most time-consuming step of this algorithm is the first one. Thus global time complexity is about 2^{16} encryptions. Memory requirements are of the same order of magnitude. This attack is thus quite practical.

TABLE 6. Bytes that can be computed at the end of round 3 (attack on 4-round SAFER++$_{128}$). The key bytes in bold are those guessed at step 2 of the attack.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$k_7^{(i)} \approx$	$\mathbf{k^{(7)}}$	$\mathbf{k^{(8)}}$	$\mathbf{k^{(9)}}$	$\mathbf{k^{(10)}}$	$\mathbf{k^{(11)}}$	$k^{(12)}$	$\mathbf{k^{(13)}}$	$\mathbf{k^{(14)}}$	$k^{(15)}$	$\mathbf{k^{(16)}}$	$k^{(0)}$	$\mathbf{k^{(1)}}$	$k^{(2)}$	$\mathbf{k^{(3)}}$	$\mathbf{k^{(4)}}$	$\mathbf{k^{(5)}}$
$k_8^{(i)} \approx$	$\mathbf{k^{(8)}}$	$\mathbf{k^{(9)}}$	$\mathbf{k^{(10)}}$	$\mathbf{k^{(11)}}$	$k^{(12)}$	$\mathbf{k^{(13)}}$	$\mathbf{k^{(14)}}$	$k^{(15)}$	$\mathbf{k^{(16)}}$	$k^{(0)}$	$\mathbf{k^{(1)}}$	$k^{(2)}$	$\mathbf{k^{(3)}}$	$\mathbf{k^{(4)}}$	$\mathbf{k^{(5)}}$	$\mathbf{k^{(6)}}$
	√	√	√	√			√							√	√	√

III.9.3.2. *An attack on 4 rounds of SAFER++$_{128}$:* In this section we adapt our previous attack, if one more round is added at the beginning of the cipher. Thus neglecting the last linear transform, the cipher we attack is $\gamma_1 \cdot \theta_1 \cdot \gamma_2 \cdot \theta_2 \cdot \gamma_3 \cdot \theta_3 \cdot \gamma_4$. The distinguisher is applied to the rounds 2 and 3. Thus we must find a set of 2^{16} plaintexts such that the corresponding inputs to the second round look like (64). Applying the inverse linear layer to the 2^{16} data of pattern (64), and assuming constants to be 0, we obtain

$$\begin{aligned}&\{(0,0,a^{(i)}+b^{(i)},-4a^{(i)}-b^{(i)};0,b^{(i)},a^{(i)},0;\\&b^{(i)},a^{(i)},0,0;a^{(i)}+b^{(i)},0,0,-a^{(i)}-4b^{(i)})\}_i.\end{aligned} \tag{69}$$

As $k_1^{(i)} \approx k_2^{(i-1)}$ $(i = 2...16)$, it is easy to see that the corresponding set of plaintexts can be obtained by guessing on $13 \cdot 8 = 104$ key bits. More precisely, $k_1^{(2)} \approx k_2^{(1)}$, $k_1^{(12)} \approx k_2^{(11)}$, and $k_1^{(15)} \approx k_2^{(14)}$ do not need to be guessed (remember that the 0's in (69) could as well be replaced by any constant). Thanks to the guess made on 13 key bytes, it is possible to compute 8 bytes at the end of the third round. Table 6 illustrates this. As for a false key guess the test on each byte has a probability 2^{-8} to succeed, a proportion of about 2^{-64} of the bad guesses will not be discardable. Let us detail how the complete attack works:

(1) Ask for encryption of all 2^{64} plaintexts of the form

$$\{(A,B,\alpha_1^{(i)},\alpha_2^{(i)};C,\alpha_3^{(i)},\alpha_4^{(i)},D;\alpha_5^{(i)},\alpha_6^{(i)},E,F;\alpha_7^{(i)},G,H,\alpha_8^{(i)})\}_i, \tag{70}$$

where $A, B, ..., H$ denote arbitrary constants.
(2) Guess on 104 key bits, not including $k^{(2)}$, $k^{(12)}$, and $k^{(15)}$.
(3) Using the key bits just guessed, compute 2^{16} plaintexts from (69). These plaintexts belong to set (70).
(4) From the corresponding ciphertexts, compute the 8 bytes 1, 2, 3, 4, 7, 14, 15, 16 after θ_3.
(5) Sum up all 2^{16} 8-byte data obtained, and check balance of these 8 bytes.
(6) If the test succeeds, check whether it is the right key e.g. by exhaustive search[1].
(7) If it is not, go back to step 2.

[1]In fact it is possible to do better. See section III.9.3.3 for details (more precisely step 6 of the attack).

TABLE 7. Bytes that can be computed at the end of round 3 (attack on SAFER++$_{256}$), using the key bytes guessed at step 2 of the attack (in bold).

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$k_1^{(i)} \approx$	$k^{(1)}$	$k^{(2)}$	$\mathbf{k^{(3)}}$	$\mathbf{k^{(4)}}$	$k^{(5)}$	$\mathbf{k^{(6)}}$	$\mathbf{k^{(7)}}$	$k^{(8)}$	$\mathbf{k^{(9)}}$	$\mathbf{k^{(10)}}$	$k^{(11)}$	$k^{(12)}$	$\mathbf{k^{(13)}}$	$k^{(14)}$	$k^{(15)}$	$\mathbf{k^{(16)}}$
$k_2^{(i)} \approx$	$k^{(18)}$	$k^{(19)}$	$\mathbf{k^{(20)}}$	$\mathbf{k^{(21)}}$	$k^{(22)}$	$\mathbf{k^{(23)}}$	$\mathbf{k^{(24)}}$	$k^{(25)}$	$\mathbf{k^{(26)}}$	$\mathbf{k^{(27)}}$	$k^{(28)}$	$k^{(29)}$	$\mathbf{k^{(30)}}$	$k^{(31)}$	$k^{(32)}$	$\mathbf{k^{(33)}}$
$k_7^{(i)} \approx$	$\mathbf{k^{(7)}}$	$k^{(8)}$	$\mathbf{k^{(9)}}$	$\mathbf{k^{(10)}}$	$k^{(11)}$	$k^{(12)}$	$\mathbf{k^{(13)}}$	$k^{(14)}$	$k^{(15)}$	$\mathbf{k^{(16)}}$	$k^{(0)}$	$k^{(1)}$	$k^{(2)}$	$\mathbf{k^{(3)}}$	$\mathbf{k^{(4)}}$	$k^{(5)}$
$k_8^{(i)} \approx$	$\mathbf{k^{(24)}}$	$k^{(25)}$	$\mathbf{k^{(26)}}$	$\mathbf{k^{(27)}}$	$k^{(28)}$	$k^{(29)}$	$\mathbf{k^{(30)}}$	$k^{(31)}$	$k^{(32)}$	$\mathbf{k^{(33)}}$	$k^{(17)}$	$k^{(18)}$	$k^{(19)}$	$\mathbf{k^{(20)}}$	$\mathbf{k^{(21)}}$	$k^{(22)}$
	√		√	√			√			√				√	√	

The time complexity of this attack is 2^{64} encryptions and $2^{104} \cdot 2^{16} = 2^{120}$ additions (these are the biggest part of the work). It requires 2^{64} chosen plaintexts, and 2^{64} memory. By not performing step 1, it is possible to make the amount of memory needed negligible; in this case, time complexity becomes $2^{104} \cdot 2^{16} = 2^{120}$ encryptions.

III.9.3.3. *An attack on 4 rounds of SAFER++$_{256}$:* The attack is very similar to the one on 4 rounds of SAFER++$_{128}$. Only some details in the key guess change due to a different key schedule. Given a 256-bit key $(k^{(1)}, k^{(2)}, ..., k^{(32)})$, the key schedule of SAFER++$_{256}$ has the following properties:

- Two parity bytes are computed: $k^{(0)} = \bigoplus_{i=1}^{16} k^{(i)}$ and $k^{(33)} = \bigoplus_{i=17}^{32} k^{(i)}$.
- Odd subkeys only depend on bytes $k^{(0)}, ..., k^{(16)}$. More precisely, $k_i^{(p)} \approx k^{(i+p-1 \mod 17)}$.
- Even subkeys only depend on bytes $k^{(17)}, ..., k^{(33)}$. More precisely, $k_i^{(p)} \approx k^{((i+p-2) \mod 17+17)}$.

Table 7 details the subkeys of layers γ_1 and γ_4. We can see that the 16 bytes we need to guess in order to compute the plaintexts (step 3 of the attack) allow the computation of 7 bytes at the end of round 3.

The attack is:

(1) Ask for encryption of all 2^{64} plaintexts looking like (70).
(2) Guess the 16 key bytes in bold in Table 2.
(3) Using the key bytes just guessed, compute 2^{16} plaintexts from (69).
(4) From the corresponding ciphertexts, compute the 7 bytes 1, 3, 4, 7, 10, 14, 15 after θ_3.
(5) Sum up all 2^{16} 7-byte data obtained, and check whether these 7 bytes are balanced.
(6) If the test succeeds, guess two more key bytes such as to check balance on one more byte at the end of round 3. Keep guessing new key bytes until one check has failed or the entire key has been guessed (as done in section III.9.3.1). In the second case, check by a trial encryption whether it is the right key.
(7) If the right key has not found, go back to step 2.

TABLE 8. A backward distinguisher on 2 rounds of SAFER++.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	(a_5, a_7)
	p	p	p	p	A	p	A	p	p	p	p	p	p	p	p	p	A
$\xrightarrow{\gamma^{-1}}$	p	p	p	p	A	p	A	p	p	p	p	p	p	p	p	p	A
$\xrightarrow{\theta^{-1}}$	A	p	p	p	p	p	p	$4A$	A	p	A	A	p	A	A	A	
$\xrightarrow{\gamma^{-1}}$	A	p	p	p	p	p	p	A_{64}	A	p	A	A	p	A	A	A	

As a fraction of about $1/2^{128}$ of the keys passes the test, 2^{128} trial encryptions must be done in order to discard wrong keys. However the biggest part of the work consists in about $2^{136} \cdot 2^{16} = 2^{152}$ additions (most of them during step 6).

III.9.4. Other Square Distinguishers on SAFER++

The two following distinguishers are described by A. Biryukov et al. in [22].

III.9.4.1. *A backward distinguisher:* The first distinguisher presented in [22] also deals with 2 rounds, but *in the decryption direction*. The fact that the matrix of θ^{-1} is sparser than the one of θ, makes it easier to find good square distinguishers. Of course, this also implies that attacks derived from these distinguishers are chosen *ciphertext* attacks. Also, this distinguisher exploits the observation that applying a multiplication by 4 on an active byte results in a multiset of which all elements have their 2 rightmost bits equal to 0; thus it takes 64 different values which appear all the same number of times. Such a property is denoted by $4A$. Applying a (possibly nonlinear) bijection to this multiset results in a multiset which still has 64 elements. This property is denoted by A_{64}.

Assume 2^{16} texts such that (a_5, a_7) is active are fed into the distinguisher. Then the state of the data through the 2-round distinguisher is described in Table 8.

Consider the 14^{th} byte after a new application of θ^{-1}, denoted b_{14}. The only non passive byte on which it depends before θ^{-1} is the 8^{th} one, denoted a_8. The link between them is $b_{14} = -4 \cdot a_8$. Then the number of possible different values for b_{14} is at most 64, which would be very unlikely to happen if the cipher is a random permutation. In fact due to particular properties of the SAFER++ S-boxes, this number is exactly 48.

This distinguisher is a good illustration of the fact that considering a linear transform or its inverse is *not* equivalent when considering resistance against square attacks. It is used in [22] to attack 3 rounds of SAFER++.

TABLE 9. A forward distinguisher on 2 rounds of SAFER++.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}
	p	p	p	p	p	p	p	p	p	p	p	p	A	p	p	p
$\xrightarrow{\gamma}$	p	p	p	p	p	p	p	p	p	p	p	p	A	p	p	p
$\xrightarrow{\theta}$	p	p	p	p	p	E	p	p	p	p	E	p	E	E	E	E
$\xrightarrow{\gamma}$	p	p	p	p	p	E	p	p	p	p	E	p	E	E	E	E
$\xrightarrow{\theta}$	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0

III.9.4.2. *A forward distinguisher:* Biryukov et al. considered a set of texts of which all bytes are passive except one that is active. They showed that after two rounds, the lowest bit of each byte is balanced. The square characteristic used is summarized in Table 9, where:

- E denotes an *even* multiset, i.e. a multiset of which all values appear an even number of times.
- B_0 denotes a multiset of which the lowest bit is balanced.

We have thus a 2-round distinguisher which only requires 2^8 encryption queries, while ours required 2^{16}. On the other hand, the probability for a random permutation to succeed the test is 2^{-16}, while it was 2^{-128} for our distinguisher. This distinguisher is used in [22] to attack up to 4.5 rounds of SAFER++.

III.10. Towards Extensions of the Square Attack ?

The best known attacks on the AES [55, 61] are extensions of the square attack. Structural attacks which are not really square-like also exist [24, 21]. The interested reader may refer to the corresponding papers. In this section, we discuss two additional ideas to extend the square attack, and their limitations.

III.10.1. Pushing Square Distinguishers Further?

Let us analyze what happens if we try to extend the distinguisher of section III.9.2 by one more round. The expressions we obtain at the output of such 3-round distinguisher (corresponding to equation (67)) can be written as functions of a and b looking like

$$\Phi^{(j)}(a,b) = \sum_{k=1}^{16} \lambda_k^{(j)} F_{3,k}\Big(\sum_{l=1}^{16} \mu_{k,l}^{(j)} F_{2,l}(\xi_{k,l}^{(j)} a + \xi_{k,l}^{\prime(j)} b)\Big) \qquad (j = 1, ..., 16), \tag{71}$$

where $\lambda_k^{(j)}$, $\mu_{k,l}^{(j)}$, $\xi_{k,l}^{(j)}$ and $\xi_{k,l}^{\prime(j)}$ are constants. To be formal, note that some $F_{r,p}$ have a slightly more general form than given in (62), as they must incorporate the constants we neglected in section III.9.2.

Just like we did with equation (67), we would like to find a criteria (possibly probabilistic) allowing to distinguish functions $\Phi^{(j)}$ that can be written as (71) from random functions. Balance does not longer work, as it is not preserved by permutations (such as $F_{3,k}$): while $\sum_{l=1}^{16} \mu_{k,l}^{(j)} F_{2,l}(\xi_{k,l}^{(j)} a + \xi_{k,l}^{\prime(j)} b)$ is balanced, $F_{3,k}(\sum_{l=1}^{16} \mu_{k,l}^{(j)} F_{2,l}(\xi_{k,l}^{(j)} a + \xi_{k,l}^{\prime(j)} b))$ is not. However it is true that very few functions $\mathbb{Z}_{256} \times \mathbb{Z}_{256} \to \mathbb{Z}_{256}$ can be written as (71). Indeed, the total number of functions $\mathbb{Z}_{256} \times \mathbb{Z}_{256} \to \mathbb{Z}_{256}$ is $2^{2^{19}}$, while the number of 32-uples of functions $\mathbb{Z}_{256} \to \mathbb{Z}_{256}$ (16 functions $F_{2,l}$ and 16 functions $F_{3,k}$) is $2^{2^{16}}$ only[2]. But despite these considerations, it is not clear whether such a criteria exists, with affordable complexity: as a matter of fact, the general problem of constructing a distinguisher for a block cipher also amounts to, given the input-output behavior of a function, deciding whether it matches with a given expression or not. The difference is that the size of the input and output sets is much more affordable in our problem than in the general distinguisher problem, which may make time and data complexity more reasonable.

A potential solution is to consider the number of different values present in a multiset, or the frequencies of these values. Consider two n-bit active multisets $\{\!\{a^{(i)}\}\!\}_{i=1,\dots,e}$ and $\{\!\{b^{(i)}\}\!\}_{i=1,\dots,e}$, such that the $2n$-bit multiset $\{\!\{(a^{(i)}, b^{(i)})\}\!\}_{i=1,\dots,e}$ is not active. Then $\{\!\{a^{(i)} \oplus b^{(i)}\}\!\}_{i=1,\dots,e}$ is not active either, but it is balanced. However this property is not preserved by application of a S-box S and addition of an unknown key k: $\{\!\{S(a^{(i)} \oplus b^{(i)}) \oplus k\}\!\}_{i=1,\dots,e}$ is not balanced. On the other hand, the number of different values in multisets $\{\!\{a^{(i)} \oplus b^{(i)}\}\!\}_i$ and $\{\!\{S(a^{(i)} \oplus b^{(i)}) \oplus k\}\!\}_i$ is the same.

The question is thus the following: is the probability distribution of the number of different values of $\{\!\{a^{(i)} \oplus b^{(i)}\}\!\}_i$ (where $\{\!\{a^{(i)}\}\!\}_i$ and $\{\!\{b^{(i)}\}\!\}_i$ are assumed to be random active multisets) different from the probability distribution of the number of different values in a random multiset $\{\!\{c^{(i)}\}\!\}_i$? We guess it is, but maybe not enough to allow efficient distinguishing. On the other hand, the fact that $\{\!\{a^{(i)}\}\!\}_i$ and $\{\!\{b^{(i)}\}\!\}_i$ are not actually truly random active multisets could be enough to make the probability distribution of the number of values of $\{\!\{a^{(i)} \oplus b^{(i)}\}\!\}_i$ more "recognizable" in some cases, and hence such criteria efficient.

III.10.2. Applying the Square Attack to Bit-Oriented Ciphers

The square attack usually applies to ciphers operating at the word level, where a word is defined as having the size of the input and output of a

[2]If we take into account the fact the $F_{r,l}$ are permutations, this number reduces to $2^{2^{15.7}}$. In fact, due to their particular form, there is far less possibilities for each function $F_{r,l}$. But a hypothetical distinguisher would probably not be able to take these particularities into account.

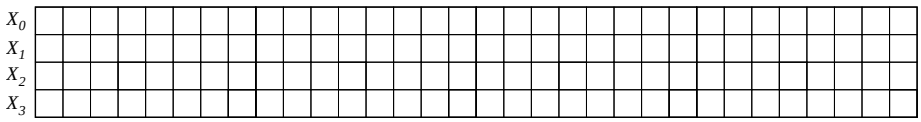


FIGURE 3. The state of Serpent.

S-box. Ciphers like SQUARE, SAFER++, or the AES, are designed this way. The only property of the S-boxes that matters in this context is for them to be bijective.

On the other hand, ciphers like Serpent [2], Noekeon [45], or `ICEBERG` [161] (see Chapter VII) use linear transforms operating at the bit level. Therefore, if we try to apply the square attack to them, the multiset entering a given S-box at some point of the encryption will most of the time not be active. Nevertheless it may happen that its restriction to only some of the S-box's input bits is active, while the complementary restriction is constant. We examine what can be deduced about the state of the S-box's output in this case.

We give an example on Serpent to make things clearer. Serpent is a 128-bit block 128-bit key block cipher. It has a Substitution-Permutation Network structure with 32 rounds. The state of Serpent is usually represented as four 32-bit words X_0, X_1, X_2, X_3; it is pictured in Figure 3. The non-linear layer is made out of 32 copies of the same 4×4 S-box, each of them being applied to one column of the state; but 8 different S-boxes are used, depending on the number of the round: round i uses S-box $i \mod 8$. Algorithm 1 describes the linear layer of Serpent.

Algorithm 1 Linear layer of Serpent.

$X_0 := X_0 \lll 13$
$X_2 := X_2 \lll 3$
$X_1 := X_1 \oplus X_0 \oplus X_2$
$X_3 := X_3 \oplus X_2 \oplus (X_0 \ll 3)$
$X_1 := X_1 \lll 1$
$X_3 := X_3 \lll 7$
$X_0 := X_0 \oplus X_1 \oplus X_3$
$X_2 := X_2 \oplus X_3 \oplus (X_1 \ll 7)$
$X_0 := X_0 \lll 5$
$X_2 := X_2 \lll 22$

Consider a set of 2^{56} plaintexts such that the bits in dark grey in the state at the top of Figure 4 form a 56-bit active multiset, while the other bits are constant. The first key addition and S-box layer do not change this property. Then passing the data through the linear layer θ results in a state as pictured at the bottom of Figure 4, where bits in dark

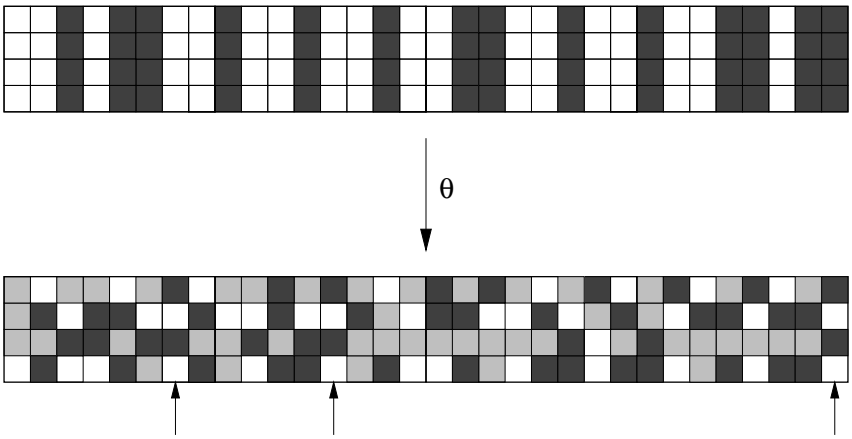


FIGURE 4. Passing an active set through the linear layer.

grey form a 47-bit active multiset[3], bits in light grey are all individually active (we mean that each of them is a 1-bit active multiset), and the others are constant. Consider the three columns marked by arrows; each of them form a 4-bit multiset entering one S-box S_1. They are such that their restriction to their first and third bits (i.e. those from X_0 and X_2) is active, while the complementary restriction is passive. Let $\{\!\{(a_0^{(i)}, a_1^{(i)}, a_2^{(i)}, a_3^{(i)})\}\!\}_{1\leq i\leq 2^{56}}$ denote the multiset at the input of one of these S_1, with its restriction to $\{\!\{(a_0^{(i)}, a_2^{(i)})\}\!\}_{1\leq i\leq 2^{56}}$ being active and $\{\!\{(a_1^{(i)}, a_3^{(i)})\}\!\}_{1\leq i\leq 2^{56}}$ being constant. Let $\{\!\{(b_0^{(i)}, b_1^{(i)}, b_2^{(i)}, b_3^{(i)})\}\!\}_{1\leq i\leq 2^{56}}$ denote the multiset at its output. We would like to identify which are the active restrictions of $\{\!\{(b_0^{(i)}, b_1^{(i)}, b_2^{(i)}, b_3^{(i)})\}\!\}_{1\leq i\leq 2^{56}}$. The problem is that it depends on the value of (a_1, a_3), which is unknown to the attacker, as it depends on key bits. Therefore for each of the four 1-bit restrictions and the six 2-bit restrictions of $\{\!\{(b_0, b_1, b_2, b_3)\}\!\}$, and each of the four possible values for (a_1, a_3), we check whether the considered restriction is active. Then we can give a probability that it is active, computed on the 4 possible values for the constant bits. The results for all 2-bit active restrictions of $\{\!\{(a_0, a_1, a_2, a_3)\}\!\}$ (when the complementary restriction is constant) are given in Table 10. As an example, the second of the six columns gives the results we were looking for in our example. Similar results when one bit only of the input is active (and the other three constant) are given in Table 11. When a 3-bit restriction of the input to S_1 is active, it is easy to see that if a multiset at the output of S_1 is active for a given value of the constant remaining bit, it is active for the other value as well. It is a consequence of the fact that S_1 is a permutation of $\mathbb{Z}_2^4$. Thus in this case an output multiset is active with probability

[3]Although only one is pictured on Figure 4, note that there are many large active multisets after θ.

TABLE 10. Active output multisets of S_1, when 2 bits of its input are active.

Act. multiset	$\{\!\{(a_0, a_1)\}\!\}$	$\{\!\{(a_0, a_2)\}\!\}$	$\{\!\{(a_0, a_3)\}\!\}$	$\{\!\{(a_1, a_2)\}\!\}$	$\{\!\{(a_1, a_3)\}\!\}$	$\{\!\{(a_2, a_3)\}\!\}$
$\{\!\{(b_0, b_1)\}\!\}$	1/4	1/4	1/2	1/2	1/2	0
$\{\!\{(b_0, b_2)\}\!\}$	1/2	0	0	1/2	1/2	0
$\{\!\{(b_0, b_3)\}\!\}$	0	1/4	0	0	0	1/2
$\{\!\{(b_1, b_2)\}\!\}$	1/4	1/2	1/4	1/2	1/2	0
$\{\!\{(b_1, b_3)\}\!\}$	1/2	1/2	1/2	1/4	1/4	0
$\{\!\{(b_2, b_3)\}\!\}$	0	0	1/4	0	0	1/2
$\{\!\{(b_0)\}\!\}$	1/2	1/2	1/2	1/2	1/2	1/2
$\{\!\{(b_1)\}\!\}$	1	1	1	1	1	0
$\{\!\{(b_2)\}\!\}$	1/2	1/2	1/2	1/2	1/2	1/2
$\{\!\{(b_3)\}\!\}$	1/2	1/2	1/2	1/2	1/2	1

TABLE 11. Active output multisets of S_1, when 1 bit of its input is active.

Active multiset	$\{\!\{(a_0)\}\!\}$	$\{\!\{(a_1)\}\!\}$	$\{\!\{(a_2)\}\!\}$	$\{\!\{(a_3)\}\!\}$
$\{\!\{(b_0)\}\!\}$	3/4	1/2	3/4	1/2
$\{\!\{(b_1)\}\!\}$	1	1	1/2	1/2
$\{\!\{(b_2)\}\!\}$	3/4	1/2	1/2	3/4
$\{\!\{(b_3)\}\!\}$	1/2	1/2	3/4	3/4

TABLE 12. Maximal active output multisets of S_1, when 3 bits of its input are active.

Input active multiset	Maximal active output multisets
$\{\!\{(a_0, a_1, a_2)\}\!\}$	$\{\!\{b_1\}\!\}$
$\{\!\{(a_0, a_1, a_3)\}\!\}$	$\{\!\{b_1\}\!\}$
$\{\!\{(a_0, a_2, a_3)\}\!\}$	$\{\!\{(b_1, b_3)\}\!\}$
$\{\!\{(a_1, a_2, a_3)\}\!\}$	$\{\!\{(b_0, b_1, b_2)\}\!\}, \{\!\{(b_0, b_2, b_3)\}\!\}$

either 0 or 1. Table 12 gives the maximal active output multisets for the four possible 3-bit active restrictions of $\{\!\{(a_0, a_1, a_2, a_3)\}\!\}$.

These considerations are interesting because they could a priori lead to square attacks on bit-oriented ciphers. Moreover, such square distinguishers have the originality to be probabilistic, while classical square distinguishers on word-oriented ciphers are not. However, our experiments on Serpent and Noekeon tended to show that such distinguishers could not be found for more than very few rounds of a cipher.

III.11. Conclusion

In this chapter we discussed the square attack and applied it to block ciphers Skipjack and SAFER++. Possible extensions of the attack were also suggested.

In spite of the simplicity and intuitiveness of square attacks, it seems that no proof of security exists against them. Theory of provable security against attacks such as linear and differential cryptanalysis for example is much more developed.

However a few good design principles should prevent most of the attacks of this style. Mainly, it seems that a "reasonably good" diffusion layer iterated over a not too small number of rounds achieves good resistance against square attacks. But "reasonably good" remains to be defined; the branch number is not the best criteria in this context, as attested by the fact that a diffusion layer can be stronger if taken in one direction rather than in the other. Besides, our experiments suggested that ciphers which operate at bit-level seem to have a naturally good resistance against square attacks. But once again, there is no proof.

The conclusion is that, as it is (and, even more, was) often the case in block cipher theory, the current state-of-the-art regarding square attacks is that we know them enough to have a reasonably good feeling on what to do to prevent them, but no proof of security.

CHAPTER IV

Key Schedule Cryptanalysis

Abstract. In this chapter we survey the most important results developed in key schedule cryptanalysis during the last fifteen years. Namely, we deal with related key slide attacks [13, 94], differential related key attacks [89, 90], and slide attacks [25, 26]. We try to focus on the general methods and concepts, rather than develop the technical details of the attack of such and such algorithm. The related key slide attack is presented in a more general framework than in the paper of Biham. Emphasize is given on the conditions under which an algorithm is vulnerable to related key or slide attack. We discuss relevance of these results to cryptanalysis of multiple encryption modes (triple encryption, Even-Mansour construction [52, 91], ...). Finally, the link between these attacks, and the possibility of devising new related key variants of attacks, are discussed.

IV.1. Introduction

The most well known attacks on block ciphers are those targeting the encryption itself, for example the linear [113] and differential [19] cryptanalysis, the interpolation attack [79] or the square attack (see Chapter III). In this chapter we present three types of attacks which are particular in the sense that they focus on the key schedule. Firstly, the related key attack introduced by L. Knudsen in 1992 [94] and E. Biham in 1993 [13], secondly the differential related key attack presented by J. Kelsey, B. Schneier, and D. Wagner [89, 90] and finally different variants of the slide attack developed by A. Biryukov and D. Wagner [25, 26]. The attacks of Biham and Knudsen and the slide attack have a surprising particularity, namely that their efficiency is independent of the number of rounds. We consider all these attacks in a general framework, and analyze under which conditions an algorithm is vulnerable to such or such attack.

Related key attacks work under a very unusual hypothesis: we suppose that encryption is performed under two different keys that have a particular (known) relationship, with the keys themselves unknown. However E. Biham presented in [13] a variant of his related key attack, that allows a reduction of the complexity of exhaustive key search, in the classical context where we are searching for an unique unknown key; this variant

is actually an improvement of an attack from L. Knudsen [94]. In this chapter, we study whether variants of other related key principles can also be used for reducing the complexity of exhaustive search.

Also regarding the slide attack, we give a minor improvement to the *sliding with a twist*, and when dealing with strong (round) functions.

We discuss application of these techniques to multiple encryption modes; we point out some attacks which are not correct in the literature. This discussion raises the question of the necessity to apply a key schedule algorithm to multiple encryption modes. Finally, the link between these attacks, and the possibility of devising new related key variants of attacks, are discussed.

IV.2. Notations

In the following of this chapter, we use the following notations:

- n denotes the block length of the block cipher considered. n_k denotes its key length, and r its number of rounds.
- $K \rightarrow (k_1, k_2, ..., k_{r-1}, k_r)$ means that the key schedule derives the sequence of round keys $(k_1, k_2, ..., k_{r-1}, k_r)$ from the key K.
- $F(x, k)$ denotes a round function applied to the data x with round key k. The function $F(., k) : x \rightarrow F(x, k)$ is sometimes denoted by F_k. X_i denotes the data after i applications of F (thus X_0 is the plaintext and X_r the ciphertext).
- E_K denotes a block cipher parametrized by key K.
- P (and P^*, P_*) always denotes a plaintext, and C (resp. C^*, C_*) the corresponding ciphertext. Also by $P \stackrel{K}{\rightarrow} C$ we denote the fact that the encryption of P under key K gives ciphertext C.

IV.3. Related Key Slide Attacks

A related key attack is an attack relying on a very unusual hypothesis: namely, the attacker has access to plaintext-ciphertext pairs computed using (at least) two different keys. There is a particular relationship between these keys, which is known to the attacker; but the keys themselves are not.

The first related key attack has been introduced by E. Biham in [13]. It has the remarkable property that it does not depend on the number of rounds of the cryptosystem. The name "related key slide attack" came long after, when the slide attack [25, 26] was derived from it (see section IV.4).

For a given block cipher, consider two keys K and K^* such that

$$\begin{aligned} K &\rightarrow (k_1, k_2, ..., k_{r-1}, k_r) \\ \text{and} \quad K^* &\rightarrow (k_2, k_3, ..., k_r, k_1). \end{aligned} \tag{72}$$

Note that this hypothesis is very restrictive, as for most block ciphers both sequences of subkeys cannot be derived simultaneously from a key! However it is the case for ciphers LOKI89 and LOKI91 attacked in [13] (see Example 1 below).

Consider also two plaintexts P and P^* such that $P^* = F(P, k_1)$. Then the encryption of P under key K and the one of P^* under key K^* will process the same way during $n-1$ rounds:

$$\begin{aligned} P &\rightarrow F_{k_1} \cdot F_{k_2} \cdot ... \cdot F_{k_{r-1}} \cdot F_{k_r} \rightarrow C \\ P^* &\rightarrow F_{k_2} \cdot F_{k_3} \cdot ... \cdot F_{k_r} \cdot F_{k_1} \rightarrow C^* \end{aligned}$$

And as a consequence, the ciphertexts will satisfy the same property as the plaintexts: $C^* = F(C, k_1)$. Two such pairs (P, C) and (P^*, C^*) will be called a ***slid pair***. $P^* = F(P, k_1)$ and $C^* = F(C, k_1)$ are called ***slid equations***. Slid pairs are the basis of several attacks we present in this chapter.

IV.3.1. The Basic Attack

Assume an attacker has access to pairs (P, C) computed under a key K, and to pairs (P^*, C^*) computed under a key K^*, such that K and K^* satisfy (72). Identifying a slid pair among pairs $((P, C); (P^*, C^*))$ is not evident, as the round key k_1 implied in the slid equations is unknown. However the following attack applies:

- Suppose that $2^{n/2}$ pairs (P_i, C_i) encrypted under key K are known, as well as $2^{n/2}$ pairs (P_i^*, C_i^*) encrypted under key K^*.
- For each pair $((P_i, C_i); (P_j^*, C_j^*))$, try to solve the system of equations
$$\begin{cases} F(P_i, k') = P_j^* \\ F(C_i, k') = C_j^* \end{cases} \tag{73}$$
until a key k' satisfying the system is found. With a high probability this k' is the subkey k_1.

It is important to note that most of the time the equations $F(P_i, k') = P_j^*$ and $F(C_i, k') = C_j^*$ cannot both hold if P_i and P_j^* do not form a slid pair, because of the structure of the round function attacked (it is even sometimes the case that one single equation gives an unique solution for the key). Thus when we effectively find a solution to the system there is a high probability that we are dealing with a slid pair, and that the solution k' we found is actually k_1. The probability for a random pair to

be a slid pair is 2^{-n}, so with $2 \cdot 2^{n/2}$ plaintext-ciphertext pairs we may expect to find one.

Note also that even when we are dealing with a slid pair, the function F must be weak enough to permit us to retrieve the key k_1. Therefore we give the following definition, that must be satisfied by the round function F in order for the attack presented to work:

DEFINITION 28. We call F a ***weak*** permutation if given both equations $F(x_1, k') = y_1$ and $F(x_2, k') = y_2$ it is "easy" to extract the key k'.

Of course it is informal since the difficulty may vary from one cipher to the other. The mean complexity of this basic attack is $\Theta(2^{n/2})$ data, and $\Theta(2^n)$ work, as there are $\Theta(2^n)$ pairs to examine.

If the cipher attacked is a Feistel network, the particular form of the round function allows more efficient attacks. This specific case is discussed in the next section.

IV.3.2. The Case of Feistel Ciphers

IV.3.2.1. *Known plaintext attack:* In the case of Feistel ciphers, the round function $F(\langle L, R\rangle) = \langle R, L \oplus f(R)\rangle$ only modifies one half of its input. Therefore, the condition $F(x, k) = y$ can be recognized simply by comparing the right half of x against the left half of y, without knowing anything about the round key k. This filtering condition eliminates all but $2^{-n/2}$ of the pairs.

This property can be used to improve the efficiency of the attack: indeed, (P, C) and (P^*, C^*) form a slid pair if and only if $F(P, k_1) = P^*$ and $F(C, k_1) = C^*$. We have a $n/2$-bit filtering condition for each of these equalities, and thus globally a n-bit filtering condition.

Therefore, potential slid pairs can be identified using two sorted lists with $2^{n/2}$ entries: we sort the texts (P_i, C_i) encrypted with key K based on the right half of P_i and C_i, and the texts (P_j^*, C_j^*) encrypted with key K^* based on the left half of P_j^* and C_j^*. It is then easy to look for a match between the left halves of P_j^* and C_j^* and the right halves of P_i and C_i (respectively).

With this filtering technique, we expect to find no more than one false match (i.e. a non-slid pair that passes the test) along with one good slid pair. The false match(-es) can be easily eliminated in a second phase. While we still need $\Theta(2^{n/2})$ plaintext-ciphertext pairs, the time complexity is now reduced to $\Theta(n \cdot 2^{n/2})$ offline work.

IV.3.2.2. *Chosen plaintext attack:* In the previous section, the fact that the round function $F(\langle L, R\rangle) = \langle R, L \oplus f(R)\rangle$ modifies only one half of its input has been used as a filtering condition. In the context of a chosen plaintext attack, it can be used to select appropriate plaintexts in order to reduce data complexity.

Suppose that we choose a pool of $2^{n/4}$ plaintexts $P_i = \langle P_{Li}, P_R\rangle$ (with P_R an arbitrary constant) encrypted under key K, and another pool of $2^{n/4}$ plaintexts $P_j^* = \langle P_R, P_{Rj}^*\rangle$ encrypted under key K^*. Thus we have $2^{n/2}$ pairs of plaintext $(P_i; P_j^*)$. A slid pair occurs with probability $2^{-n/2}$ (namely, when $f(P_R) = P_{Li} \oplus P_{Rj}$), so we expect to find one slid pair with only $\Theta(2^{n/4})$ chosen plaintexts and $\Theta(n \cdot 2^{n/4})$ work rather than $\Theta(2^{n/2})$ known plaintexts and $\Theta(n \cdot 2^{n/2})$ work; it can be recognized using the $n/2$-bit filtering condition on the ciphertexts.

Note that such a chosen plaintext attack can be performed only when we know something about the round function, as the structure of the set of plaintexts we chose depends on the particular structure of the round function. However both attacks we presented in sections IV.3.2.1 and IV.3.2.2 are not restricted to Feistel ciphers. It is often possible to exploit the particular structure of a given round function, especially if the cipher is "more or less Feistel-like", rather than a substitution-permutation network.

IV.3.2.3. *Probable-plaintext attack:* When plaintexts contain some redundancy, the data complexity of the related key slide attack can often be significantly reduced.

Let us consider a simple model: the plaintext source emits blocks where the four most significant bits of each byte are always zero, so that the resulting n-bit plaintext only has $n/2$ bits of entropy.

We show that with $2^{3n/8}$ known plaintext-ciphertext pairs (P_i, C_i) encrypted with K, and $2^{3n/8}$ known pairs (P_j^*, C_j^*) encrypted with K^*, we can expect to find one slid pair:

- There are $2^{3n/4}$ pairs $((P_i, C_i); (P_j^*, C_j^*))$.
- Let us consider one pair $P_i = \langle L_i, R_i\rangle$, $P_j^* = \langle L_j^*, R_j^*\rangle$. We compute the probability that $F(P_i, k_1) = P_j^*$.
- $\mathcal{P}[L_j^* = R_i] = 2^{-n/4}$, due to the redundancy in the plaintexts.
- $\mathcal{P}[R_j^* = L_i \oplus f(R_i, k_1)] = 2^{-n/2}$, assuming f behaves randomly.
- We conclude that $F(P_i, k_1) = P_j^*$ with probability $2^{-3n/4}$, so we can expect to find about one slid pair.

Thus the attack needs $\Theta(2^{3n/8})$ plaintext-ciphertext pairs, and its time complexity is $\Theta(n \cdot 2^{3n/8})$ work.

IV.3.2.4. *Probable-plaintext attack - ciphertext-only variant:* This attack can be converted to a ciphertext-only attack. We consider $2^{3n/8+1}$ ciphertexts encrypted under key K, and $2^{3n/8+1}$ encrypted under K^*. Following the reasoning above, we expect to find about 4 slid pairs among these.

Using the $n/2$-bit filtering condition on the ciphertexts, we come up with a set of $2^{n/4+2}$ potential slid pairs. The list of potential slid pairs can be identified with $\Theta(n \cdot 2^{3n/8})$ steps by sorting them.

Next, we make a guess at a correct slid pair $(C_i; C_j^*)$. For each remaining potential slid pair $(C_{i'}; C_{j'}^*)$, we compute the key value k' suggested by equations

$$\begin{cases} F(C_i, k') = C_j^* \\ F(C_{i'}, k') = C_{j'}^* \end{cases} \tag{74}$$

We store all these values in a table, in which we search for collisions. If our guess at $(C_i; C_j^*)$ was a correct slid pair, the right key value will be suggested about three times (corresponding to the three other slid pairs). If not, the probability for a wrong key value to be suggested three times is negligible.

Note that this algorithm uses a lot of key recoveries (by *key recovery*, we mean the problem of finding the key k', given $F(u, k') = v$, $F(u', k') = v'$). It is thus important that such key recovery be fast. The attack takes $\Theta(2^{n/2})$ work ($\Theta(2^{n/4})$ guesses of $(C_i; C_j^*)$, performing $\Theta(2^{n/4})$ operations per guess to build the table) and needs $\Theta(2^{n/4})$ space.

As a final remark, note that the complexity of these two probable-plaintext attacks widely depends on the exact plaintext distribution.

IV.3.2.5. *Attacking more general key schedules:* It must be noted that the attack presented in section IV.3.1 works only if the key K^* is derived from key K by a perfect rotation of the subkeys:

$$K \to (k_1, k_2, ..., k_{r-1}, k_r) \Leftrightarrow K^* \to (k_2, k_3, ..., k_r, k_1). \tag{75}$$

We call this type of relationship between K and K^* the ***strong assumption***.

Suppose now that the relationship between K and K^* is slightly weaker, i.e. that we only have

$$K \to (k_1, k_2, ..., k_{r-1}, k_r) \Leftrightarrow K^* \to (k_2, k_3, ..., k_r, k_{r+1}), \tag{76}$$

where k_{r+1} is not necessarily equal to k_1. This second type of relationship will happen every time the key schedule consists in iteratively applying a given function ϕ: $k_i = \phi(k_{i-1})$. It will be called ***weak assumption***.

Example 1. Let us consider the case of LOKI89, attacked by E. Biham in [13]. LOKI89 has 16 rounds, and its key schedule can be described as

$$\begin{cases} k_1 = k_L \\ k_2 = k_R \\ k_i = k_{i-2} \lll 12 \quad (3 \leq i \leq 16) \end{cases}$$

(where the master key is $K = (k_L, k_R)$).

It is easy to see that for any key $K = (k_1, k_2)$, if $K \to (k_1, k_2, ..., k_{15}, k_{16})$, then $K^* := (k_2, k_1 \lll 12) \to (k_2, ..., k_{16}, k_1)$, and thus K and K^* satisfy the strong assumption.

Suppose now that we consider the same algorithm, but reduced to 11 rounds. This time, if $K = (k_1, k_2) \to (k_1, k_2, ..., k_{10}, k_{11})$, then $K^* := (k_2, k_1 \lll 12) \to (k_2, ..., k_{11}, k_{12})$, where $k_{12} \neq k_1$ except for very particular choices of K. The strong assumption is thus no longer satisfied, but the weak one is.

Under the weak assumption hypothesis, the attack we presented does not work anymore in the general case: indeed, a slid pair $((P, C); (P^*, C^*))$ would satisfy both equations

$$\begin{cases} F(P, k_1) = P^* \\ F(C, k_{r+1}) = C^* \end{cases} \tag{77}$$

But for any pair (x, y) the equation $F(x, k) = y$ has at least one solution. Therefore if k_1 and k_{r+1} share few or no bits, it is not possible to identify slid pairs by simply trying to solve this system, as we did in section IV.3.1. However we have seen in section IV.3.2.1 that in particular cases, such as Feistel ciphers, it is possible to use filtering conditions. In this case we can mount the following attack:

(1) Suppose we know $2^{n/2}$ plaintext-ciphertext pairs (P, C) encrypted with key K, and $2^{n/2}$ plaintext-ciphertext pairs (P^*, C^*) encrypted with key K^*.
(2) Sort the texts (P_i, C_i) encrypted with key K based on the right half of P_i and C_i, and the texts (P_j^*, C_j^*) encrypted with key K^* based on the left half of P_j^* and C_j^*. Then look for a match between the left halves of P_j^* and C_j^* and the right halves of P_i and C_i (respectively). Doing this, we filter pairs $((P, C); (P^*, C^*))$ for which the distinguishing condition for $F(P) = P^*$ and $F(C) = C^*$ hold.
(3) For each of the pairs selected (we hope to find around one false alarm along with one slid pair), both equations

$$F(P, k_1) = P^* \quad \text{and} \quad F(C, k_{r+1}) = C^*$$

are solved separately. Note that for certain round functions, we will get many possibilities for k_1 and k_{r+1}. But if k_1 and

k_{r+1} have several bits in common, the number of possibilities for (k_1, k_{r+1}) is not too big.

(4) For each suggestion for (k_1, k_{r+1}), we try to find the remaining key bits by exhaustive search, until we succeed.

This algorithm has a data complexity of $\Theta(2^{n/2})$, and a time complexity of $\Theta(n \cdot 2^{n/2})$.

IV.3.3. A Related Key Slide Attack on DES?

In [95] L. Knudsen showed that for every DES key K, there exists a related key K^* such that these two keys have 12 common round keys. This result has been improved by R.C.-W. Phan and S. Furuya [38].

THEOREM 29. [38] *For every DES key K, there exists a key K^* such that*

$$k_i^* = k_{i+7} \qquad i \in \{1, ..., 9\}$$

and

$$k_i^* = k_{i-8} \qquad i \in \{10, ..., 15\}.$$

Thus K and K^* have 15 common round keys.

Thus if E_a denotes 6-round DES keyed by $k_2, ..., k_7$ (or equivalently $k_{10}^*, ..., k_{15}^*$), E_b denotes 9-round DES keyed by $k_8, ..., k_{16}$ (or equivalently $k_1^*, ..., k_9^*$), E_1 denotes the first round of DES keyed by K and E_{16}^* denotes the last round of DES keyed by K^*, then the two encryptions can be slided against each other and we obtain:

$$\begin{aligned} P \to E_1 \cdot E_a \cdot E_b \to C \\ P^* \to E_b \cdot E_a \cdot E_{16}^* \to C^* \end{aligned}$$

The attack we described in section IV.3.2.5 would have applied provided the two encryptions have the following patterns: $DES_K(P) = E_a(E_1(P))$ and $DES_{K^*}(P^*) = E_{16}^*(E_a(P))$, with E_a being 15-round. Then the slid equations are

$$\begin{cases} P^* = E_1(P) \\ C^* = E_{16}^*(C) \end{cases} \tag{78}$$

However in the present case the slid equations are

$$\begin{cases} P^* = E_a(E_1(P)) \\ C^* = E_{16}^*(E_a(C)) \end{cases} \tag{79}$$

Again these two equations can be slided against each other. It is why the authors of [38] called their (tentative) attack *double slide attack*. We obtain:

$$\begin{aligned} P \to E_1 \cdot E_a \to P^* \\ C \to E_a \cdot E_{16}^* \to C^* \end{aligned}$$

which gives two other slid equations

$$\begin{cases} C = E_1(P) \\ C^* = E_{16}^*(P^*) \end{cases} \tag{80}$$

We note that (80) has the same pattern as (78). Then finding a pair $((P, C); (P^*, C^*))$ satisfying (80) would allow retrieval of key material. However a pool of 2^{64} plaintexts is needed to find a pair (P, C) satisfying $C = E_1(P)$ (as we have no control on the link between P and C - they are related by a DES encryption), which makes the attack impractical. In their paper [38] R.C.-W. Phan and S. Furuya suggest another attempt to use the property of DES pointed in Theorem 29. It is based on the *domino effect* (see section IV.4.5), but it does not work either.

IV.3.4. A More Classical Attack

The hypothesis we made in section IV.3.1 (i.e. that we are dealing with two unknown keys with a very particular relationship between them) looks very restrictive. Nevertheless this attack could be useful if combined with an attack on key exchange protocols that do not guarantee key integrity, or if bad key update protocols are used, as mentioned in [90]. Moreover, the existence of such an attack can be considered as a weakness from a theoretical point of view.

We are now going to use the related key principle in a more classical context: namely, we have a certain number of chosen plaintext-ciphertext pairs encrypted under an unknown key K that we are searching for. The idea of such an attack was first suggested by L. Knudsen in an attack on LOKI91 [94]. This attack was then improved by Biham [13].

First we will show how to deal with the case of a Feistel cipher, following the works of L. Knudsen and E. Biham. Then we analyze whether this principle can be used for non-Feistel ciphers.

For a key K such that $K \to (k_1, ..., k_r)$, we will denote K^* as being the key such that $K^* \to (k_2, k_3, ..., k_{r+1})$ (for some k_{r+1}) and K_* as being the key such that $K_* \to (k_0, k_1, ..., k_r)$ (for some k_0)[1]. K^* and K_* are assumed to be easily computable.

IV.3.4.1. *The case of Feistel ciphers:* The attack is as follows:

- The following plaintext-ciphertext pairs are chosen:
  - $P = \langle P_L, P_R \rangle \xrightarrow{K} C$ for some plaintext P.
  - $\forall a \in \{0, 1, ..., 2^{n/2} - 1\} : P_a^* = \langle P_R, a \rangle \xrightarrow{K} C_a^*$.
  - $\forall a \in \{0, 1, ..., 2^{n/2} - 1\} : P_{*a} = \langle a, P_L \rangle \xrightarrow{K} C_{*a}$.

[1] We emphasize on the fact that only some particular algorithms are such that for a given K the keys K^* and K_* exist; thus the attack we present is only applicable to very specific algorithms.

- Let Φ be a set of keys such that $\Phi \cup \{K^* | K \in \Phi\} \cup \{K_* | K \in \Phi\}$ covers the complete key space.
- For each key $K' \in \Phi$:
  - Compute $P \xrightarrow{K'} C'$. If $C' = C$ we can guess that with high probability $K = K'$ and stop.
  - Compute $F(P, k'_1)$, that equals $P^*_{a'}$ for some a' (in fact, we already did it when computing $P \xrightarrow{K'} C'$). Compute also $F^{-1}(P, k'_0) = P_{*a''}$.
  - Compute one round backward and one round forward from the ciphertext C'. We obtain $C'^* = F(C', k'_{r+1})$ and $C'_* = F^{-1}(C', k'_r)$ (in fact, C'_* was already computed).
  - If $C'^* = C^*_{a'}$, we deduce that $K = K'^*$. If $C'_* = C_{*a''}$, we deduce that $K = K'_*$.

The underlying philosophy is the following: instead of proceeding through all the possible keys, as in the basic exhaustive key search, we only deal with a part of the whole key space. For each trial encryption we make, we get two more encryptions as a bonus, by computing only two more rounds ($P_{*a''} = F^{-1}(P, k'_0)$ and $C'^* = F(C', k'_{r+1})$).

Remark 3. Note that the weak assumption is sufficient for this attack.

Remark 4. It is not necessarily easy to construct a set Φ, as small as possible, such that $\Phi \cup \{K^* | K \in \Phi\} \cup \{K_* | K \in \Phi\}$ covers the complete key space, and such that the elements of Φ are easy to characterize. Ideally, the size of Φ would be one third of the size of the key space; in his attack on LOKI, E. Biham obtained a size of 3/8 of the key space.

Remark 5. This attack uses the same kind of approach as those based on complementation properties. As a matter of fact, it is based on two equivalencies: $P \xrightarrow{K'} C' \Leftrightarrow F(P, k'_1) \xrightarrow{K'^*} F(C', k'_{r+1})$ and $P \xrightarrow{K'} C' \Leftrightarrow F^{-1}(P, k'_0) \xrightarrow{K'_*} F^{-1}(C', k'_r)$. Complementation attacks are also based on equivalencies. For example, the complementation property of DES may be written as: $P \xrightarrow{K'} \overline{C} \Leftrightarrow \overline{P} \xrightarrow{\overline{K'}} C$.
Furthermore, these attacks (related key and complementation) can be combined, as E. Biham did in the case of LOKI (that has many complementation properties).

IV.3.4.2. *The general case:* At first sight, the fact that we are dealing with a Feistel cipher might appear to play no role in the previous attack. Simply, the set $\{P^*_a\}_a$ is defined in a general framework as: $\{F(P, k)\}_{k \in RK}$, where P is a given plaintext and RK is the set of all possible round keys; and the set $\{P_{*a}\}_a$ is $\{F^{-1}(P, k)\}_{k \in RK}$.

However the attack is practical only under certain conditions:

TABLE 1. Summary.

Algorithm	Hyp. on the Key Schedule	Chosen key attack	Classical attack
General Case	Strong	$\Theta(2^{n/2})$ known (P, C) $\Theta(2^n)$ work	•$\lvert\{P_a^*\}_a \cup \{P_{*a}\}_a\rvert$ +1 chosen (P, C) • At best, $2^{n_k}/3$ keys to explore
	Weak	Impossible, except if k_1 and k_{r+1} share a sufficient number of key bits	
Feistel	Strong	$\Theta(2^{n/2})$ known (P, C) $\Theta(n \cdot 2^{n/2})$ work	•$\lvert\{P_a^*\}_a \cup \{P_{*a}\}_a\rvert$ +1 chosen (P, C) • At best, $2^{n_k}/3$ keys to explore
	Weak	$\Theta(2^{n/2})$ known (P, C) $\Theta(n \cdot 2^{n/2})$ work	

- If the sets of pairs $\{(P_a^*, C_a^*)\}_a$ and $\{(P_{*a}, C_{*a})\}_a$ that are needed for the attack cover the complete codebook, it is not really interesting anymore to discover the key, as we have a list of all plaintext-ciphertext pairs. It is generally the case with substitution-permutation structures.
- If the cardinality of $\{P_a^*\}_a \cup \{P_{*a}\}_a$ approaches (or is greater than) the size of the key space, it is easier to perform an exhaustive key search.

Thus the attack is possible for ciphers with a non-Feistel structure as well, but makes sense only if the following two inequalities are met:

$$\begin{aligned} &|\{P_a^*\}_a \cup \{P_{*a}\}_a| < 2^n \\ &|\{P_a^*\}_a \cup \{P_{*a}\}_a| \ll 2^{n_k}. \end{aligned} \tag{81}$$

IV.3.5. Summary

Table 1 summarizes our discussions. In the next section, we deal with the slide attack; it relies on the same idea as the related key slide attack, but applies to the classical context where one only key is used. Then in sections IV.5 and IV.6 we consider other related key attacks and applications of them.

IV.4. Slide Attacks

The slide attack has been developed by A. Biryukov and D. Wagner in [25, 26]. It relies on the same principle as the related key slide attack,

but applies to even more specific key schedules. The good news is that it does not require related key queries.

IV.4.1. Basic principle

Let us consider a block cipher that may be decomposed into r identical permutations $X_j = F(X_{j-1}, k)$. Note that F does not necessarily correspond to one single round of the cipher; it might include several rounds. Thus a necessary condition for a key schedule to be vulnerable to the slide attack is to be periodic. For simplicity purpose, in the remaining of this chapter we will speak about one round to designate the function F.

Similarly to Biham's related key attack, the idea of the slide attack is to "slide" a copy of the encryption process against another copy of the encryption process, so that both processes are one round out of phase:

$$\begin{aligned} P &\rightarrow F_k \cdot F_k \cdot ... \cdot F_k \cdot F_k \rightarrow C \\ P^* &\rightarrow F_k \cdot F_k \cdot ... \cdot F_k \cdot F_k \rightarrow C^* \end{aligned}$$

The only difference with the related key slide attack is that both encryptions have been computed under the same key $K \rightarrow (k, k, ..., k)$. We have the slid equations:

$$\begin{cases} P^* = F(P, k) \\ C^* = F(C, k) \end{cases} \tag{82}$$

As always for slid equations, if two plaintexts are such that the first equation is satisfied, the two corresponding ciphertexts satisfy the second equation. If the function F is *weak* (see definition 28), these two equations permit us to retrieve key k. The attack is similar to the one described in section IV.3.1. By the birthday paradox, about $2^{n/2}$ pairs (P_i, C_i) are enough to find one slid pair. In general, we expect one slid pair to disclose n key bits. If needed, we can search for a few more slid pairs or use exhaustive search to recover the rest of the key.

The time complexity of the attack is $\Theta(2^n)$, as there are $\Theta(2^n)$ pairs to examine.

IV.4.2. An Extension to More General Ciphers

In [128], an attack is presented on ciphers of the form $E_{k,x,y}(P) = y \oplus F_k^r(P \oplus x)$ where F is a Feistel round, P is the plaintext, k is the round key and x, y are whitening keys. It is worth noting that this type of ciphers does not match the general pattern presented as mandatory in the previous section (and in [25, 26]) in order to be vulnerable to slide attacks.

Let us thus examine what happens if we try to apply the slide attack to a cipher of the form $G_x \cdot F_k^r \cdot H_y$ (where F, G, and H are round functions respectively parameterized by key material k, x, y). Such a cipher is a generalization of the one examined in [128]. Moreover, it is the most general pattern a cipher must comply with if we want to get a chance to see the slide attack work on it. The sliding of the two encryption processes looks like:

$$\begin{aligned} P &\rightarrow G_x \cdot F_k \cdot F_k \cdot ... \cdot F_k \cdot H_y \rightarrow C \\ P^* &\rightarrow G_x \cdot F_k \cdot ... \cdot F_k \cdot F_k \cdot H_y \rightarrow C^* \end{aligned}$$

The corresponding slid equations are:

$$\begin{cases} (G_x \cdot F_k \cdot G_x^{-1})(P) = P^* \\ (H_y^{-1} \cdot F_k \cdot H_y)(C) = C^* \end{cases} \tag{83}$$

These two equations must meet two conditions in order to be useful in a basic slide attack:

- The functions F, G and H must be weak enough, so that it is possible to retrieve key material from a slid pair.
- The system must be sufficiently overdetermined, so that we usually get no solution from a bad pair.

Of course, in this case as well as in the basic one of section IV.4.1, the particular structure of the round functions can be used in order to improve the performance of the attack. In the following section we deal with the case of Feistel ciphers.

IV.4.3. The Particular Case of Feistel Ciphers

When the cipher considered has a Feistel structure, the improved attacks described in sections IV.3.2.1 to IV.3.2.4 still apply. The only difference is that instead of two sets (one corresponding to encryption under K, the other corresponding to encryption under K^*) of N plaintext-ciphertext pairs each, only one set of N pairs is considered.

IV.4.4. Advanced Slide Techniques for Feistel Ciphers

Let us introduce the following definition that gives us a good classification of key schedules with regard to their resistance against slide attacks.

Definition 30. A ***p-round-self-similar*** cipher is a cipher whose key schedule is periodic with period p.

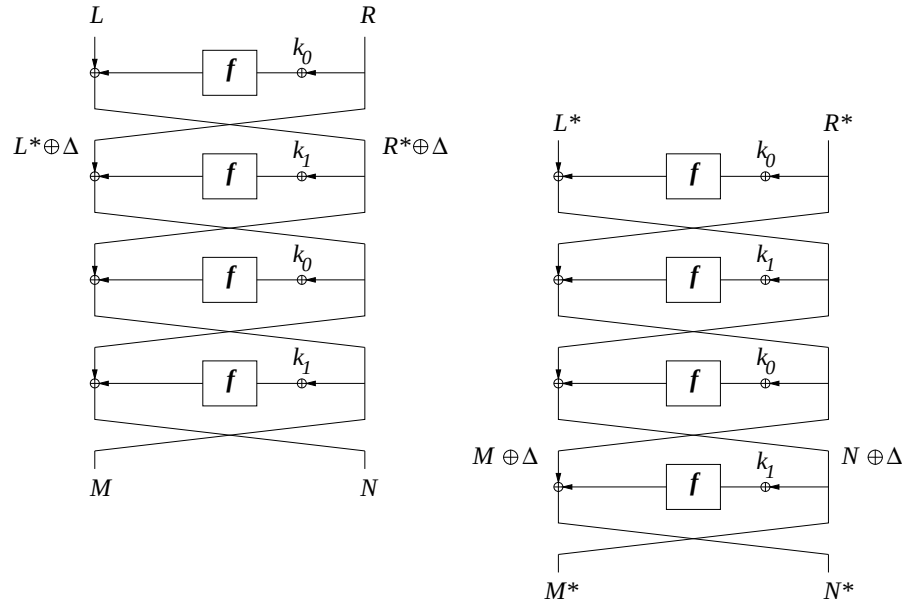


FIGURE 1. The complementation slide attack.

The basic slide attack generally deals with 1-round-self-similar ciphers (except if the permutation F corresponds to more than one round). The goal of advanced attacks is to be able to attack p-round-self-similar ciphers, with $p > 1$. Various advanced techniques are proposed in [26], that only work in the case of Feistel ciphers.

IV.4.4.1. *The complementation slide:* Let us consider a 2-round-self-similar Feistel cipher. The classical attack would require to slide by 2 rounds. Now look what happens if we slide the cipher by only one round (see Figure 1): the problem is that the rounds keyed by k_0 and k_1 are no longer lined up. So even if $P^* = F(P, k_0)$, the continuation of the computation will not be the same for both executions of the algorithm.

However there is a solution to this problem: namely, to choose a pair $(P; P^*)$ such that the difference $F(P) \oplus P^*$ cancels the difference between the subkeys.

DEFINITION 31. A pair of plaintexts $(P; P^*)$ has ***slid difference*** d if $F(P) \oplus P^* = d$.

It is easy to check that if instead of considering plaintext pairs with slid difference 0 as we did until now, we search for pairs with slid difference $\langle \Delta, \Delta \rangle := \langle k_0 \oplus k_1, k_0 \oplus k_1 \rangle$, the slid difference will propagate through all the rounds. More precisely $\forall r : X^*_{r-1} = X_r \oplus \langle \Delta, \Delta \rangle$. A slid pair must thus satisfy the following equations, with $P = \langle L, R \rangle$ denoting the

plaintext and $C = \langle M, N\rangle$ the ciphertext:

$$\begin{cases} \langle L^*, R^*\rangle = \langle R, L \oplus f(k_0 \oplus R)\rangle \oplus \langle \Delta, \Delta\rangle \\ \langle M^*, N^*\rangle = \langle N, M \oplus f(k_1 \oplus N \oplus \Delta)\rangle \oplus \langle \Delta, \Delta\rangle \end{cases} \tag{84}$$

Thus we have

$$L^* \oplus M^* = R \oplus N, \tag{85}$$

which is a $n/2$-bit condition to recognize a slid pair (note that the filtering condition was n-bit long in the classical attack; it is now reduced to $n/2$ bits because the slid difference is unknown). Moreover, the equality $L^* = R \oplus \Delta$ (or equivalently $M^* = N \oplus \Delta$) gives us a $n/2$-bit candidate for Δ. Then if the round function is weak enough we are able to derive k_0 using both equations.

The attack uses $\Theta(2^{n/2})$ known plaintexts. However only $\Theta(2^{n/2})$ pairs need to be examined, due to the filtering condition; each of them suggests at most one candidate key. We hope to find one slid pair.

IV.4.4.2. *Sliding with a twist:* This attack applies again to 2-round-self-similar Feistel ciphers with an even number of rounds. We now slide a decryption process against an encryption process by one round (see Figure 2). If the data after the first round of the encryption are the same as those before the *decryption* process (neglecting one swap of left and right parts), then the data after the encryption process are the same as those before the last round of the decryption process. The slid equations are thus:

$$\begin{cases} \langle M^*, N^*\rangle = \langle L \oplus f(k_0 \oplus R), R\rangle \\ \langle L^*, R^*\rangle = \langle M \oplus f(k_0 \oplus N), N\rangle \end{cases} \tag{86}$$

These give us a n-bit filtering condition on slid pairs (namely $N^* = R$ and $R^* = N$). Thus given $\Theta(2^{n/2})$ known texts, a slid pair can be found easily with $\Theta(n \cdot 2^{n/2})$ work by sorting them. The subkey k_0 may be easily deduced (provided f is sufficiently weak, as usual).

The method proposed in [26] in order to find the other subkey k_1 is somewhat complicated: it consists in using a conventional sliding, and encrypt ciphertexts partially using k_0. The method we propose is both simpler and slightly faster: simply, we suggest to slide the decryption process in the other direction (see Figure 3)! The slid equations are now:

$$\begin{cases} \langle L, R\rangle = \langle M^*, N^* \oplus f(k_1 \oplus M^*)\rangle \\ \langle M, N\rangle = \langle L^*, R^* \oplus f(k_1 \oplus L^*)\rangle \end{cases} \tag{87}$$

and the n-bit filtering condition becomes $L = M^*$ and $L^* = M$. This time it is the subkey k_1 that can be deduced.

There is a chosen plaintext/ciphertext variant that allows us to reduce the number of plaintexts needed to recover k_0 to $2^{n/4+1}$. The attacker generates a pool of $2^{n/4}$ plaintexts of the form $\langle L_i, R\rangle$ and another of $2^{n/4}$

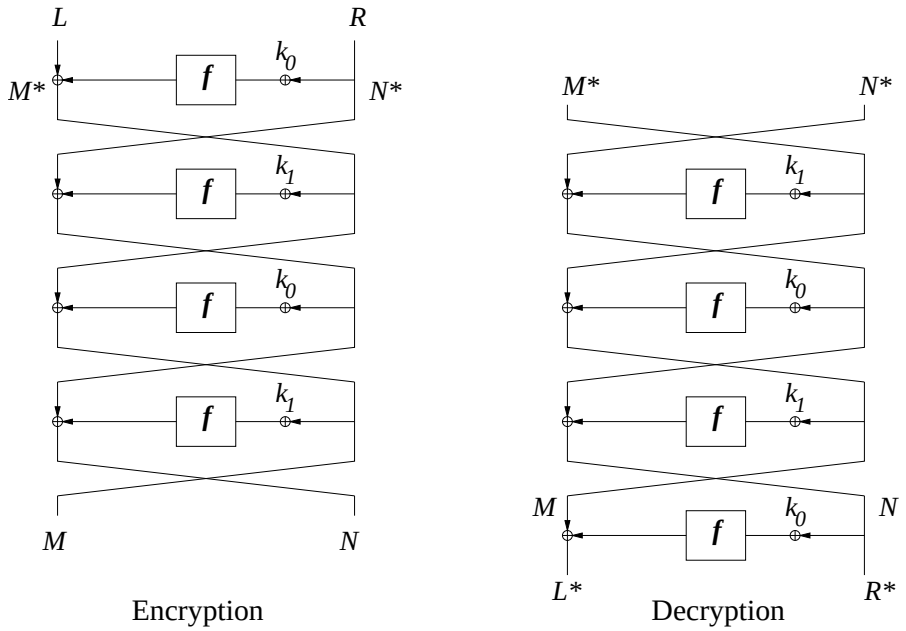


FIGURE 2. Sliding with a twist.

ciphertexts of the form $\langle M_j^*, N^* \rangle$ (of which she obtains the encryptions or decryptions, respectively). The value $N^* = R$ is fixed throughout the attack. This is expected to provide us with one slid pair, from which k_0 may be easily deduced.

Note that the same type of chosen plaintext/ciphertext attack can be applied to recover k_1 instead of k_0.

This demonstrates that sliding with a twist allows us to attack any n-bit 2-round-self-similar Feistel cipher (with an even number of rounds) with $\Theta(2^{n/2})$ known plaintexts and $\Theta(n \cdot 2^{n/2})$ work, or with $\Theta(2^{n/4})$ chosen plaintext/ciphertext and $\Theta(n \cdot 2^{n/4})$ work. However it must be pointed out that the chosen plaintext/ciphertext version requires the attacker to be allowed to make both encryption and decryption queries. This variant also applies to related key slide attacks; an example is the attack of [37] against triple encryption.

IV.4.4.3. *Dealing with 4-round periodicity:* It is possible to combine the complementation slide and sliding with a twist to be able to attack 4-round-self-similar ciphers. The exact way it is done is pictured in Figure 4. We observe that the odd rounds always line up, while the even ones have a constant difference of $k_1 \oplus k_3$ in their subkeys. It is however possible to cancel this difference using the complementation technique. The texts used need to have a slid difference of $\langle k_1 \oplus k_3, 0 \rangle$.

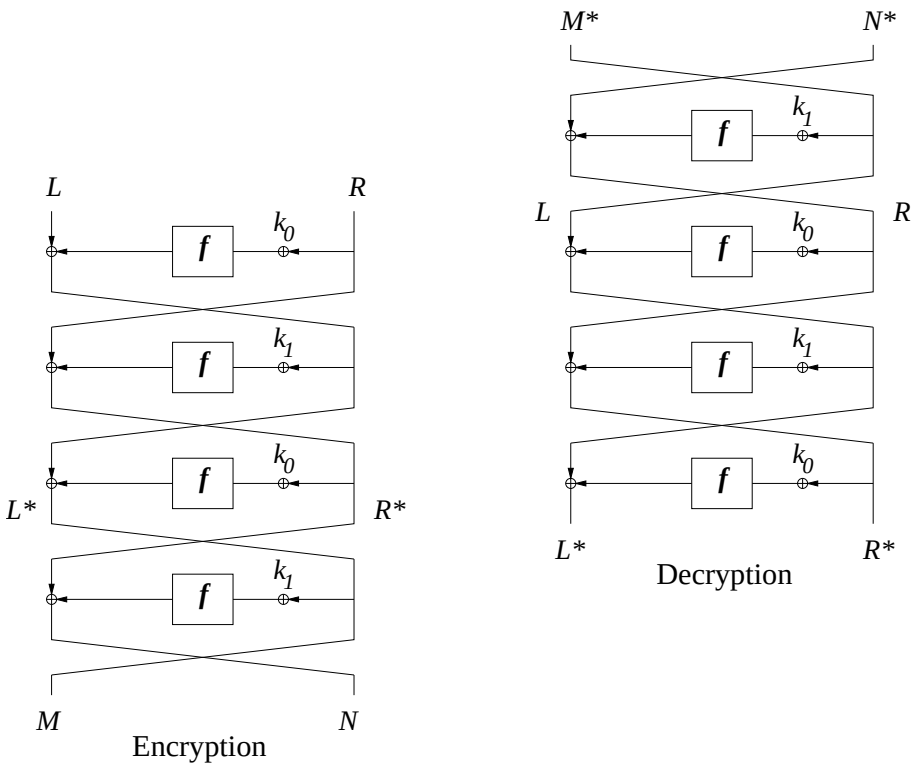


FIGURE 3. Sliding with a twist in the other direction.

We obtain an n-bit filtering condition (namely, $R = N^*$ and $N = R^*$), so detecting slid pairs is easy. The data complexity of the attack is $\Theta(2^{n/2})$ and its time complexity is $\Theta(n \cdot 2^{n/2})$, while its chosen plaintext/ciphertext variant has data complexity $\Theta(2^{n/4})$ and time complexity $\Theta(n \cdot 2^{n/4})$.

IV.4.5. How to Deal with Stronger Functions?

In [26], methods to deal with functions F that need more than two pairs (x_1, y_1) and (x_2, y_2) with $F(x_1, k) = y_1$, $F(x_2, k) = y_2$ in order to be broken are presented. Typically, it is useful in order to attack p-round-self-similar-ciphers ($p > 1$), especially if $p \neq 2, 4$.

One approach is suggested that uses differential analysis. Suppose that the round function F has a non-trivial differential characteristic $\Delta X \to \Delta Y$ holding with probability p. Then if we find a slid pair $(P; P^*)$ with $F(P) = P^*$, the pair $(P \oplus \Delta X; P^* \oplus \Delta Y)$ will be a slid pair as well with probability p. This principle is exploited in a chosen plaintext attack, by generating a set of $3 \cdot 2^{n/2} p^{-1/2}$ chosen plaintexts such that for plaintext P in the chosen set the plaintexts $P \oplus \Delta X$ and $P \oplus \Delta Y$ are also in the

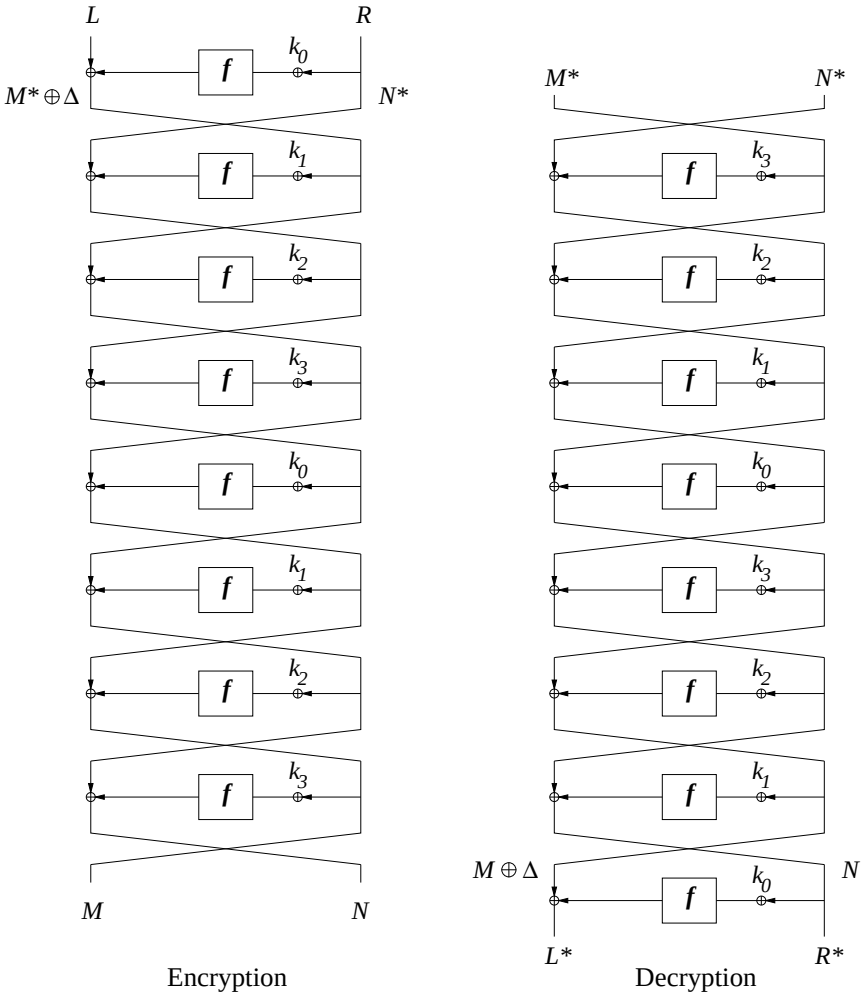


FIGURE 4. Combining the complementation slide and sliding with a twist techniques.

set. Then it may be expected to see one pair $(P; P^*)$ satisfying both the slide and the differential patterns.

However we claim that the number of plaintexts needed can be improved by choosing structures of four plaintexts of the form $(P, P \oplus \Delta X, P \oplus \Delta Y, P \oplus \Delta X \oplus \Delta Y)$. It is easy to verify that for any plaintext P in the chosen set, $P \oplus \Delta X$ and $P \oplus \Delta Y$ are also in the set. In such a set, with only $2^{n/2}p^{-1/2}$ plaintexts we may hope to find p^{-1} slid pairs and one pair satisfying both the slide and the differential patterns. Besides, if we are dealing with a set of *known* plaintexts, once a slid pair has been found, it is possible to develop strategies on the order we check the other pairs, based on the probability of the different characteristics $\Delta X \to \Delta Y$.

Another approach is the following. Suppose that we need N pairs (x_i, y_i) to recover key k. For each plaintext P it is suggested to get the encryption $E(P)$, the double encryption $E^2(P) = E(E(P))$, and so on, until we obtain $E^{2N}(P)$. Suppose now that $P^* = F(E^i(P))$ for some pair $(P; P^*)$ and some i; then we find $2N - i$ slid pairs "for free", by the relation $E^j(P^*) = F(E^{j+i}(P))$ for $j = 1, ..., 2N - i$. This phenomenon has been called *domino effect* in [38]. With $2^{n/2}N^{1/2}$ chosen texts, we have $2^n N$ potential slid pairs, and thus expect to find about N slid pairs (probably all in the same batch formed from a single coincidence of the form $P^* = F(E^i(P))$).

This principle is used in [58] to attack ikDES4, i.e. DES with a 4-round periodic key schedule: suppose we know $2^{n/2}$ plaintexts (in the case of DES (and ikDES4), $n = 64$). We may hope to find one slid pair among them. We have seen that when applying the attack to a 1-round-self-similar Feistel cipher, it was possible to distinguish a slid pair using only both plaintexts and their corresponding ciphertexts (i.e. knowing both pairs (P, C) and (P^*, C^*)). However the function F is now a 4-rounds DES, which makes it much more difficult to distinguish a slid pair. The trick is to compute multiple encryptions $C_1 = E(P)$, $C_2 = E^2(P)$,... for each plaintext P. If $(P; P^*)$ is a slid pair (i.e. a plaintext-ciphertext pair corresponding to a 4-round DES, keyed by the unknown (k_1, k_2, k_3, k_4)), then $(C_1; C_1^*)$, $(C_2; C_2^*)$,... also are. Thus a distinguisher on 4-round-DES can be used in order to check if the pairs $(P; P^*)$, $(C_1; C_1^*)$, $(C_2; C_2^*)$,... are indeed plaintext-ciphertext pairs of 4-round-DES. The distinguisher used in [58] relies on linear cryptanalysis [113].

The technique applied may be roughly summarized by:

(1) Consider $2^{n/2}$ plaintexts $\{P_0, ..., P_{2^{n/2}-1}\}$.
(2) For each P_i, compute $C_{i,k} := E^k(P_i)$, where $k = 1, 2, ..., N$. N must be big enough, such that a set of pairs $\{(P_i; P_j), (C_{i,1}; C_{j,1}), ..., (C_{i,N}; C_{j,N})\}$ which are plaintext-ciphertext pairs of F can be efficiently recognized.
(3) For each pair $(P_i; P_j)$, apply the distinguisher to the set of pairs $\{(P_i; P_j), (C_{i,1}; C_{j,1}), ..., (C_{i,N}; C_{j,N})\}$, until we indeed find a slid pair.
(4) Once a slid pair has been found, use a key recovery attack on F, in order to recover the key (the key recovery attack may be quite similar to the distinguisher).

In the case of the attack on ikDES4 described in [58], the distinguisher used is a linear approximation of four DES rounds, while the key recovery attack consists in guessing the first and last round keys, and using a linear approximation for the remaining two rounds. These techniques are described in detail in [113]. The distinguisher used needs about

4000 pairs, thus the slide attack needs about $2^{32} \cdot 4000 \approx 2^{44}$ chosen plaintext-ciphertext pairs for the whole cipher.

The main limitation of the approach presented is that we have to consider about 2^n pairs of plaintexts, and for each of them, to apply a chosen plaintext attack. Thus this last attack must be kept light enough in terms of computation and data complexity[2], otherwise the whole slide attack will be more costly than exhaustive key search (which requires about 2^n_k encryptions). Assuming this, this technique is rather powerful to attack algorithms with periodic key schedules.

IV.5. Differential Related Key Attacks

The basic idea of differential cryptanalysis is to perform the encryption of pairs of plaintexts, say $(P; P \oplus \Delta)$, with Δ a chosen difference. After a large number of rounds (typically, $r-2$ or $r-3$, with r the number of rounds of the whole cipher), we hope to observe another given difference, say Δ', with a "high" probability.

The idea of differential related key attack is not very different: simply we allow the two plaintexts P and $P \oplus \Delta$ to be encrypted with different keys, say K and $K \oplus \Delta K$:

$$C = F(P, K), \qquad C' = F(P \oplus \Delta, K \oplus \Delta K). \tag{88}$$

ΔK must be chosen carefully, so as to obtain the desired difference in the subkeys.

This allows much more freedom for the attacker, as she can act on the key difference too. This way she can find attacks that are not possible otherwise. Note however that the basic hypothesis are not the same as those of differential cryptanalysis, as we assume we get an access to encryption under two (or more) related keys.

Below we present the attack on 3-WAY given in [90], as it is a simple and illustrative example of this kind of attacks.

IV.5.1. Attack on 3-WAY

3-WAY is an 11-round cipher on 96-bit blocks. It has a classical substitution-permutation network structure. Its round function can (roughly) be described as

$$F(x) = \theta(\gamma(x)) \oplus K \oplus C_i, \tag{89}$$

where:

- γ is a fixed nonlinear layer built out of 32 parallel 3-bit permutation S-boxes,

[2] which is not easy, as we need excellent success rates in order to be able to distinguish one good slid pair among the 2^n other ones.

- θ is a fixed linear function,
- K is the 96-bit master key,
- C_i is a fixed round-dependent public constant (which prevents using the slide attack).

A 1-round characteristic (for classical differential cryptanalysis) is the following:

- It is easy to find a differential characteristic for one S-box with probability 1/4, so we can construct a characteristic $\Delta x \rightarrow \Delta y$ with probability 1/4 for the non-linear layer γ by using only one active S-box.
- By linearity $\Delta y \rightarrow \theta(\Delta y) =: \Delta z$ is a characteristic of probability 1 for the linear layer θ.
- The key addition does not change the difference.

We thus have a 1-round characteristic $\Delta x \rightarrow \Delta z$ with probability 1/4. But this characteristic is not iterative: the difference Δz probably affects many S-boxes in the second round.

If now we place ourselves in the context of differential related key attacks, we can pick $\Delta K = \Delta x \oplus \Delta z$. The 1-round characteristic becomes $\Delta x \rightarrow \Delta x$. It is iterative, with probability 1/4.

We can derive a 9-round characteristic with probability 2^{-18}, and apply a 2R-analysis to the last two rounds. This breaks 3-WAY with one related key query and about 2^{22} chosen plaintexts.

To avoid this type of attack, it is suggested in [89, 90] to use a non-linear (or even one-way, but it is probably an overkill) key schedule, which makes difficult to produce controlled changes in the round keys.

IV.6. Using Related Key Attacks Against Multiple Encryption

Due to the fact that their different parts are keyed independently, multiple encryption modes, such as DESX [91] (or more generally, the Even-Mansour construction [52]) or Triple-DES are particularly vulnerable to related key attacks, or even key schedule attacks in a classical context. Meet-in-the-middle attacks [118, 165, 166] are well-known. However several other attacks have been published.

In [26] A. Biryukov and D. Wagner present an attack on the Even-Mansour extension of a block cipher. It is the best known attack on this type of construction. Let E^* denote a randomly chosen permutation of block size n. Then the Even-Mansour construction around E^* is keyed by $K := (k_a, k_b)$, where $k_a, k_b \in \mathbb{Z}_2^n$, and defined as $EX_{(k_a,k_b)}(P) =$

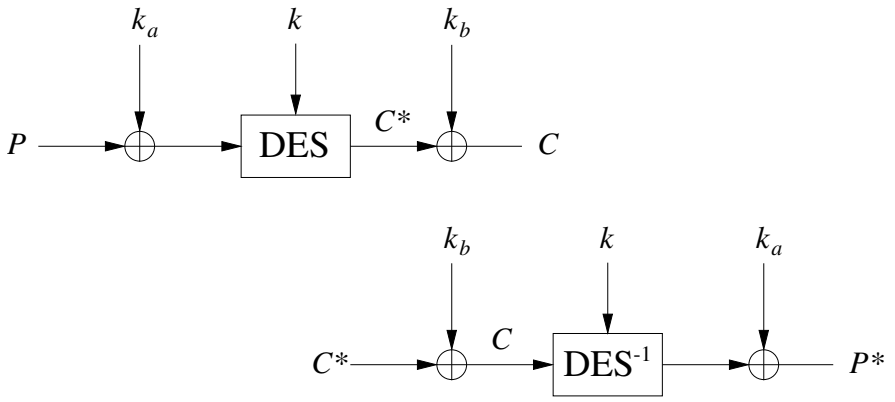


FIGURE 5. Sliding with a Twist on DESX.

$k_b \oplus E^*(P \oplus k_a)$. If the random permutation is replaced by a DES, we obtain a construction called DESX:

$$\mathrm{DESX}_{(k_a,k,k_b)}(P) = k_b \oplus \mathrm{DES}_k(P \oplus k_a). \tag{90}$$

We can apply a sliding with a twist to two copies of DESX (see Figure 5). The slid equations are:

$$\begin{cases} P^* = k_a \oplus \mathrm{DES}_k^{-1}(C) \\ P = k_a \oplus \mathrm{DES}_k^{-1}(C^*) \end{cases} \tag{91}$$

These equations imply

$$P^* \oplus \mathrm{DES}_k^{-1}(C) = P \oplus \mathrm{DES}_k^{-1}(C^*). \tag{92}$$

Provided a sufficient number of known pairs (P_i, C_i) is available, an attack begins with guessing the value of k. For each k, pairs $((P, C); (P^*, C^*))$ satisfying (92) are identified using a lookup table. Given such a pair, it is easy to compute the value of k_a and k_b. Once a candidate for (k_a, k, k_b) has been found, it is checked using a few other plaintext-ciphertext pairs. This attack requires about $2^{\frac{n+1}{2}}$ known plaintexts; its average complexity is $2^{\frac{n+1}{2}+\frac{n_k}{2}}$ trial DES encryptions.

This attack works provided one of the two additions of whitening keys (k_a and k_b) is done using $\oplus$. But there are *related key* attacks against a much wider range of algorithms; namely such an attack can be mounted for any multiple encryption with all subkeys being different (even if the algorithm used at each stage is different). Consider for example a 3-key triple encryption (such as Triple-DES) associated with algorithm E: $3E_{(k_1,k_2,k_3)} := E_{k_1} \cdot E_{k_2} \cdot E_{k_3}$. Assume the attacker knows a pair (P, C) encrypted under an unknown key $K = (k_1, k_2, k_3)$, and is allowed to make a decryption query to the same cipher keyed by $K^* = (k_1^*, k_2, k_3) =$

$(k_1 \oplus \Delta, k_2, k_3)$, where Δ is an arbitrary known constant. Then she can obtain $P^* = (3E_{K^*})^{-1}(C)$. The two pairs $P \overset{K}{\to} C$ and $P^* \overset{K^*}{\to} C$ satisfy

$$P^* = E^{-1}_{k_1 \oplus \Delta}(E_{k_1}(P)). \tag{93}$$

This equation permits exhaustive search on k_1. Once k_1 has been found, k_2 and k_3 are found using a meet-in-the-middle attack. If only *encryption* related key queries are available, the attack still works, but its data complexity increases: the attacker first asks for $2^{n/2}$ pairs (P, C) encrypted under K, and $2^{n/2}$ pairs (P^*, C^*) encrypted under K^*. Then she searches for a pair $(P; P^*)$ of plaintexts having the same ciphertext C. P and P^* satisfy (93).

This attack has been applied to Triple-DES in [89]; it was presented as a differential related key attack. But properly speaking it is not differential. As a matter of fact, the attacker could a priori use any known permutation $\Pi : \mathbb{Z}_2^{n_{k1}} \to \mathbb{Z}_2^{n_{k1}}$ (where n_{k1} is the key size of the first cipher) to compute k_1^* as a function of k_1. The only requirement on Π (besides being usable in a practical attack context and easy to compute) is that for a given $(P; P^*)$ the number of solutions of (93) is not too big. Extremely bad choices of Π in this context are when E_{k_1} is simply a key addition $P \mapsto P \boxplus k_1$, for $\boxplus$ some group operation, and Π is $x \mapsto x \boxplus \Delta$ for some constant Δ; in this case all choices of k_1 satisfy (93).

The attack has also been applied to DESX+ (a variant of DESX where the pre- and post- whitening $\oplus$ operations are replaced by additions mod 2^{64}) by R.C.-W. Phan [37]. The permutation Π he used is also $x \mapsto x \oplus \Delta$. R.C.-W. Phan claims it does not work on DESX, which should imply that DESX is in some way stronger than DESX+. It is true for $\Pi : x \mapsto x \oplus \Delta$, because of the restriction we gave on the choice of Π. However the easy choice $\Pi : x \mapsto x + \Delta \mod 2^{64}$ makes the attack work. Thus the statement about the respective strength of DESX and DESX+ is not true.

In [37] related key attacks on 2-key Triple-DES and DES-EXE[3] are also presented, but they are erroneous; it is claimed that most of the candidates for subkey k_1 can be discarded during the attack, but in fact it is not the case: most of them survive the test. This makes these two attacks impractical. The attack against 2-key Triple-DES could be repaired, but it makes its complexity comparable to the one of the meet-in-the-middle attack [118, 165] (which is not a related key attack). Finally, a related key sliding with a twist attack on triple encryption (using keys $K = (k_1, k_2, k_3)$ and $K^* = (k_1, k_3, k_2)$) is proposed.

[3]DES-EXE is a three-layered encryption defined by $\text{DES-EXE}_{(k_1,k_2,k_3)}(P) := \text{DES}_{k_3}(k_2 \oplus \text{DES}_{k_1}(P))$.

We think that all these attacks illustrate the need for some kind of key scheduling in multiple encryption modes, at least when related key attacks are a concern. Passing the key through a hash function before using it would be ideal. Another very good solution has been suggested by I. Damgärd and L. Knudsen in [48] for the case of triple encryption. They suggest to use the cipher itself in the construction of the key schedule. More precisely, if (k_1, k_2) is the master key, the three keys x_1, x_2, x_3 used for encryption are computed as

$$\begin{aligned} x_1 &= E_{k_1}(E_{k_2}(IV_1)), \\ x_2 &= E_{k_1}(E_{k_2}(IV_2)), \\ x_3 &= E_{k_1}(E_{k_2}(IV_3)), \end{aligned} \tag{94}$$

where E denotes encryption under the block cipher considered and IV_i are three different initial values.

The problem of building multiple encryption modes with resistance against related-key attacks as a primary design criteria has also been dealt with by S. Lucks in [111].

IV.7. Link Between the Attacks

Slide attacks and differential cryptanalysis have in common that the attacker tries to find pairs of plaintext that have a certain relationship, so that there is some link between their encryption process.

The only difference between differential cryptanalysis and differential related key attacks is that in the second one the pairs of plaintexts are encrypted using two different keys (that have a particular relationship). On the one side, it makes the attack more powerful, in the sense that it permits finding better characteristics on a larger number of rounds, and thus also attacking a broader range of algorithms. On the other side, it must be noted that it implies an attack scenario which is more difficult to carry out.

Exactly in the same way, the slide attack and the related key slide attack are distinguished by the fact that the first one deals with a single key, while the second one attacks a pair of keys. And once again, it allows attacking a wider range of algorithms, as the key schedule does not need to be periodic any longer in order to be vulnerable, but only to have a weaker property (see section IV.3 for more details). Thus we can say that Biham's related key attack is to the slide attack just as the differential related key attack is to differential cryptanalysis.

We could wonder whether other known attacks can be turned into a related key version. First, it must be noted that it is only applicable for attacks that deal with pairs (or bigger groups) of plaintexts. Thus linear cryptanalysis, for example, does not fit. On the contrary, there is a related key counterpart to the differential-linear cryptanalysis. Moreover

a recent paper of J. Kim et al. [92] deals with a related key rectangle attack. Yet another example of this type of counterpart has been given by J. Nakahara et al. in [123]: they presented a related key version of the square attack. However such an attack is far from realistic, as it would impose to have plaintext-ciphertext pairs encrypted under no less than 2^{16} related keys (in their example on IDEA). But one can argue it has a theoretical interest.

We have seen in section IV.3 that the related key slide attack principle can be used to reduce the complexity of an exhaustive key search, as noted by L. Knudsen [94]. The question we will now deal with is: is it possible to reduce the complexity of exhaustive search using other related key principles?

First, we deal with the differential related key attack. For a given block cipher, suppose we can find differences ΔP, ΔC and ΔK such that for randomly chosen plaintext P and key K, $P \xrightarrow{K} C$ implies $P \oplus \Delta P \xrightarrow{K \oplus \Delta K} C \oplus \Delta C$ with a "high" probability p. An improved key search algorithm would be:

- Let (P, C) and $(P \oplus \Delta P, C^*)$ be two plaintext-ciphertext pairs obtained under the unknown key K.
- For each trial key K', compute $P \xrightarrow{K'} C'$:
  - If $C' = C$, we conclude that $K = K'$
  - If $C' = C^* \oplus \Delta C$, we conclude that with a probability p, $K = K' \oplus \Delta K$. Assuming $p > 2^{-n_k}$ (where n_k is the number of key bits), $K' \oplus \Delta K$ can be considered as a favorite key candidate. We immediately check whether it is the right key by computing the encryption of P under $K' \oplus \Delta K$. If it is not the right one, we insert it in a (sorted) list of already tried keys.

At first glance, this algorithm may look fine, but let us consider the complexity analysis:

- If $P \xrightarrow{K \oplus \Delta K} C^* \oplus \Delta C$, the key search will be reduced by about half of the encryptions, provided we try $K \oplus \Delta K$ before K. Knowing that $P \oplus \Delta P \xrightarrow{K} C^*$, it will happen with probability p. Thus the average gain is $p \cdot 2^{n_k - 2}$ encryptions.
- The drawback is that for each trial key it is necessary to do one comparison (namely, check if $C' = C^* \oplus \Delta C$ during the first half of the key search; then for each key check if it has not already been tried[4]). The time needed to build the sorted list of already tried keys may be considered as negligible (as it is small).

[4]We assume here that the key search begins with exploring one half of the key space, say Ψ (with $|\Psi| = 2^{n_k - 1}$), such that $\forall K' \in \Psi : K' \oplus \Delta K \notin \Psi$.

The average gain of our algorithm is $p \cdot 2^{n_k-2}$ encryptions, while the drawback is 2^{n_k} comparisons. As a consequence, for our algorithm to make sense the ratio between the time for one comparison and the one for one encryption must not be more than $p/4$. But as p is very small most of the time, it will usually not be the case.

The conclusion is that it is the fundamentally probabilistic nature of the differential related key attack that makes it (almost) impossible to use in order to improve exhaustive key search. The square attack is not probabilistic. One can thus wonder if its related key counterpart could not be used with more success for reducing complexity of exhaustive search. However, because of the large number of related keys implied, this technique would only allow sparing a very small part of the key space ($1/2^{16}$ in the case of IDEA studied by J. Nakahara et al. [123], if we assume that the key schedule makes it possible to apply such an attack). Therefore, the extra work needed to manage this variant of exhaustive search will clearly compensate the key trials we spare.

IV.8. Conclusion

We have seen that the condition for a block cipher to be vulnerable to Biham's related key attack is that the sliding of all the subkeys derived from a given key by one round (or eventually more) gives rise to a sequence of subkeys that can be derived from another key. The slide attack has a strong relationship with Biham's attack: it may be viewed as a kind of "auto-related key attack", as the algorithms vulnerable to it are those for which the sliding of the subkeys gives rise to the same sequence of subkeys!

We also presented the differential related key attack, that is the related key counterpart of differential cryptanalysis, just as Biham's attack is the related key counterpart of the slide attack. We discussed under which conditions a related key version could be found for other attacks.

Finally, we discussed whether these various related key attacks can be used in reducing the complexity of exhaustive key search. The conclusion is that the probabilistic nature of most of these attacks makes them very difficult to apply in this context; other attacks that are not probabilistic but deal with a large number of related keys (such as the square related key attack) cannot be applied either.

Although they were often inspired by glaring weaknesses in old algorithms, these attacks are not necessarily impossible to apply to more recent key schedules. As an illustration, several papers present attacks on recent algorithms [55, 78, 92]. Also, we have seen that they can often apply to multiple encryption modes with independent subkeys, which are widely in use today. Maybe some kind of key schedule could increase security of these encryption modes.

Nor are these attacks purely theoretical: they must be taken into considerations when designing protocols, especially key derivation, key update, or key exchange. As an illustration of this, R.C.-W. Phan and H. Handschuh recently showed that the key management used in the IBM 4758 cryptoprocessor is potentially vulnerable to related key attacks [39].

Part 2

Side-Channel and Implementation Aspects

CHAPTER V

Differential Fault Attacks - Application to SP-Networks

Abstract. In this chapter we deal with the application of a particular class of side-channel attacks to block ciphers: ***fault attacks***. After discussion of the context of these attacks, we survey the most relevant works in the field.
The core of the chapter focuses on our personal contribution, originally published in [140]; we devised a differential fault attack technique working against any Substitution-Permutation Network. It requires very few faulty ciphertexts, and works under several realistic fault models. We applied our attack to KHAZAD and the AES; we were able to break the AES-128 with only 2 faulty ciphertexts (assuming the fault occurs between the antepenultimate and the penultimate `MixColumns`); this is better than the previously known fault attacks against AES [27, 36, 51, 63].
Finally, countermeasures against fault attacks on block ciphers are discussed.

V.1. Introduction

The interest in side-channel attacks appeared in the late nineties, with the seminal papers of P. Kocher [100] and D. Boneh et al. [28]. The underlying idea is to look at the way cryptographic algorithms are implemented, rather than at the algorithm itself. The literature makes a distinction between two types of attacks:

- ***Passive Attacks*** observe the behavior of a computing device through various channels, and try to use the measurements made in order to recover information about a secret key. Typical types of channels used may be time of computation (timing attacks [100]), power consumption (single and differential power analysis [101]), electromagnetic radiations (electromagnetic analysis [144, 60]).
- ***Active Attacks***, more frequently called ***Fault Attacks***, try to *modify* the functioning of the computing device (typically a smart card) in order to retrieve (part of) the secret key. More precisely, the attacker induces a fault during cryptographic computations; the faulty results are used for key recovery.

Side-channel attacks have become a much bigger threat for the cryptography providers (mainly the smart card industry) than classical cryptanalytic attacks. A lot of resources are spent in searching for countermeasures against them. In this chapter we deal with fault attacks only. We first present a famous example of application in asymmetric cryptography in section V.2. Then we focus on fault attacks against block ciphers. The first example of such an attack is due to E. Biham and A. Shamir [20]. Among other things, they show how to attack DES; a more generic attack under stronger hypothesis is also presented. After a description of the different possible settings for a fault attack in section V.3, we describe Biham-Shamir's results, and some more recent discussions about them, in section V.4. Sections V.5, V.6 and V.7 are the core of this chapter. They describe an attack devised by the author and applicable to any substitution-permutation network. We examine in detail the particular case of the AES. These results have been originally published in [140]. Finally, section V.8 discuss the problem of countermeasures. Section V.9 is the conclusion.

V.2. An Introductory Example: Fault Attack Against a CRT Implementation of RSA

The attack we present in this section is bound to public-key cryptography, and is thus slightly off-topic. We chose to present it anyway because it is a nice illustration of the power of fault attacks, and historically one of the first of them (see [28]). Besides it is elegant and easy to understand.

Let $N = p \cdot q$ be the product of two secret big primes p and q. An RSA signature [150] of a message m is computed as $S = m^d \mod N$ where d is a secret exponent; the verification is performed by computing $S^e \mod N$ where e is the public verification exponent, and checking whether $S^e \equiv m \pmod N$. However the trivial implementation of signature (direct exponentiation $\mod N$ using square-and-multiply) is not the fastest one: a speed factor from 4 to 8 can be gained by using the *Chinese Remainder Theorem*. The signer first computes $S_p = m^d \mod p$ and $S_q = m^d \mod q$. Application of the Chinese Remainder Theorem tells us that we can compute a, b such that:

$$\begin{cases} a \equiv 1 \pmod p \\ a \equiv 0 \pmod q \end{cases} \qquad \begin{cases} b \equiv 0 \pmod p \\ b \equiv 1 \pmod q \end{cases}$$

Note that knowing p and q, a and b can be precomputed. Then it is easy to show that

$$S \equiv a \cdot S_p + b \cdot S_q \pmod N.$$

Therefore, instead of an exponentiation modulo N, the signer needs to do two exponentiations with moduli p and q, and a modular addition.

The fact that p and q have about twice less bits than N, makes the whole algorithm faster than a trivial implementation.

Assume now a fault occurs during the computation of $m^d \mod q$. Then a wrong $\widetilde{S}_q$ is obtained, and with overwhelming probability $\widetilde{S}_q \not\equiv S_q \pmod q$. Let $\widetilde{S} = a \cdot S_p + b \cdot \widetilde{S}_q$ be the wrong signature obtained. Then

$$\begin{aligned} & \widetilde{S} \mod p = S_p \mod p = S \mod p \\ \text{and} \quad & \widetilde{S} \mod q = \widetilde{S}_q \mod q \neq S \mod q. \end{aligned}$$

So we have $\widetilde{S}^e \equiv S^e \equiv m \pmod p$ but $\widetilde{S}^e \not\equiv S^e \equiv m \pmod q$. This implies that $\gcd(N, m - \widetilde{S}^e) = p$. As all variables of the left-hand side are known to the attacker, she can compute p.

Remark that this attack is very powerful, as it only requires one fault to occur at any time during one (and only one) of the two exponentiations (without more constraint), and the right signature is not even mandatory, only the wrong one is needed. It means that someone receiving a certificate with a false authority's signature, could retrieve the private key of the system! However several countermeasures exist against this attack[1] (see for example [81, 156, 170]). In [28], similar attacks against other public key cryptosystems, such as Fiat-Shamir [53] and Schnorr [155] identification schemes, are presented. Attacks on DSA and ECDSA also exist (see for example the survey of C. Giraud and H. Thiebeauld [64]).

V.3. Fault Model

The feasibility of a fault attack (or at least its efficiency) depends on the exact capabilities of the attacker and the type of faults she can induce. This constitutes the ***fault model***. It should specify:

- The precision an attacker can reach in choosing the time and location on which the fault occurs during the execution of a cryptographic algorithm. Choosing the time at which the fault occurs can often be done by using side-channel information (such as power consumption) to monitor the progress of the algorithm. Choosing the location is more difficult; this ability depends on the fault induction technique used.
- The length of the data affected by a fault: one only bit, or one word (typically one byte, as current smart card architectures are still 8-bit).

[1] Note that inducing a fault during the recombination of S_p and S_q makes most of them inefficient.

- Whether the fault is *transient* (disturbing the smart card during its processing, therefore affecting a single execution of the algorithm) or *permanent* (resulting from a permanent damage of the smart card, such as cutting a wire or destroying a memory cell).
- The type of the fault:
  - Flip one bit.
  - Flip one bit, but only in one direction (e.g. from 1 to 0).
  - Byte changed to a random (unknown) value.
  - Bit(s) permanently stuck at 0 (or 1).
  - Skip of one or more instructions of the program code.
  - Repeat one or more instructions of the program code.

There are many different ways of inducing faults:

- Cutting a wire in a hardware circuit results in permanently fixing one bit to 0.
- Changing the power supply voltage or the frequency of the external clock might cause a transient fault that changes an entire byte to a random value; in this case the attacker has little control on the timing of the fault, and no control on its location, but on the other hand this attack needs very few material to be performed. Note that if the probability p for the physical stimuli to induce a fault during the encryption is small, then the probability that two different faults occur during the same encryption is $\sim p^2$, which can be assumed to be negligible. This is important as most of the differential fault attacks (and more particularly the one presented in section V.5) assume that a faulty ciphertext results from one unique fault.
- Applying radiation to one of the transistors constituting a memory cell could cause the corresponding bit to switch, hence a transient fault [158].

The interested reader can find more details on the way faults are physically induced in [3, 6, 64] for example. In the next section we will see how various fault models can be exploited in fault attacks against block ciphers, following the work of E. Biham and A. Shamir. Then we present our attack against substitution-permutation networks, that can work under several of these models.

V.4. Fault Attacks on Block Ciphers

E. Biham and A. Shamir's paper [20] is the first work dealing with fault attacks on block ciphers. They present several different attacks, under various fault models.

V.4.1. Differential Fault Attack on DES Using Flips on a Single Unknown Bit

In differential fault attacks, a right ciphertext (resulting from an encryption without induction of a fault) is considered together with a faulty ciphertext corresponding to the same plaintext. The comparison of both encryptions allows the retrieval of key material using techniques related to those of differential cryptanalysis [19] (hence the name).

E. Biham and A. Shamir presented a differential fault attack on DES (see description in Appendix B.2; note that in the remaining of this section the final permutation FP is neglected). The faulty ciphertexts used in the attack are assumed to result from one bit being flipped in the register keeping the right half of the data in one of the 16 rounds. The index of the round affected and the precise location of the bit are unknown; there are thus $16 \cdot 32 = 512$ possibilities. However by observing the difference between the right and the faulty ciphertext, the attacker can deduce whether the round affected is the 16^{th}, the 15^{th}, the 14^{th}, or one of the rounds before. If the fault occurred before the 11^{th} round, it is not exploitable by Biham-Shamir's attack[2].

Assume the round affected is the last one. The output difference of the last round function is equal to the difference in the left part of the ciphertext; and the data entering this round function is equal to the right part of the ciphertext. The attack is simple: we guess the key bits entering the S-box (or the two S-boxes) affected by the fault, and check whether the guessed value agrees with the expected difference at the output of this (these) S-box(-es). On average, about 4 possible 6-bit values of the key remain for each of the S-boxes concerned.

If the round affected is the last but one, the attack is very similar. If it is the last but two (or earlier), a counting method [19] must be used, where for each S-box a counter is associated to each 6-bit candidate, and the right value is expected to be counted more than the others.

Assuming faults occur randomly in all rounds, E. Biham and A. Shamir found that between 50 and 200 ciphertexts were needed to retrieve the key. If the attacker can choose the exact position of the fault, this number can be reduced to about 3 ciphertexts.

If the induced faults affect several bits simultaneously, the attack still works; in fact it works even better because more S-boxes are affected (but on the other hand it could be more difficult to identify the round during which the fault occurred). In [64] C. Giraud and H. Thiebeauld claimed to have succeeded in retrieving the key using only 2 ciphertexts; the faults were induced by a light attack [158] using a camera flash.

[2]However, an attack allowing to exploit faults occurring on early rounds of DES has been recently published [71].

V.4.2. A Generic Attack if Memory Bits can be Stuck at 0

Consider a memory register containing a key K of length n. We assume the attacker is able to apply some kind of stimuli to the smart card, whose effect is that each bit at 1 in this register could flip to 0 with a small probability p_1. As p_1 is assumed to be small, the probability for two bits to flip simultaneously is considered as negligible. Then a very generic attack applies:

(1) In a first step, the attacker asks for the encryption of some plaintext P, then applies a "stick-at-0" stimuli, asks for the encryption of P again, etc. This way she obtains a sequence of ciphertexts $C_0, C_1, ..., C_r$ (if the same ciphertext has been obtained several times due to the stimuli having no effect, we keep only one copy of it). C_0 is the ciphertext corresponding to key K, and if the process described above has been conducted a sufficient number of times, C_r corresponds to the encryption of P under the all-0 key, and r is the Hamming weight of the initial key.
(2) Let K_i be the key used to obtain C_i ($K_0 = K$, $K_r = 0$). Then K_i and K_{i+1} differ only by one bit. Knowing K_{i+1}, it is possible to retrieve K_i by doing trial encryptions under keys $K_{i+1} \vee \delta_j (j = 1, ..., n)$, where δ_j has all its bits equal to 0 except the j^{th}. As K_r is known, K_0 can be retrieved by iterating this process.

The overall complexity of the attack is $\Theta(r^2)$. Trial encryptions in the second stage require prior knowledge of the encryption algorithm. However this is not mandatory provided we can ask the smart card to load a chosen key and perform encryption using this key. Nor is it if the location of the bit stuck at 0 can be chosen.

In [129] P. Paillier examined the following modified model (probably more realistic): when the card is exposed to the physical constraint mentioned above, in addition to the $1 \rightarrow 0$ transitions, it is also possible for a 0-bit to flip at 1 with a small probability $p_0 < p_1$. Under this hypothesis he shows that the sequence $(K_i)_i$ does not converge to 0: as the Hamming weight decreases, the number of candidates for a $0 \rightarrow 1$ transition grows, while the number of candidates for $1 \rightarrow 0$ decreases. If p_0 is not small enough compared to p_1, an equilibrium is reached at some Hamming weight; this can make the complexity of the attack much bigger.

Furthermore, he proposes several countermeasures against this attack[3]:

[3]Note that these countermeasures are not necessarily efficient if the fault location can be chosen.

- Checking whether the Hamming weight of the encryption key is not too far from $n/2$. If yes, encrypt under the all-0 key; or return an error.
- Using keys with parity-checking bits (as in the case of DES), and return an error if the key does not pass the check.
- Authenticating the key using a hash function.

Finally, he notes that the attack does not work against probabilistic algorithms (such as ElGamal encryption, DSA signature, or several other public-key algorithms).

Regarding the particular case where the location of the bit stuck at 0 can be chosen by the attacker (which is a very restrictive hypothesis), W.W. Fung and J.W. Gray propose in [57] a countermeasure where each key bit in the memory register is repeated m times and all these bits are mixed. The key is read through an hidden bit permutation layer; all m copies of a given bit have to agree, else an error is given. They show that applying this countermeasure the number of faults required for the attack to succeed is $\Theta(n^m)$.

An attack similar to this one has been put into practice in 1994, aiming at a DES key; see section 3.3 of [6] for more details.

V.4.3. Reconstructing Unknown Ciphers

Under the same fault model as in section V.4.1, E. Biham and A. Shamir showed that it is possible to reconstruct an unknown DES-like cipher. First the number of S-boxes and their size are identified, as well as their input and output bits (depending on the bit permutation layers). Then the content of the S-boxes is identified, up to two constants (i.e. a table $T(x)$ is retrieved, related to the real S-box $S(x)$ by $T(x) = S(x \oplus k) \oplus u$ where u is a constant, and k a subkey). Finally, the exact content of the S-boxes can be found by considering encryption under several different keys. If the key scheduling algorithm is as simple as the one of DES, it can be retrieved too. Running their algorithm choosing DES as the unknown cipher, E. Biham and A. Shamir found that about 10000 faulty ciphertexts were necessary to reconstruct the whole cipher.

In [4] R. Anderson and M. Kuhn note that retrieving the structure of an algorithm such as RC5 [148] is even easier. They suggest that if a cipher is to be kept secret (which is *per se* a questionable practice...), an algorithm with large S-boxes kept in EEPROM (that is separate from the program memory) should be preferred.

V.4.4. Other Attacks

Other attacks are also possible, like:

- By cutting a wire on a hardware implementation, one can definitively fix a bit to 0. It is then not difficult to use this defectiveness to retrieve key bits, even if correctly encrypted ciphertexts are not available.
- A fault affecting a loop counter so that only 2 or 3 rounds of a 16-round DES are executed would trivially allow the cipher to be broken.

V.5. Our Attack on Substitution-Permutation Networks

In this section we present a fault attack working against any block cipher with a Substitution-Permutation structure. The attacker uses faults occurring on words, which are changed to a random value (although we will show that the attack would work against bit flip faults as well). The block cipher structure considered is described in more details in section V.5.1. The fault model is made explicit in section V.5.2. The attack itself is presented in sections V.5.3 to V.5.5. Finally, section V.5.6 deals with badly timed faults and section V.5.7 analyzes the attack under other fault models.

V.5.1. Description of the SP-Network Attacked

As we have seen in Chapter I, the round function of a substitution-permutation network has the form $\sigma[k^r] \circ \theta_r \circ \gamma_r$ (r is the round number). In our attack:

- The non-linear layer γ_r is assumed to be made out of n 8×8 S-boxes (not necessarily identical).
- $k^r = (k_1^r, k_2^r, ..., k_n^r)$ denotes the r^{th} round key.
- The key addition $\sigma[k]$ is

  $$\sigma[k](a) = b \Leftrightarrow b_j = a_j \oplus k_j \qquad (j = 1, 2, ..., n).$$

  The group operation $\oplus$ is assumed to be exclusive or, but the attack will work against other group operations as well.
- The diffusion layer θ_r is linear with respect to $\oplus$.

We denote the block size by $8n$. Note that the fact that the S-boxes are 8×8 is absolutely not mandatory for our attack; we restricted it to this parameter as it is common to choose such a size, well fitted to implementation considerations. 4×4 and 2×2 S-boxes can be viewed as 8×8 S-boxes as well, by considering groups of 2 (resp. 4) of them.

The last round of the cipher has the special form $\sigma[k^R] \circ \gamma_R$ and the first round is preceded by a key addition layer. Thus the whole cipher can be described as

$$\sigma[k^R] \circ \gamma_R \circ (\bigcirc_{r=1}^{R-1} \sigma[k^r] \circ \theta_r \circ \gamma_r) \circ \sigma[k^0].$$

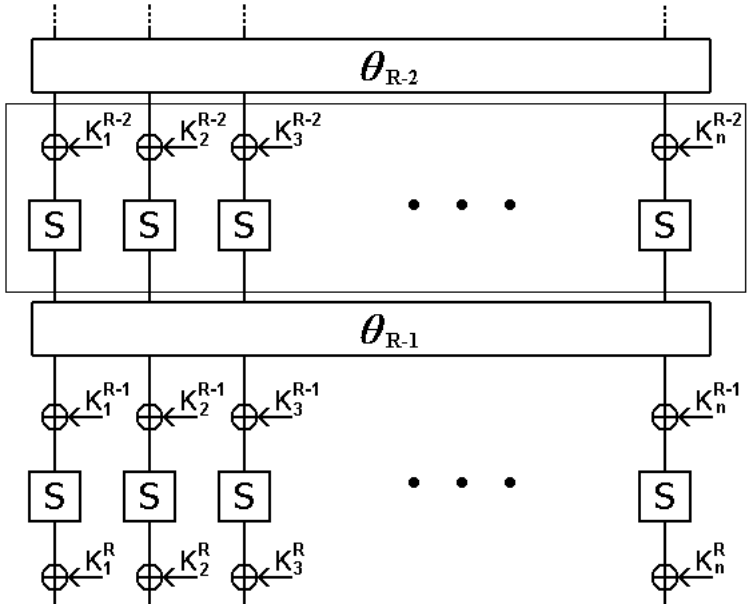


FIGURE 1. Last 2.5 rounds of a Substitution-Permutation network.

Remark that the γ_r and θ_r layers need not be identical for all rounds. Only the last two rounds[4] are important for our attack. They are depicted in Figure 1.

V.5.2. Fault Model

We are dealing with faults occurring on one word, as they are usually easier to induce than faults on bits. Concretely, we restrict ourselves to considering bytes, because it is the word size of current smart cards, but the attack would work for other sizes. It could even be adapted to the case of faults on bits. We also assume that a faulty ciphertext results from the occurrence of one unique fault (see section V.3).

We consider *random faults*, in the sense that the faulty value is assumed to be random and uniformly distributed. Moreover we assume that the fault occurred sometime between the before-last layer θ_{R-2} and the last layer θ_{R-1} (i.e. somewhere inside the frame of Figure 1). Under this condition, the exact stage of the computation at which the fault occurred is indifferent, and cannot be guessed by observing the ciphertexts either. In section V.5.6 we will deal with the problem of discarding ciphertexts resulting from faults occurring at another time. In the remaining of this chapter, by $(C; C^*)$ we always denote a right ciphertext C and its corresponding faulty ciphertext C^*. Also, unless otherwise stated,

[4]Or sometimes three in extensions of the attack. See section V.7.

indices will refer to byte positions (for example, C_1 denotes the leftmost byte of C).

V.5.3. Basic Attack

Consider thus 1-byte differences at the input of the linear layer θ_{R-1}. We count $255n$ possible such differences (n different possible locations, and 255 different possible values). Because of the linearity, the number of corresponding possible differences at its output is also $255n$; but while the input difference affected one byte only, the output difference affects several ones, because of the diffusion (if the diffusion layer has a maximal branch number, all bytes are affected; otherwise only a few of them may be affected). Note that the key addition $\sigma[k^{R-1}]$ does not change the set of possible differences.

These considerations lead to a first sketch of attack. For simplicity we assume the θ_{R-1} layer achieves optimal diffusion.

(1) Compute the $255n$ possible differences at the output of θ_{R-1}, i.e. the $255n$ values $\theta_{R-1}(x)$, where x has a byte Hamming weight of 1. Store them in a list $\mathcal{D}$.
(2) Consider a plaintext P, C its corresponding ciphertext, and C^* the faulty ciphertext.
(3) For all possible values of round key k^R, compute the difference

$$\gamma_R^{-1} \circ \sigma[k^R](C) \oplus \gamma_R^{-1} \circ \sigma[k^R](C^*).$$

Check whether it is in $\mathcal{D}$. If yes, add the round key to the list $\mathcal{L}$ of possible candidates.
(4) Consider a new plaintext P (with corresponding C and C^*) and go back to step 2 (this time step 3 only goes through the list $\mathcal{L}$ of possible candidates; if the difference computed is not in $\mathcal{D}$, remove the candidate from $\mathcal{L}$). Repeat until there remains only one candidate in $\mathcal{L}$.

If the diffusion layer does not have a maximal branch number, only a limited number of bytes of the cipher are usually affected by a given fault. Thus each pair $(C; C^*)$ gives information only on a subset of the round key bytes; the guess is made only on these bytes. The AES is a good example of this fact.

After the last round key has been found, and if it is not sufficient to retrieve the whole key, the last round is peeled off, and the attack is repeated on the reduced cipher.

V.5.4. Complexity Analysis

We compute the fraction of the round keys k^R that are suggested by a single pair $(C; C^*)$ with difference $\Delta = C \oplus C^*$. Suppose the number of possible differences Δ' before γ_R (for Δ fixed and considering all possible k^R) is 255^n. Among these, $255n$ are elements of $\mathcal{D}$. Thus the fraction of the keys surviving the test is $255n/255^n = n \cdot 255^{1-n}$.

However this computation does not take into account the fact that the number of differences Δ' before γ_R that can cause difference Δ after it is far less than 255^n; this is due to the fact that the XOR distribution table[5] of each S-box contains a lot of 0's. Thus we made the hypothesis that for the observed Δ, the fraction of elements of $\mathcal{D}$ among the corresponding possible Δ' is also about $255n/255^n$.

We conclude that the number of remaining wrong candidates for k^R after N $(C; C^*)$ pairs have been treated is about $256^n(n \cdot 255^{1-n})^N$. The conclusion (for all practical values of n) is that one pair $(C; C^*)$ is not enough to retrieve k^R, but two are (still under the hypothesis that the diffusion layer is optimal; see the AES case in section V.7 for an example where it is not).

V.5.5. A Practical Attack

As it is presented, this attack is not really practical, as it implies a guess on the last round key, that is to say a complexity $\sim 2^{8n}$. We show that slightly modifying the attack considerably reduces this complexity. Once again, for simplicity reasons we assume the diffusion layer considered is optimal. A similar technique, applied only to the bytes affected by the fault, can be used when it is not.

(1) Compute the $255n$ possible differences at the output of θ_{R-1}. Store them in a list $\mathcal{D}$.
(2) Consider 2 right ciphertext/faulty ciphertext pairs $(C; C^*)$ and $(D; D^*)$.
(3) Consider the two leftmost bytes of k^R:
    - For each of the 2^{16} candidates, compute[6]

$$\gamma_R^{-1} \circ \sigma[\langle k_1^R, k_2^R \rangle](\langle C_1, C_2 \rangle) \oplus \gamma_R^{-1} \circ \sigma[\langle k_1^R, k_2^R \rangle](\langle C_1^*, C_2^* \rangle)$$

and

$$\gamma_R^{-1} \circ \sigma[\langle k_1^R, k_2^R \rangle](\langle D_1, D_2 \rangle) \oplus \gamma_R^{-1} \circ \sigma[\langle k_1^R, k_2^R \rangle](\langle D_1^*, D_2^* \rangle).$$

[5] See section I.5 for definition of this concept.

[6] We commit a small abuse in notations by applying σ and γ_R to data of improper length. The right way to understand this is to think that e.g. $\langle C_1^*, C_2^* \rangle$ has been right-padded with 0's, and that only the 2 leftmost bytes of the output are considered.

- Compare the results with the two leftmost bytes of the $255n$ differences in list $\mathcal{D}$. Make a list $\mathcal{L}$ of the $\langle k_1^R, k_2^R\rangle$ for which a match is found for both ciphertext pairs.

(4) For each $k^\bullet \in \mathcal{L}$, try to extend it by one byte:
- Remove $k^\bullet$ from $\mathcal{L}$.
- For all 2^8 k_3^R, compute

$$\gamma_R^{-1} \circ \sigma[\langle k_2^\bullet, k_3^R\rangle](\langle C_2, C_3\rangle) \oplus \gamma_R^{-1} \circ \sigma[\langle k_2^\bullet, k_3^R\rangle](\langle C_2^*, C_3^*\rangle)$$

and

$$\gamma_R^{-1} \circ \sigma[\langle k_2^\bullet, k_3^R\rangle](\langle D_2, D_3\rangle) \oplus \gamma_R^{-1} \circ \sigma[\langle k_2^\bullet, k_3^R\rangle](\langle D_2^*, D_3^*\rangle).$$

- Compare the differences obtained with bytes 2 and 3 of the $255n$ differences in $\mathcal{D}$. If a match is found (again for both ciphertext pairs), add the newly extended key $\langle k^\bullet, k_3^R\rangle$ to $\mathcal{L}$.

(5) Repeat step 4 until elements of $\mathcal{L}$ have a length of n bytes.

(6) Apply now the first algorithm we gave using the same pairs $(C; C^*)$ and $(D; D^*)$, but consider only the candidates k^R in $\mathcal{L}$ (their number is much smaller than 2^{8n}).

The idea of this algorithm is that its first 5 steps compute a set of candidates of which the candidates selected by the first algorithm are a subset; otherwise stated, every candidate obtained by applying the first algorithm to pairs $(C; C^*)$ and $(D; D^*)$ will be returned by steps 1→5 of the second algorithm too, but the converse is not true. Thus, the first 5 steps of the second algorithm (that have a low complexity) perform a "first sorting" of the candidates. After that, the size of the set of candidates is quite small, so the first algorithm can easily deal with it.

V.5.6. Faults Occurring with a Wrong Timing

As the attacker does not always have control on the timing of the fault, it is important to be able to distinguish pairs $(C; C^*)$ resulting from faults occurring between θ_{R-2} and θ_{R-1} (we call such pairs *right pairs*) from other pairs (these are called *bad pairs*). It is trivial in the case of diffusion layers for which a 1-byte difference in the input implies an output difference affecting only some bytes of the output: in this case it is enough to observe whether some bytes are identical in both C and C^*.

But in the case of optimal diffusion layers, it is not possible to decide whether one only pair $(C; C^*)$ is a right or a bad one. However applying our attack to 2 pairs $(C; C^*)$ one of which is bad will very probably result in no solution for the key k^R. Thus we can indeed distinguish bad pairs $(C; C^*)$ from right ones, but only by considering *pairs* of ciphertext pairs $(C; C^*)$. Nevertheless the attack should be practical: if we consider that 1 ciphertext pair out of 50 is right, which is more than reasonable, we

have 5000 pairs to examine before finding two right pairs, which is still feasible.

V.5.7. Considering Other Fault Models

As we already mentioned, the attack would work as well if the faulty ciphertexts result from one random bit flip. As a matter of fact, the number of elements in list $\mathcal{D}$ would simply be reduced from $255n$ to $8n$, the remaining of the attack being all the same. As the distinguishing criteria has become stricter, the attack will in fact perform slightly better.

If the faults result from one register being permanently stuck at 0, the attack is still valid. Indeed, this fault model will result in the same difference distribution as the original one, as the data after θ_{R-2} can be assumed to be random[7]. Of course, right ciphertexts must be collected before the hardware is permanently damaged.

Finally, if it is difficult to induce only one fault, while two simultaneous faults (thus affecting 16 bits, and both properly timed) can be created, the attack still works but is less efficient; this time, the size of list $\mathcal{D}$ is $255^2 \cdot \frac{n(n-1)}{2}$.

V.6. Application to KHAZAD

V.6.1. Brief Description of KHAZAD

KHAZAD is a 64-bit block 128-bit key block cipher submitted to the NESSIE [126] European project by P.S.L.M. Barreto and V. Rijmen [8]. It has 8 rounds, whose structure is the one described in section V.5.1, with exclusive or used for key addition. Its γ layer (identical for all rounds) is made up of the application of 8 identical involutive 8×8 S-boxes. Its θ layer (also identical for all rounds) has optimal byte branch number (i.e. 9) and is also involutive.

V.6.2. Our Attack Applied to KHAZAD

Two faults occurring between θ_{R-1} and θ_{R-2} are enough to retrieve k^R (as each fault gives information on all bytes of k^R; remember that θ is optimal). However knowledge of k^R is not enough to retrieve the whole key. Thus once k^R is known the last round is peeled off. Then a fault occurring between θ_{R-2} and θ_{R-3} is exploited to select about $256^8 \cdot (8 \cdot 255^{-7}) \simeq 2105$ candidates for k^{R-1}. We conclude the attack by searching exhaustively among these candidates; knowledge of k^R and k^{R-1} allows the main key to be computed. Thus globally three faults

[7]However the ciphertext obtained will be correct with probability 1/256.

are needed.

We implemented our attack on a PC. It showed that using 2 right pairs $(C; C^*)$ we obtain one unique candidate for k^R in about 90% of the cases (otherwise 2 candidates remain, sometimes 4). One reason for this bad score happens to be related to the choice of the S-boxes: it seems that the worse an S-box is with respect to differential cryptanalysis, the better it resists our fault attack. As an illustration, we applied our attack to a modified version of KHAZAD using the AES S-boxes (whose δ-parameter is smaller than the one of the KHAZAD S-boxes); then a unique candidate is obtained from 2 right pairs $(C; C^*)$ with probability 96%. Section V.7.3 sketches an explanation for this.
Note that the number of faulty ciphertexts needed to retrieve the key is not affected by these figures; only the time complexity of the attack (which remains small anyway) is. Also, when trying to recover k^R with 2 ciphertext pairs one of which is bad, the set of candidates returned by our algorithm was always empty.

V.7. Application to the AES

The AES [47] is another example of a substitution-permutation structure, as defined in section I.2. Its block size is 128 bits, while its key size can be 128, 192, or 256 bits (see Appendix B.1 for details). We will only deal with the 128-bit key variant, as it is the most widely used. Our attack can be extended trivially to other variants.

V.7.1. Previous Work

Several papers have been written about fault analysis on the AES. We summarize here their contributions, by chronological order. A comparison with our results will be made in section V.9.

The first paper we know of is the one of J. Blömer and J.-P. Seifert [27]. Mainly, two attacks are presented:

- The first one assumes that one can *force to 0* the value of a bit, as in section V.4.2. Moreover, the attacker must be able to choose the location of the bit affected. This attack is uninteresting, as under these strong hypothesis the generic attack described in section V.4.2 is applicable. 128 faulty encryptions of plaintext $\mathbf{0}$ are required to retrieve the key using this technique.
- The second attack is implementation-dependent, and has several variants depending on the implementation. Its principle is to turn the timing attack on AES suggested by F. Koeune and J.-J. Quisquater [102] into a fault based cryptanalysis. The

fault model used also depends on the implementation. The authors claim that about 16 faulty ciphertexts (with the fault occurring at a carefully chosen location) are needed to retrieve one key byte.

In [63], C. Giraud presents two fault attacks on the AES. Both require the ability to obtain several faulty ciphertexts originating from the *same* plaintext (contrary to our attack):

- The first one assumes it is possible to induce a fault on only one bit of an intermediate result. More precisely, it exploits faults induced on one bit before the last γ layer (while we exploit faults occurring before the last diffusion layer). Under these conditions, about 50 faulty ciphertexts are necessary to retrieve the full key (provided the location of the fault can be chosen).
- The second attack exploits faults on bytes. It requires the ability of inducing faults at several chosen places, including the key schedule. The author claims that 250 faulty ciphertexts are needed (it is assumed that the attacker can choose the stage of the computation where the fault takes place, but not the exact byte), and that the time needed for computation is about 5 days.

The fault attack presented by C.-N. Chen and S.-M. Yen in [36] is an improvement of Giraud's second attack. It deals with byte faults as well, but occurring on the key schedule only. For the attack to work, several faults occurring at various locations of its last three rounds are mandatory. If these conditions on the fault location are fulfilled, from 32 to 42 faults are needed to retrieve the key.

Finally, P. Dusart, G. Letourneux, and O. Vivolo [51] take advantage of byte faults occurring between the 8^{th} and the 9^{th} `MixColumns`. Thus the fault model and the hypothesis on the fault location are exactly the same as in our attack. However the way they exploit faults is different from ours: they use the particular form of the `MixColumns` transformation and of the AES S-box to write and solve a system of equations (one by S-box) of which the unknown value is the one of the fault (i.e. of the byte difference engendered by the fault). Suggestions for 4 key bytes follow. The authors show that 5 properly timed faults are necessary to retrieve 4 key bytes. However using the same trick we will describe in the next section, they show that 10 faults occurring between the 7^{th} and 8^{th} `MixColumns` are enough to retrieve the whole key.

V.7.2. Our Results

It is easy to see that a 1-byte difference at the input of the θ layer of AES results in a 4-byte difference at its output. Concretely, it means

that a fault on one byte before the θ_{R-1} layer will give information on only 4 bytes of the last round key (the other bytes of both ciphertexts C and C^* being identical). More precisely, with the different bytes of the state numbered as in Appendix B.1:

- A fault on byte $a_{0,0}$, $a_{1,1}$, $a_{2,2}$, or $a_{3,3}$ will release information about round key bytes $k^R_{0,0}$, $k^R_{1,3}$, $k^R_{2,2}$, $k^R_{3,1}$.
- A fault on byte $a_{0,1}$, $a_{1,2}$, $a_{2,3}$, or $a_{3,0}$ will release information about round key bytes $k^R_{0,1}$, $k^R_{1,0}$, $k^R_{2,3}$, $k^R_{3,2}$.
- A fault on byte $a_{0,2}$, $a_{1,3}$, $a_{2,0}$, or $a_{3,1}$ will release information about round key bytes $k^R_{0,2}$, $k^R_{1,1}$, $k^R_{2,0}$, $k^R_{3,3}$.
- A fault on byte $a_{0,3}$, $a_{1,0}$, $a_{2,1}$, or $a_{3,2}$ will release information about round key bytes $k^R_{0,3}$, $k^R_{1,2}$, $k^R_{2,1}$, $k^R_{3,0}$.

Consider a fault occurring on one of the bytes $a_{0,0}$, $a_{1,1}$, $a_{2,2}$, or $a_{3,3}$. We compute that with one pair $(C; C^*)$ about 1036 candidates for $(k^R_{0,0}, k^R_{1,3}, k^R_{2,2}, k^R_{3,1})$ remain (see complexity analysis in section V.5). If two pairs are exploited, we are in principle left with the right candidate only. Thus with 8 faults at carefully chosen locations, we are able to recover the whole key.

However it is possible to do better. Suppose a fault occurs on one byte sometime between θ_{R-3} and θ_{R-2} (rather than between θ_{R-2} and θ_{R-1}). The corresponding difference after the θ_{R-2} layer has 4 non-zero bytes. Each of them can be exploited as described previously, and releases information about a different part of the last round key. For example, a fault on $a_{0,0}$ before θ_{R-2} will result in a non-zero difference on $a_{0,0}$, $a_{1,0}$, $a_{2,0}$, and $a_{3,0}$ after it. Thus using faults occurring sometime between θ_{R-3} and θ_{R-2} allows us to kill 4 birds with one stone. As a consequence, only 2 such faults are needed to retrieve the whole AES-128 key.

The results obtained after implementation of our attack well matched our estimates. When one fault between θ_{R-2} and θ_{R-1} was considered, the average number of candidates for 4 bytes of k^R obtained was 1046 (instead of the expected 1036). A more surprising point (a priori) was that 2 pairs $(C; C^*)$, both giving information on the same 4 bytes of k^R, allowed the retrieval of a unique value for these bytes in only 98% of the cases; otherwise two possible values for these 4 bytes remained (or even four, but it was very rare). These deviations from the expected results are due to the fact that we were making very few hypothesis on the θ layer and the S-boxes in our complexity analysis. Thus our estimations did not take into account particular features of these components. We give a more detailed explanation for this 98% figure in the next section.

Using 2 faults between θ_{R-3} and θ_{R-2}, the number of candidates left for the whole key never exceeded 16, and we obtained one only candidate in 77% of the cases. The time needed to complete the attack is a few seconds. Also, when applying the attack to 2 ciphertext pairs one of which is bad (i.e. corresponds to a fault occurring before θ_{R-3}), the set of candidates returned by our algorithm was always empty.

V.7.3. Thorough Complexity Analysis

In this section we analyze why 2 right pairs $(C; C^*)$ and $(D; D^*)$, both releasing information on the same 4 bytes of k^R, do not allow the computation of a unique value for these 4 bytes in about 2% of the cases.

Let $k^R_\bullet := \langle k^R_{0,0}, k^R_{1,3}, k^R_{2,2}, k^R_{3,1}\rangle$ denote 4 bytes of the last round key of an AES. Let $C_\bullet := \langle C_{0,0}, C_{1,3}, C_{2,2}, C_{3,1}\rangle$ and $C^*_\bullet := \langle C^*_{0,0}, C^*_{1,3}, C^*_{2,2}, C^*_{3,1}\rangle$ be a right ciphertext and its faulty counterpart, both limited to the same 4 bytes. Then the corresponding difference Δ' before γ_R is

$$\begin{aligned}
\Delta' &= \gamma_R^{-1} \circ \sigma[k^R_\bullet](C) \oplus \gamma_R^{-1} \circ \sigma[k^R_\bullet](C^*) \\
&= \gamma_R^{-1}(C \oplus k^R_\bullet) \oplus \gamma_R^{-1}(C^* \oplus k^R_\bullet) \\
&= \gamma_R^{-1}(\langle C_{0,0}, C_{1,3}, C_{2,2}, C_{3,1}\rangle \oplus \langle k^R_{0,0}, k^R_{1,3}, k^R_{2,2}, k^R_{3,1}\rangle) \\
&\quad \oplus \gamma_R^{-1}(\langle C^*_{0,0}, C^*_{1,3}, C^*_{2,2}, C^*_{3,1}\rangle \oplus \langle k^R_{0,0}, k^R_{1,3}, k^R_{2,2}, k^R_{3,1}\rangle) \\
&= [S^{-1}(C_{0,0} \oplus k^R_{0,0}) \oplus S^{-1}(C^*_{0,0} \oplus k^R_{0,0})] \\
&\quad \| [S^{-1}(C_{1,3} \oplus k^R_{1,3}) \oplus S^{-1}(C^*_{1,3} \oplus k^R_{1,3})] \\
&\quad \| [S^{-1}(C_{2,2} \oplus k^R_{2,2}) \oplus S^{-1}(C^*_{2,2} \oplus k^R_{2,2})] \\
&\quad \| [S^{-1}(C_{3,1} \oplus k^R_{3,1}) \oplus S^{-1}(C^*_{3,1} \oplus k^R_{3,1})],
\end{aligned} \tag{95}$$

where S is the AES S-box and $\|$ denotes the concatenation of two bit strings.

From (95), it is immediate to check that replacing the key candidate $k^R_\bullet$ by $k^R_\bullet \oplus C_\bullet \oplus C^*_\bullet$, or any of the 14 other candidates obtained when only some bytes of $C_\bullet \oplus C^*_\bullet$ are XORed to $k^R_\bullet$, does not change Δ'. Therefore if the pair $(C_\bullet; C^*_\bullet)$ suggests $k^R_\bullet$ as a possible candidate for the key (which will occur when $\Delta' \in \mathcal{D}$, using the notations of section V.5), then the 15 other candidates will be suggested as well.

Consider a second pair $(D; D^*)$ with $D_\bullet := \langle D_{0,0}, D_{1,3}, D_{2,2}, D_{3,1}\rangle$ and $D^*_\bullet := \langle D^*_{0,0}, D^*_{1,3}, D^*_{2,2}, D^*_{3,1}\rangle$; $D^*_\bullet$ is the faulty counterpart of $D_\bullet$. Assume $D_\bullet \oplus D^*_\bullet$ shares some bytes with $C_\bullet \oplus C^*_\bullet$; suppose for example $C_{0,0} \oplus C^*_{0,0} = D_{0,0} \oplus D^*_{0,0}$. Then if our attack applied to both $(C; C^*)$ and $(D; D^*)$ returns k^R, then $k^R \oplus \langle C_{0,0} \oplus C^*_{0,0}, 0, 0, 0\rangle$ will be returned as well.

As the probability of having the same value at a given position of $C \oplus C^*$ and $D \oplus D^*$ is $1/255$, the probability that we observe the same value at at least one position is $1 - (254/255)^4 \simeq 0.015$. So we have found the main reason why more than one key is returned in 2% of the cases. Note that this phenomenon is not specific to the AES. Furthermore this explanation could be generalized by referring to the XOR distribution table (see Chapter I) of the S-box. Consider a given difference $C_{0,0} \oplus C^*_{0,0}$ at the output of one S-box of γ_R. For a given $\Delta'_{0,0}$, we have shown that if $k^R_{0,0}$ is such that

$$S^{-1}(C_{0,0} \oplus k^R_{0,0}) \oplus S^{-1}(C^*_{0,0} \oplus k^R_{0,0}) = \Delta'_{0,0}, \tag{96}$$

then $k^R_{0,0} \oplus C_{0,0} \oplus C^*_{0,0}$ satisfies (96) as well. But it may occur that $\Delta'_{0,0}$ is obtained for other values of $k^R_{0,0}$[8]. The number of solutions $k^R_{0,0}$ of (96) is given by the value of the entry $(\Delta'_{0,0}, C_{0,0} \oplus C^*_{0,0})$ in the XOR distribution table of the S-box. Thus it appears then that paradoxically good S-boxes with respect to differential cryptanalysis are also those making our fault attack the most efficient (as they make the mean value of a non-zero entry in the XOR distribution table smaller)...

V.8. Countermeasures

First, we should mention that hardware countermeasures exist: smart cards contain security detectors that erase all the memory content or reset the microprocessor if an abnormal event occurs, such as radiations, power supply irregularity, clock glitch,... However in order to achieve a very good security for a reasonable cost, a combination of both hardware and software countermeasures is desirable. In this section we focus on algorithmic countermeasures, as hardware protections are out-of-scope.

Regarding algorithmic countermeasures, a distinction is to be made between software and hardware implementations. In the case of software implementations, the classical countermeasure is simply to perform the computation twice, compare the results, and give an output only in the case where they are identical. This method will work under the reasonable hypothesis that the attacker is not able to induce twice exactly the same fault at the same time and location. Note that practically, only the first few and the last few rounds are computed twice, as an attack exploiting faults on the middle rounds of a block cipher is much more difficult to mount[9]. It is why analysis of the resistance of block ciphers against differential fault attacks is important, as it permits to assess the number of rounds that should be protected.

[8] This explains the difference between the probability 0.02 observed experimentally and the probability 0.015 computed before.

[9] On the other hand, the first few rounds are potentially vulnerable. See the recent paper of L. Hemme [71].

The same trick cannot be applied "as is" in the case of hardware. Indeed in this case by damaging the hardware at some point, it is possible to reproduce exactly the same fault several times. The solution consists in using *hardware redundancy*. In its simplest form, the hardware is simply duplicated (which has the drawback of doubling the area taken by the implementation); the result of both encryptions is compared before outputting. However if a decryption module that is completely separate from the encryption module is present in the circuit[10], it can be used to do fault detection, by using it to decrypt intermediate results while encryption keeps processing, and checking whether the original data is obtained. This way of working has been examined for example in [88]. In [80] N. Joshi et al. show how to use the involutive character of some ciphers (KHAZAD [8], `ICEBERG` [161] (see Chapter VII)) to do fault detection (almost) without hardware overhead.

An alternate approach consists in using error correcting codes. However it is really not convincing for two reasons. The first is that several papers on the subject where written by error-correction theoreticians, who often miss the real issue of fault attacks. An extreme example is [56], where the authors suggest to insert error-correction bits in the plaintext, with a view to detect errors after decryption, i.e. much to late to permit protection against a fault attack (to their discharge, the authors seem to have been simply unaware of the fault attack issue; their only purpose was to protect against transmission errors). The second is more fundamental, and derives from a too small efficiency: for example, the system proposed by R. Karri et al. [87] detects virtually all 1-bit faults, but will fail roughly one half of the times, which is much too much, if the fault is spread over several bits.

Finally, a middle term between these two solutions (i.e. lighter than duplicating the hardware, but significantly more efficient than error-correcting codes) can sometimes be devised. For example in the particular case of the AES S-box (which, up to an affine transform, consists of an inversion in the field $\mathrm{GF}(2^8)$), M. Karpovsky et al. suggested in [86] to do partial multiplication in $\mathrm{GF}(2^8)$ between the input and the output of the S-box, and check whether the result is $1 \in \mathrm{GF}(2^8)$; the number of bits of the product that are computed depends on the security wanted.

To summarize, countermeasures against differential fault attacks exist, but they have a certain cost (as for all other side-channel attacks). Devising and analyzing fault attacks is necessary as it permits us to estimate the strength of the countermeasures to be deployed.

[10] But it is often the case that the same hardware is basically used for both encryption and decryption. Consider for example Feistel ciphers such as DES, or involutive ciphers such as KHAZAD [8] or `ICEBERG` [161] (see Chapter VII).

TABLE 1. Comparison of existing fault attacks against the AES.

Ref.	Fault Model	Fault Location	# Faulty Encryptions
[27]	Force 1 bit to 0	Chosen	128
[27]	Fct of impl.	Chosen	256
[63]	Switch 1 bit	Any bit of chosen bytes	~ 50
[63]	Disturb 1 byte	Anywhere among 4 bytes (including in the key schedule)	~ 250
[36]	Disturb 1 byte	Anywhere among 3 bytes in the key schedule	usually 32
[51]	Disturb 1 byte	Anywhere between θ_{R-3} and θ_{R-2}	10
Our result	**Disturb 1 byte**	**Anywhere between θ_{R-3} and θ_{R-2}**	**2**

V.9. Conclusion

In this chapter we first discussed the different types of faults that can be induced during the execution of a cryptographic algorithm. We have shown how these various fault models can be used to retrieve the key of a block cipher, based on several papers from the literature.

No block cipher is a priori immune against differential fault attacks. However the efficiency of such an attack may vary from cipher to cipher. Quantifying this efficiency is interesting in order to implement ad-hoc countermeasures, both cheap and efficient. As an example, it has been widely believed for years that only the last few rounds of a cipher like DES or Triple-DES needed to be protected against fault attacks. The recent paper of L. Hemme [71] will probably change this practice.

We have devised a technique of attack that can work against any algorithm with a SP-Network structure. The basic idea of our attack is to use the diffusion property of the last θ layer, in order to determine whether the difference before the last nonlinear layer γ possibly originates in a fault or not. This provides us with a distinguishing criteria for the last round key. The fault model used is quite liberal and realistic: we simply need random faults occurring on bytes. Moreover, the attack would work under several other fault models as well. The ability to choose the timing of the fault is not important either: of course only faults occurring in a given time window (between θ_{R-2} and θ_{R-1} in the general case, between θ_{R-3} and θ_{R-1} in the case of AES) are exploitable, but those occurring outside it can be discarded.

Application of our technique to the AES revealed it was pretty efficient, as under proper hypothesis on the timing of the fault only two faulty ciphertexts are needed to retrieve the key, while at least ten were previously needed. Moreover only one properly timed fault reduces the size of the key space to be explored to $1046^4 \simeq 2^{40}$, which (almost) allows an exhaustive key search.

For comparison purposes, a summary of existing attacks against the AES is given in Table 1. Among these, the most similar to ours is the one

of P. Dusart et al. [51]. The difference mainly lies in the way faults are exploited. [51] exploits the particular structure of the AES S-box and `MixColumns`, while we do not. The consequence is that their attack is not adaptable to other algorithms; ours can be used to attack KHAZAD (as we showed in section V.6), but also ciphers like Serpent [2], Anubis [7][11], or `ICEBERG` [161] (see Chapter VII). On the other hand, note that an algorithm such as SAFER++ [112] is not directly vulnerable to our attack, due to the use of two different group operations for key mixing.

It is amusing to note that it is the very simple and elegant structure of SPN structures that makes our attack so efficient... It is not clear whether ciphers with a more intricate structure could be broken with so few ciphertext pairs. More fundamental is the following problem: nowadays no differential fault attack exists exploiting faults on the middle rounds of AES. Should these rounds be protected? This question illustrates the limits of our knowledge on the subject. As in several areas of cryptography (and more specially block ciphers), the main security assessment we have is the absence of attack, but no security proofs are known. Finding such proofs in the case of fault attacks on block ciphers is an open problem.

[11] But this last must be rewritten in order to comply with our description of an SPN structure.

CHAPTER VI

Scrambling Functions: On the Security of $DeKaRT$

Abstract. This chapter deals with *scrambling functions*. These functions are intended to provide some obfuscation of data transiting inside a smart card, in order to protect against probing attacks; however due to hardware constraints they are much less secure than block ciphers.
We analyze the $DeKaRT$ scrambling function presented at CHES 2003 [65] using linear cryptanalysis. We show that despite its key-dependent behavior, $DeKaRT$ still has strongly linear structures, that can be exploited even under the particular hypothesis that only one bit of the ciphertexts is available to the attacker (as it is the case in the context of probing attacks), and using very few plaintext-ciphertext pairs. The attack methodology we describe could be applied to other data scrambling primitives exhibiting highly biased linear relations.
The results presented in this chapter have originally been published in [143].

VI.1. Introduction

In this chapter we deal with a different type of side-channel attack, namely *invasive* techniques, by opposition with *non-invasive* techniques such as those mentioned in the previous chapter. The particularity of invasive attacks is that they require depackaging of the card, which constitutes a challenge in itself, as smart card manufacturers foresee protections to prevent this [3, 104]: typically the chip is covered by meshes of conductors, and any interruption or short-circuit in them causes the card to delete all its sensitive data. After a successful depackaging, the attacker gets direct access to the card's components.

More precisely, we focus on *probing attacks*: by introducing a conductor in some point of a tamper-resistant chip, one is able to monitor the electric signal at this point, and therefore to observe information transiting there. For example, the bus connecting the RAM and the microprocessor is particularly vulnerable, as it has an easily recognizable structure, and conveys secret information, such as RSA private keys.

An obvious algorithmic countermeasure to protect against probing attacks would be to encrypt transiting data using a block cipher. However, as the encryption has to be performed by a hardware circuit preferably within one only clock cycle (in order to be transparent for the other components of the smart card), and as strong constraints exist regarding the size of the circuit, using a classical block cipher such as DES or AES is illusory. On the other hand, as the probing technique would usually allow the attacker to see only one or a few bits of the encrypted data (because of the difficulty of depackaging), security requirements for such an encryption function are lowered compared to classical criteria for block ciphers. Such a "light" encryption algorithm is usually referred as a *data scrambling* (or *obfuscation*) function.

In section VI.2, we briefly present the contribution of E. Brier, H. Handschuh and C. Tymen [29] on the theory of data scrambling. In section VI.3, we present a more secure but heavier structure, due to J.D. Golić: $DeKaRT$ [65]. Then in the remaining sections we analyze the security of $DeKaRT$ in a probing scenario where the attacker has access to one ciphertext bit only, by using linear cryptanalysis (see [113] and Chapter I). These results were initially presented in [143]. We conclude that security of $DeKaRT$ is a bit deceptive given its complexity.

VI.2. On the Use of Bit Permutations for Data Scrambling

The scrambling functions examined by E. Brier et al. in [29] consist in keyed bit permutations (thus only the position of the bits is modified, depending on the key). More formally, a (n, k)-*keyed permutation* is defined as a map from the key space $\mathbb{Z}_2^k$ to the group S_n of the permutations of $\{0, ..., n-1\}$:

$$\begin{aligned} \sigma : \mathbb{Z}_2^k &\to S_n \\ K &\mapsto \sigma_K \end{aligned}$$

They assume the attacker is allowed to play with the microprocessor, which implies that she can send known data to the memory. Her goal is to decipher a secret data present in the card and read from the memory at some time. She has access only to a subset of the bits $(c_{n-1}, ..., c_1, c_0)$ of the corresponding ciphertexts. Then the following definition makes sense.

DEFINITION 32. The ***degree of freedom*** of an (n, k)-keyed permutation is the smallest integer $m \geq 1$ such that there exists an $(m+1)$-tuple $(i_1, ..., i_{m+1})$ of pairwise distinct elements of $\{0, ..., n-1\}$ such that the map

$$\begin{aligned} \{\sigma_K | K \in \mathbb{Z}_2^k\} &\to \{0, ..., n-1\}^{m+1} \\ \sigma_K &\mapsto (\sigma_K(i_1), ..., \sigma_K(i_{m+1})) \end{aligned}$$

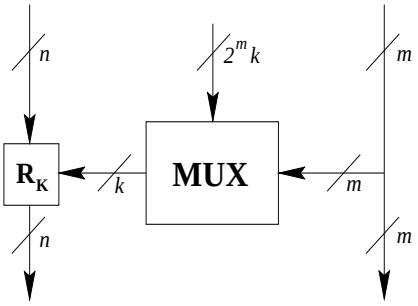


FIGURE 1. Generic *DeKaRT* building block.

is injective.

Informally, the degree of freedom corresponds to the maximal number of ciphertext bits an attacker may learn without being able to retrieve σ_K. The authors explain the construction of several (n, k)-keyed permutations; the design criteria are a large degree of freedom, small number of gates and logical depth. The best constructions have a degree of freedom of at least $n/2 - 1$. And as an example, for a 16-bit block size[1] the number of multiplexers of one of the constructions is 88 and the logical depth 4.

VI.3. A New Paradigm: *DeKaRT*

Keyed bit permutations have the drawback to be perfectly linear. The solution proposed by J.D. Golić [65] does not have this disadvantage; however its hardware complexity and logical depth are greater.

The overall *DeKaRT* construction looks a bit like a SP-Network: it alternates linear key-independent layers (denoted BP for **B**it **P**ermutation) and non-linear key-dependent layers (denoted KT for **K**eyed **T**ransform).

The key-dependent layer is made out of the parallel application of a *generic building block* as the one depicted in Figure 1. A *generic building block* acts on a small number of input data bits which are divided into two groups of m and n bits. The m input bits are used for control and are passed to the output intact, like in the Feistel structure. They are used to select k out of $2^m k$ key bits by the multiplexer (MUX) circuit with m control bits, $2^m k$ input bits and k output bits. Depending on the key bits, the multiplexer can implement any possible function. Finally, R_K denotes a family of invertible functions parameterized by the key bits K. More desirable criteria for its choice are given in the original paper.

[1]Due to the sometimes small size of the data exchanged and the need for efficiency, the block size of a data scrambling function is small.

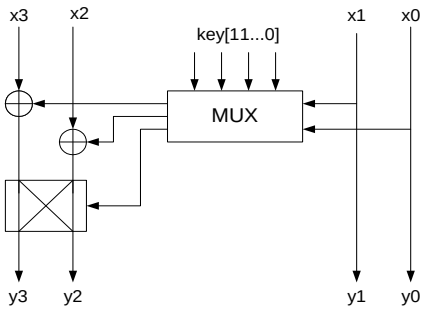


FIGURE 2. Elementary $DeKaRT$ building block.

As indicated by its name, the BP layer consists in bit permutations. Moreover two designs rules were imposed in order to optimize diffusion:

- The m control bits in each layer should be used as the transformed bits in the next layer. It imposes $m \leq n$.
- For each generic block of a given KT layer, input bits should come from the maximum possible number of blocks in the previous KT layer.

The inverse $DeKaRT$ network is obtained by replacing R_K by R_K^{-1} and BP by BP^{-1}. Note that the name of the construction comes from *"Data-chooses-Key-chooses-Reversible_Transformation"*, which is condensed by D $\rightarrow$ K $\rightarrow$ RT or $DeKaRT$.

VI.4. Specification of a Concrete Instance of $DeKaRT$ and Notations

In Golić's paper a more concrete instance of $DeKaRT$ is given, but it is not fully specified. We believe it is a mistake, as security analysis requires a completely specified cipher; and security of a cipher which has not been subject to a substantial effort regarding security analysis can be questioned (except if a security proof is given, which is never the case for block ciphers). It is why we give here a full specification of a $DeKaRT$ instance.

Golić suggests an *elementary DeKaRT building block* with parameters $(m, n, k) = (2, 2, 3)$ as shown in Figure 2 where after a XOR with 2 key bits, x_3 and x_2 pass through a conditional switch parameterized by a third key bit. We will use this building block in our specification. Each box requires 12 key bits. We will denote these bits by

$$(k^{(11)}, k^{(10)}, k^{(01)}, k^{(00)}) = \\ (k_X^{(11)}, k_{\oplus 1}^{(11)}, k_{\oplus 2}^{(11)}; k_X^{(10)}, k_{\oplus 1}^{(10)}, k_{\oplus 2}^{(10)}; k_X^{(01)}, k_{\oplus 1}^{(01)}, k_{\oplus 2}^{(01)}; k_X^{(00)}, k_{\oplus 1}^{(00)}, k_{\oplus 2}^{(00)}),$$

TABLE 1. The BP layer.

Input Bit Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Output Bit Position	5	0	15	10	1	4	11	14	13	8	7	2	9	12	3	6

where $k^{(i)}$ denotes the three key bits selected by control bits $(x_1, x_0) = i$; $k_X^{(i)}$ is conditioning the switch, while $k_{\oplus 1}^{(i)}$ is XORed with x_3 and $k_{\oplus 2}^{(i)}$ is XORed with x_2. The set of such 12 key bits is called a *subkey*.

The instance of $DeKaRT$ we will consider acts on blocks of 16 bits. Thus KT consists in the parallel application of 4 such elementary blocks. The number of rounds considered is 5 (this number is given as an example in [65], p.104). We will denote the subkeys parameterizing the four elementary blocks of the r^{th} round by $RK_3^r, RK_2^r, RK_1^r, RK_0^r$, from left to right. The set of 4 such subkeys is called a *round key* (thus it has 48 bits).

We arbitrarily chose a bit permutation layer following the rules given in section VI.3. It is given in Table 1.

Finally, as suggested in [65], the key expansion algorithm also alternates "keyed" layers (with the key being an arbitrarily chosen constant), with bit permutations layer. The 48-bit data obtained every two rounds is used as a round key. The keyed layer is made out of the parallel application of 16 *reduced DeKaRT building blocks* with 3-bit input and output, alternating with a bit permutation layer complying with the design rules given above. Figure 3 pictures part of this algorithm. Note that the bit permutation layer could have been chosen less "regular", but in fact its influence on the results of our attack is negligible.

VI.5. Analysis of an Elementary $DeKaRT$ Block

A $DeKaRT$ building block generates key-dependent Boolean functions. If a (2,2,3) block is used, 4096 substitution tables (4×4 bits) can be generated, as this type of block is parameterized by 12 key bits.

In this section, we investigate the possible linear approximations of an elementary $DeKaRT$ block. For each output bit, there exist $2^4 - 1$ possible non-trivial input masks (i.e. $2^4 - 1$ possible linear combinations of input bits) and if we combine output bits together, we have $(2^4 - 1) \times (2^4 - 1)$ possible non-trivial linear approximations of the $DeKaRT$ block. For such a small block size, the problem of finding good linear approximations is therefore easily solved by exhaustive search.

As in Chapter I, we define the ***bias*** of a linear approximation that holds with probability p as $\varepsilon = |p - 1/2|$. We also denote a linear approximation with probability $p = 0$ or $p = 1$ as a ***perfect*** linear approximation.

We computed the bias $\varepsilon_{\max}$ of the best linear approximation of each of the 4096 substitution tables; however approximations involving bits

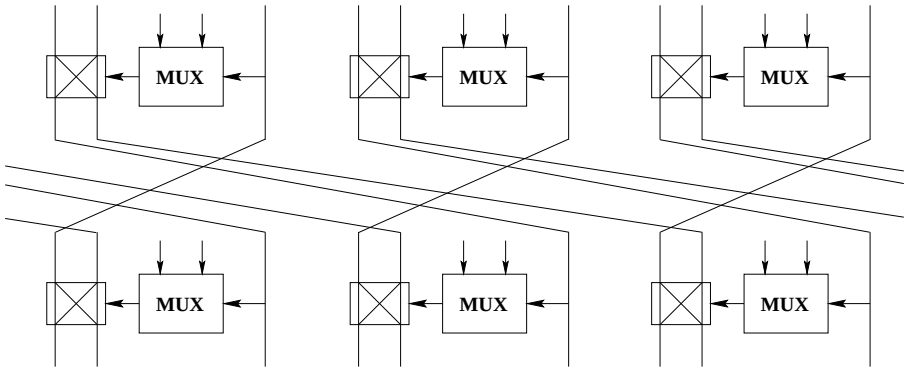


FIGURE 3. a part of the key expansion.

TABLE 2. Linear approximations of a single $DeKaRT$ block.

$\varepsilon_{\max}$	1/2	3/8	1/4
Nbr. Approx.	2304	1024	768

y_0, y_1 only were rejected as they are obviously perfectly linear. The results are summarized in Table 2. We observe that more than one half of the generated tables present perfect linear approximations and may therefore be considered as very weak from a cryptographic point of view.

On the basis of this first experiment, we may therefore assess that a large number of keys will generate weak building blocks for the scrambling function as they are perfectly approximated by a linear approximation. This motivated a more general analysis.

VI.6. The Attack

The scenario of the attack is similar to the one of section VI.2. We assume the attacker is allowed to play with the microprocessor, which implies that she can send known data to the memory. She has access to one only bit of the encrypted data via a probing attack. Formally, this means that she can obtain a certain number m of pairs (P, c), where P is a known plaintext, and c is one fixed bit of the corresponding ciphertext C. Her goal is to obtain information about a secret data (such as an RSA private key) present in the card and read from the RAM at some time. Thus it it **not** a key recovery attack, in the sense that we do not intend to retrieve the key used for $DeKaRT$ data scrambling (it can be changed at each use of the card anyway).

Due to the building blocks of $DeKaRT$ being implemented using key-dependent MUXs, the probability of a linear relation between a given bit

of the ciphertext and some bits of the plaintext is highly key-dependent. In our attack, pairs (P, c) are used to identify a linear relation between only one bit of the ciphertext and several bits of the plaintext, that holds with a high bias for the key used. Then when the secret data passes through the channel between the RAM and the processor, the relation we just identified permits probabilistic information about it to be retrieved.

Let λ be the linear relation we are considering. Knowing m pairs (P, c), we count how many of these satisfy the linear relation λ; let n_λ^m be the random variable corresponding to this number. We would like to compute the probability that λ holds for a random plaintext, provided n_λ^m takes value B

$$P_{K,P}[\lambda \text{ holds} | n_\lambda^m = B]. \tag{97}$$

$|P_{K,P}[\lambda \text{ holds} | n_\lambda^m = B] - 1/2|$ gives the reliability of the prediction we will make. Let us define the random variable N_λ as the number of plaintexts for which λ holds, out of all 2^{16} plaintexts this time. Then we have:

$$\begin{aligned}
\mathcal{P}_{K,P} \quad & [\lambda \text{ holds} | n_\lambda^m = B] \\
= \quad & \sum_{A=0}^{2^{16}} \mathcal{P}[\lambda \text{ holds} | N_\lambda = A] \cdot \mathcal{P}[N_\lambda = A | n_\lambda^m = B] \\
= \quad & \sum_{A=0}^{2^{16}} A/2^{16} \cdot \mathcal{P}[N_\lambda = A | n_\lambda^m = B] \\
= \quad & \sum_{A=0}^{2^{16}} A/2^{16} \cdot \frac{\mathcal{P}[n_\lambda^m = B | N_\lambda = A] \cdot \mathcal{P}[N_\lambda = A]}{\sum_{A'=0}^{2^{16}} \mathcal{P}[N_\lambda = A'] \cdot \mathcal{P}[n_\lambda^m = B | N_\lambda = A']} \\
\cong \quad & \frac{\sum_{A=0}^{2^{16}} A/2^{16} \cdot \mathcal{P}[\mathrm{Bi}(m, A/2^{16}) = B] \cdot \mathcal{P}[N_\lambda = A]}{\sum_{A'=0}^{2^{16}} \mathcal{P}[\mathrm{Bi}(m, A'/2^{16}) = B] \cdot \mathcal{P}[N_\lambda = A']}
\end{aligned} \tag{98}$$

where $\mathrm{Bi}(.,.)$ denotes the binomial distribution law, the third equality uses Bayes formula, and the final approximation assumes $m \ll 2^{16}$. Thus computation of (97) requires knowledge of the probability distribution of N_λ, i.e. $\{\mathcal{P}_K[N_\lambda = A]\}_{A=0}^{2^{16}}$. The next section will show how such a distribution can be computed.

VI.7. Computing Probability Distribution for a Linear Relation Through a 5-Round Cipher

Consider 5 rounds of $DeKaRT$, beginning and ending with a keyed layer (see Figure 4). We denote the plaintext by $(p_{15}, p_{14}, ..., p_1, p_0)$ or

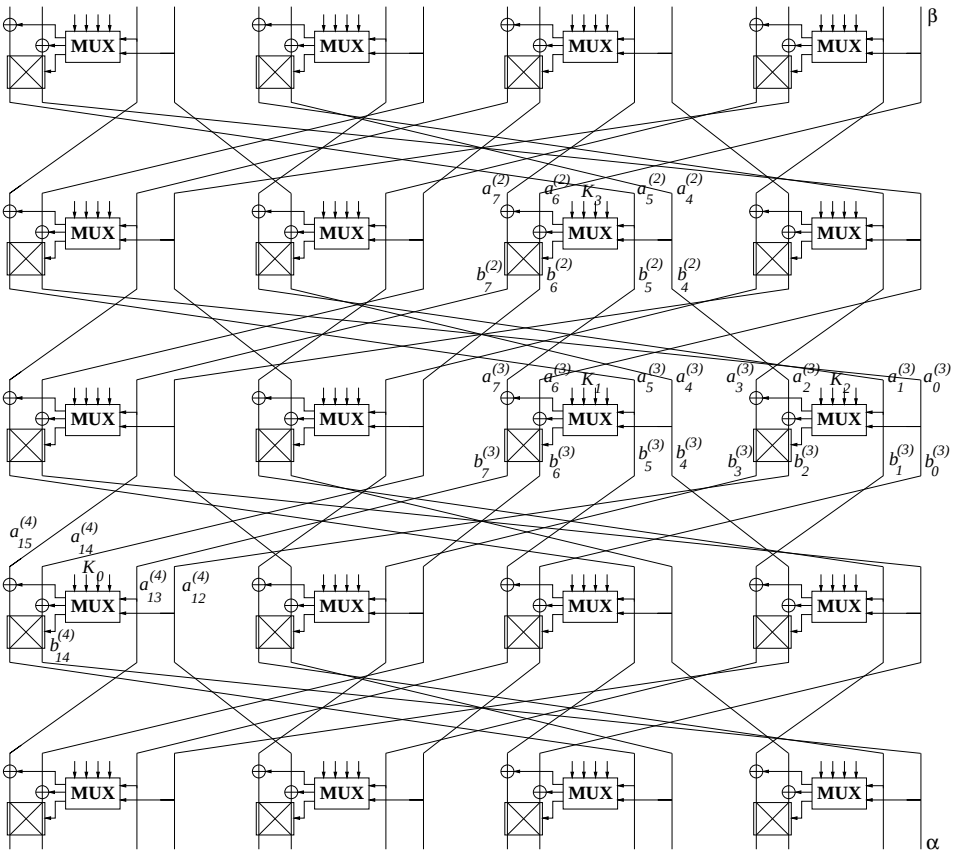


FIGURE 4. Linear approximation through a 5-round *DeKaRT*.

$(a_{15}^{(1)}, a_{14}^{(1)}, ..., a_1^{(1)}, a_0^{(1)})$. Moreover

$$KT((a_{15}^{(i)}, a_{14}^{(i)}, a_{13}^{(i)}, ..., a_1^{(i)}, a_0^{(i)})) =: (b_{15}^{(i)}, b_{14}^{(i)}, b_{13}^{(i)}, ..., b_1^{(i)}, b_0^{(i)}),$$

and

$$BP((b_{15}^{(i)}, b_{14}^{(i)}, b_{13}^{(i)}, ..., b_1^{(i)}, b_0^{(i)})) =: (a_{15}^{(i+1)}, a_{14}^{(i+1)}, a_{13}^{(i+1)}, ..., a_1^{(i+1)}, a_0^{(i+1)}).$$

The exponent denotes the round number and the ciphertext will be denoted by

$$(b_{15}^{(5)}, b_{14}^{(5)}, ..., b_1^{(5)}, b_0^{(5)}) \text{ or } (c_{15}, c_{14}, ..., c_1, c_0).$$

Consider one bit $\alpha := b_0^{(5)}$ of the ciphertext after a 5-round cipher. As an example, we will analyze the linear relation between α and $\beta := a_0^{(1)}$ (see Figure 4). Other relations can be analyzed similarly. We write successively:

- $\alpha = b_{14}^{(4)}$.

- $b_{14}^{(4)}$ is a function of $(a_{15}^{(4)}, a_{14}^{(4)}, a_{13}^{(4)}, a_{12}^{(4)})$ depending on $K_0 := RK_3^4$ (see Figure 4).
- $(a_{15}^{(4)}, a_{14}^{(4)}) = (b_7^{(2)}, b_6^{(2)})$.
- $a_{13}^{(4)} = b_7^{(3)}$ is a function of $(a_7^{(3)}, a_6^{(3)}, a_5^{(3)}, a_4^{(3)})$ depending on $K_1 := RK_1^3$. As for β fixed $a_6^{(3)}$ is *active* (i.e. it takes values 0 and 1 equally often; see Chapter III) and as this bit affects α only by means of the block keyed by K_1, key bits $k_{\oplus 2}^{(i)}$ of K_1 do not affect the probability of the equation $\alpha = \beta$.
- $a_{12}^{(4)} = b_2^{(3)}$ is a function of $(a_3^{(3)}, a_2^{(3)}, a_1^{(3)}, a_0^{(3)})$ depending on $K_2 := RK_0^3$. For the same kind of reason as before, key bits $k_{\oplus 1}^{(i)}$ of K_2 do not affect the probability.
- Finally, $(b_7^{(2)}, b_6^{(2)})$ is a function of $(a_7^{(2)}, a_6^{(2)}, a_5^{(2)}, a_4^{(2)})$ depending on $K_3 := RK_1^2$. Still using the same arguments, we note that key bits $k_{\oplus 1}^{(i)}$ of K_3 do not affect the probability.

Thus the probability of $\alpha = \beta$ (computed over all 2^{16} plaintexts) depends on 4 subkeys (or at least part of them): K_0, K_1, K_2, K_3. We write them as:

$$K_0 = (k_0^{(11)}, k_0^{(10)}, k_0^{(01)}, k_0^{(00)})$$
$$K_1 = (k_1^{(11)}, k_1^{(10)}, k_1^{(01)}, k_1^{(00)})$$
$$K_2 = (k_2^{(11)}, k_2^{(10)}, k_2^{(01)}, k_2^{(00)})$$
$$K_3 = (k_3^{(11)}, k_3^{(10)}, k_3^{(01)}, k_3^{(00)})$$

There are $12 + 3 \times 8 = 36$ subkey bits implied. A priori it does not allow easy exhaustive computation of the probability for every key (complexity $2^{36} \times 2^{16}$). However there are groups of values for which the associated probability is the same. Let us say that 2 subkeys K_1 and K_{1*} are *equivalent* if for any K_0, K_2, K_3, the probability associated to (K_0, K_1, K_2, K_3) and (K_0, K_{1*}, K_2, K_3) is identical; Equivalence between 2 subkeys K_2 and K_{2*} is defined similarly. We can make the following observations:

- For β fixed $(a_5^{(3)}, a_4^{(3)})$ is *active* (i.e. it takes values 00, 01, 10 and 11 equally often; see Chapter III). Therefore two subkeys K_1 and K_{1*} such that there exists a permutation $\Pi : \{0,1\}^2 \to \{0,1\}^2$ satisfying $k_1^{(i)} = k_{1*}^{(\Pi_i)} (\forall i \in \{00, 01, 10, 11\})$ are equivalent.
- The same argument can be used for K_2.
- As from the output of the block keyed by K_1 only bit $b_7^{(3)}$ matters, if $k_{1,X}^{(i)} = 1$ (for some $i \in \{00, 01, 10, 11\}$), then $k_{1,\oplus 1}^{(i)}$ does not affect the probability. Otherwise stated, if K_1 and K_{1*} are such that $k_1^{(i)} = (1, 0, 0)$ while $k_{1*}^{(i)} = (1, 1, 0)$ (other bits being the same), they are equivalent.

TABLE 3. Equivalence classes for K_1 and K_2 with their cardinalities.

K_1	#	K_2	#
(000; 000; 000; 000)	16	(000; 000; 000; 000)	16
(000; 000; 000; 010)	448	(000; 000; 000; 001)	448
(000; 000; 000; 100)	128	(000; 000; 000; 100)	128
(000; 000; 010; 010)	1120	(000; 000; 001; 001)	1120
(000; 000; 010; 100)	896	(000; 000; 001; 100)	896
(000; 010; 010; 010)	448	(000; 001; 001; 001)	448
(000; 010; 010; 100)	896	(000; 001; 001; 100)	896
(010; 010; 010; 010)	16	(001; 001; 001; 001)	16
(010; 010; 010; 100)	128	(001; 001; 001; 100)	128

- With the same kind of argument, $k_2^{(i)} = (1,0,0)$ and $k_{2*}^{(i)} = (1,0,1)$ are equivalent.
- Similarly, as output $b_{15}^{(4)}$ of the block keyed by K_0 does not matter, we have:

$$k_0^{(i)} = (0,0,0) \sim k_0^{(i)} = (0,1,0) \qquad k_0^{(i)} = (0,0,1) \sim k_0^{(i)} = (0,1,1)$$
$$k_0^{(i)} = (1,0,0) \sim k_0^{(i)} = (1,0,1) \qquad k_0^{(i)} = (1,1,0) \sim k_0^{(i)} = (1,1,1)$$

- Suppose $\exists i,j : k_1^{(i)} = (0,0,0)$ and $k_1^{(j)} = (0,1,0)$. Then, the key bit added to $a_7^{(3)}$ is 0 for the 2^{14} plaintexts for which $(a_5^{(3)}, a_4^{(3)}) = i$; it is 1 for the 2^{14} plaintexts for which $(a_5^{(3)}, a_4^{(3)}) = j$. Thus (taking into account that for $(a_5^{(3)}, a_4^{(3)})$ fixed $a_7^{(3)}$ is active) when $(a_5^{(3)}, a_4^{(3)}) \in \{i,j\}$, $b_7^{(3)} = 1$ one half of the times.
  Now if we replace $k_1^{(i)}$ and $k_1^{(j)}$ by $(1,0,0)$, when $(a_5^{(3)}, a_4^{(3)}) \in \{i,j\}$ we have $b_7^{(3)} = a_6^{(3)}$. As $a_6^{(3)}$ is active (for fixed $(a_5^{(3)}, a_4^{(3)})$), we still have that $b_7^{(3)} = 1$ one half of the times.
  The conclusion is that if a given key is such that $\exists i,j : k_1^{(i)} = (0,0,0)$ and $k_1^{(j)} = (0,1,0)$, then if $k_1^{(i)}$ and $k_1^{(j)}$ are replaced by $(1,0,0)$ the key obtained is equivalent to the former one.

- Similarly, if $\exists i,j : k_2^{(i)} = (0,0,0)$ and $k_2^{(j)} = (0,0,1)$, replacing these bits by $k_2^{(i)} = k_2^{(j)} = (1,0,0)$ does not change the probability.

Putting all these observations together, there are 9 equivalence classes for K_1 as well as for K_2. They are given in Table 3, with the number of elements in each class (out of 2^{12}).

Finally, the number of different quadruples (K_0, K_1, K_2, K_3) to explore in order to compute the probability distribution $\{\mathcal{P}_K[N_{\alpha=\beta} = A]\}_{A=0}^{2^{16}}$ is $9^2 \cdot (2^8)^2 \cong 2^{22}$. This number could be further reduced, by exploiting

more complex equivalences such as: "if K_0 has such value, then value of K_1 does not matter". However it is not necessary as with complexity $2^{22}\cdot 2^{16}$, the probability distribution of $N_{\alpha=\beta}$ is computable. It is roughly given in Table 4. Out of the $2^{16}+1$ a priori possible values for $N_{\alpha=\beta}$, only 199 occur with a non-zero probability. We only mention the ones having probability ≥ 0.01. Moreover we give probabilities associated with intervals. Because of theorem 33, the table only goes from 0 to 2^{15}.

THEOREM 33. *For* $A \in \{0, ..., 2^{16}\}$,

$$\mathcal{P}_K[N_{\alpha=\beta} = A] = \mathcal{P}_K[N_{\alpha=\beta} = 2^{16} - A].$$

PROOF. For a given $K_0 := RK_3^4$, consider the transform $\widehat{.} : K_0 \to \widehat{K_0}$ defined by:

- For $i \in \{00, 01, 10, 11\}$, $\widehat{k_{0,X}^{(i)}} = k_{0,X}^{(i)}$.
- For $i \in \{00, 01, 10, 11\}$, if $k_{0,X}^{(i)} = 0$ then $\widehat{k_{0,\oplus 1}^{(i)}} = k_{0,\oplus 1}^{(i)}$ and $\widehat{k_{0,\oplus 2}^{(i)}} = \overline{k_{0,\oplus 2}^{(i)}}$.
- For $i \in \{00, 01, 10, 11\}$, if $k_{0,X}^{(i)} = 1$ then $\widehat{k_{0,\oplus 1}^{(i)}} = \overline{k_{0,\oplus 1}^{(i)}}$ and $\widehat{k_{0,\oplus 2}^{(i)}} = k_{0,\oplus 2}^{(i)}$.

If is easy to check that for a given plaintext, changing K_0 to $\widehat{K_0}$ also changes α to $\overline{\alpha}$. Therefore if a quadruple (K_0, K_1, K_2, K_3) is such that $N_{\alpha=\beta} = A$, the quadruple $(\widehat{K_0}, K_1, K_2, K_3)$ is such that $N_{\alpha=\beta} = 2^{16} - A$. It is easy to check that $\widehat{.}$ is an automorphism and an involution. Consider the sets

$$\begin{aligned} S_A &= \{(K_0, K_1, K_2, K_3) \text{ s.t. } N_{\alpha=\beta} = A\}, \\ \widehat{S_A} &= \{(\widehat{K_0}, K_1, K_2, K_3) \text{ s.t. } (K_0, K_1, K_2, K_3) \in S_A\}. \end{aligned} \tag{99}$$

Because $\widehat{.}$ is an automorphism we have

$$|S_A| = |\widehat{S_A}|. \tag{100}$$

The observation we made before implies $\widehat{S_A} \subseteq S_{2^{16}-A}$. Thus we have $\widehat{S_{2^{16}-A}} \subseteq S_A$ as well, which implies $\widehat{\widehat{S_{2^{16}-A}}} \subseteq \widehat{S_A}$ and, because $\widehat{.}$ is an involution, $S_{2^{16}-A} \subseteq \widehat{S_A}$. So we get

$$\widehat{S_A} = S_{2^{16}-A}. \tag{101}$$

(100) and (101) imply $|S_A| = |S_{2^{16}-A}|$. □

It is worth mentioning that in the previous discussion we made the (classical) hypothesis that the round keys are independent and uniformly distributed, while in practice they derive from the master key using the key expansion described in section VI.4; in fact some quadruples

TABLE 4. Probability distribution of $N_{\alpha=\beta}$.

$N_{\alpha=\beta}$	$\mathcal{P}_K$
$\in [0, 16383]$	0.006
$= 16384$	0.010
$\in [16385, 22527]$	0.029
$= 22528$	0.013
$\in [22529, 24575]$	0.013
$= 24576$	0.053
$\in [24577, 26623]$	0.024
$= 26624$	0.028
$\in [26625, 27647]$	0.009
$= 27648$	0.022
$\in [27649, 28671]$	0.014
$= 28672$	0.045
$\in [28673, 29695]$	0.012
$= 29696$	0.026
$\in [29697, 30719]$	0.013
$= 30720$	0.050
$\in [30721, 31743]$	0.015
$= 31744$	0.016
$\in [31745, 32767]$	0.013
$= 32768$	0.178

(K_0, K_1, K_2, K_3) simply cannot be derived from a master key. Computation of the value taken by $N_{\alpha=\beta}$ for a small number of random keys from which round keys have been derived perfectly validated the hypothesis.

VI.8. Searching for Other Linear Relations Through 5 Rounds

The procedure described in the previous section to compute the distribution of N_λ for a given linear relation λ is complicated and has non-negligible time complexity. It is however possible to identify linear relations having a big mean bias by evaluating this bias using only a part of the 2^{16} plaintexts, and this for a relatively small number of keys. Doing this, we observed that there are "families" of linear relations having about the same mean bias. Moreover linear relations from some families have their output bit belonging to $S_1 \equiv \{c_0, c_1, c_4, c_5, c_8, c_9, c_{12}, c_{13}\}$, while those from the other families have their output bit belonging to $S_2 \equiv \{c_2, c_3, c_6, c_7, c_{10}, c_{11}, c_{14}, c_{15}\}$. This is due to the fact that the last round of $DeKaRT$ need not be approximated if the output bit $\in S_1$. Details about the families with the best mean bias are given in Table 5.

TABLE 5. Families of linear relations with the best mean bias.

Mean bias	Number of lin. rel.	Output bit
$7 \cdot 10^{-2}$	16	$\in S_1$
$4 \cdot 10^{-2}$	64	$\in S_2$
$2.5 \cdot 10^{-2}$	192	$\in S_1$
$1.5 \cdot 10^{-2}$	64	$\in S_2$

TABLE 6. Linear relations through 5-round $DeKaRT$ with the highest mean bias.

$p_0 \oplus c_0$	$p_4 \oplus c_1$	$p_8 \oplus c_8$	$p_{12} \oplus c_9$
$p_0 \oplus c_5$	$p_4 \oplus c_4$	$p_8 \oplus c_{13}$	$p_{12} \oplus c_{12}$
$p_1 \oplus c_1$	$p_5 \oplus c_0$	$p_9 \oplus c_9$	$p_{13} \oplus c_8$
$p_1 \oplus c_4$	$p_5 \oplus c_5$	$p_9 \oplus c_{12}$	$p_{13} \oplus c_{13}$

The linear relations of the first family (with bias $\sim 7 \cdot 10^{-2}$) are given in Table 6.

VI.9. Implementation of the Attack

As explained in section VI.6, we assume the attacker knows for example 128 pairs $\{(P^j, c_i^j)\}_{j=0,\ldots,127}$, where P^j is a plaintext and c_i^j is the i^{th} bit of the corresponding ciphertext. One attack strategy is:

(1) Consider all $2^{16} - 1$ possible input masks μ (i.e. all possible linear combinations of input bits).

(2) For each of them, compute the bias of $\mu \bullet P = c_i$ over the 128 pairs.

(3) The attacker intercepts a ciphertext bit c_i^* of which she does not know the corresponding plaintext P^*. The input mask μ^* with the highest bias (computed at step 2) is used to predict the unknown bit $\mu^* \bullet P^*$.

The efficiency of this algorithm is measured by the bias associated to μ^*, **computed over all 2^{16} plaintexts** this time. Indeed, it gives the reliability of the guess made at step 3. Practical experiments show that we have a mean bias of 0.059 when the ciphertext bit considered $\in S_1$ and of 0.022 when it is in S_2.

However it is possible to do better if in step 1 of the attack, we restrain ourself to the 336 relations mentioned in section VI.8 (or more precisely, to those of them concerning bit c_i). Then the bias obtained is 0.107 when the ciphertext bit considered $\in S_1$ and 0.074 when it is in S_2. Moreover the probability computed over 128 plaintexts almost always "goes in the same direction" than the one computed on all 2^{16} plaintexts

TABLE 7. Mean bias as a function of the number of pairs known by the attacker

# pairs	if $c_i \in S_1$	if $c_i \in S_2$
64	0.095	0.067
128	0.107	0.074
256	0.118	0.083
512	0.123	0.087
1024	0.127	0.092

(i.e. suggests the same value for $\mu \bullet P \oplus c_i$). This significant improvement is due to the fact that if we consider all possible input masks, it is often the case that estimation on 128 plaintexts happens to emphasize a linear relation which in fact has a small (or null) bias when computed over all 2^{16} plaintexts; a pre-selection of "a priori good" input masks greatly reduces this phenomenon. It is this improvement that motivated the research of *a priori* good linear relations described in section VI.8.

In Table 7 we give mean biases for different numbers of pairs (P^j, c_i^j) known by the attacker. We insist on the fact that these figures are *mean* biases. This means that sometimes the bias associated to μ^* will be 0, which means that the attack has completely failed. Other times the bias will be 1/4 (or even 1/2) and the information gained by the attacker is real. The attacker must be able to compute a priori the bias she can expect. It is given by equation (2) in section VI.6.

As an improvement to the attack, it could also be possible to consider several approximations (implying the same ciphertext bit c_i) simultaneously in order to retrieve more information. However this is far from trivial, as the possible correlation between these approximations must be taken into account. The paper from A. Biryukov et al. [23] could help in this context.

VI.10. Conclusion

In this chapter we have seen that $DeKaRT$, despite the fact that its structure is significatively more complex than previous primitives, is vulnerable to linear cryptanalysis. Even using one only bit of the ciphertext (as it is often the case in the context of probing attacks), it is still possible to obtain information about an unknown plaintext using very few known (Plaintext, Ciphertext bit) pairs. Moreover some keys are much more vulnerable than others in that respect, which is a drawback in itself. Furthermore we can expect a much better attack if several bits of ciphertext are available, as besides allowing retrieval of more bits of information about the plaintext, linear combinations of

these ciphertext bits can be considered, with hopefully a better mean bias.

We do not claim *DeKaRT* is useless for data scrambling: indeed, the requirements for such a type of primitive can be relaxed in comparison with usual requirements for block ciphers. More than the overall structure, some proposals for the number of rounds provided in the original paper seem to be too optimistic for a really strong security.

Our purpose was rather to show that such type of key-dependent transform still has strongly linear structures. Golić suggests that his method could be suitable for design of hardware-oriented block ciphers. While it is clear that a *DeKaRT* structure with 20 or 30 rounds (and a classical block size) would be much more difficult to break than the reduced version we were dealing with, we believe that classical paradigms of block cipher design (i.e., using constant and highly non-linear S-boxes) are more promising. Designing such a block cipher with hardware efficiency as a primary goal is the topic of the next chapter.

CHAPTER VII

ICEBERG

An Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware

Abstract. The design of a block cipher must take security but also implementation efficiency issues into account. This chapter presents ICEBERG, a block cipher deliberately optimized towards efficiency in reconfigurable hardware (FPGAs).
ICEBERG operates on 64-bit blocks and uses 128-bit keys. All its components are involutional, which allows very efficient combinations of encryption and decryption. Furthermore hardware implementations of ICEBERG allow changing the key and Encrypt/Decrypt mode for every plaintext, without any performance loss and the round keys are derived "on-the-fly" in encryption and decryption modes (no storage of round keys is needed). The resulting designs exhibit better hardware efficiency than other recent 128-bit-key block ciphers. Resistance against side-channel attacks was also considered as a design criteria for ICEBERG.
The results presented in this chapter have originally been published in [161, 160].

VII.1. Introduction

There are basically two opposite paradigms regarding algorithms implementation. On the one hand, general-purpose processors can be (re-) programmed at will using a set of instructions from an *assembly language*; by changing the instructions, the functionality of the system is modified without changing the hardware. We talk about *software* computing. On the other hand, hardware circuits are designed and manufactured once and for all to perform a given computation. A typical example is the ASIC (**A**pplication **S**pecific **I**ntegrated **C**ircuit) technology. We talk about *hardware* computing. The high specialization of hardware circuits makes them much faster to accomplish a given task than software-programmed microprocessors. But the drawback is that any desired change in their functionality requires redesign and remanufacturing of the chip.

Halfway between these opposites, reconfigurable hardware tries to combine the high performance of the hardware approach with the flexibility

of software programming. Among them, FPGAs (**F**ield **P**rogrammable **G**ate **A**rray) were introduced in the mid-1980's. They are basically an array of simple computational elements whose functionality is determined by programmable configuration bits, which are stored in volatile memories. While they are significantly slower than "classical" dedicated hardware, FPGA implementations can perform up to 500 times faster than software implementations.

Cryptographic algorithms are destined to be implemented on a wide range of platforms: low-cost 8-bit architectures as those of smart cards, personal computers, or different types of hardware architectures. It is why flexibility, i.e. ability to perform well on various platforms, is often a key argument when selecting a given block cipher as the next standard. A good example is the selection of the new Advanced Encryption Standard by NIST (National Institute for Standards and Technology) in October 2000. Rijndael was selected to become the new AES for a large part on the basis of its good performances on a number of platforms. However, it appeared that its design was not optimal regarding hardware efficiency (see for example [159, 163]). Its highly expensive S-boxes are a typical bottleneck but the problem of the combination of encryption and decryption in hardware is probably as critical.

While at the beginning FPGAs were mainly considered as a preliminary step in the development of ASICs, they may now be considered as a practical solution for real time processing of multi-Gbps data streams. Video-processing is a typical context where high throughput has to be provided at low hardware cost. But although present encryption algorithms may provide very high encryption rates, it is often at the cost of expensive designs. This motivated us to design a new block cipher, which contrary to the majority is deliberately oriented towards efficiency on a particular platform, namely FPGAs: `ICEBERG`.

This chapter presents the design and security analysis of `ICEBERG`. Although we are not specialists in hardware, in a care of completeness we also describe implementation results for `ICEBERG` and compare them to those obtained for other recent block ciphers. The chapter is organized as follows. Section VII.2 gives more details about structure of FPGAs. Section VII.3 presents the design rationales and specifications of `ICEBERG`. Its security analysis is in section VII.4 and its performance analysis in section VII.5. Finally, comparisons with other recent block ciphers are in section VII.6, and a brief analysis of software efficiency of our cipher in section VII.7. Section VII.8 is the conclusion. Some tables and proofs are given in appendices.

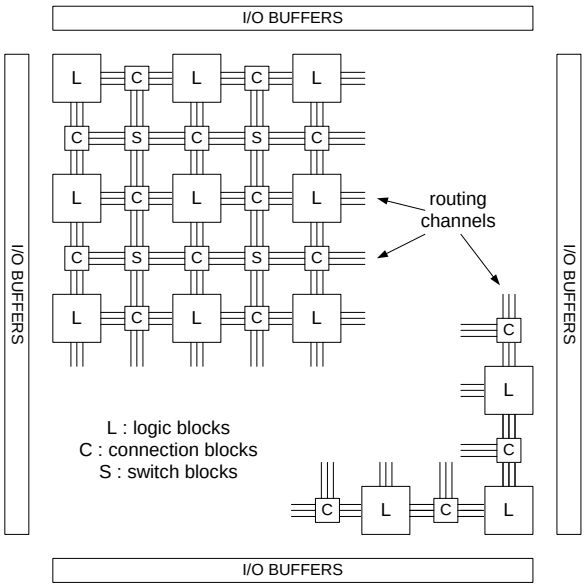


FIGURE 1. FPGA: overall view.

VII.2. FPGA Architectures: an Introduction

FPGAs are mainly constituted of an array of *logic blocks*, which are computational elements of which the functionality can be programmed through configuration bits, stored in volatile memories. These logic blocks are connected through *routing channels* which are programmable as well. Figure 1 represents an overall view of an FPGA. Several possibilities have been studied about the nature of the computational elements that should be implemented into a logic block. It now seems well established that the best component to deploy is the Lookup Table (LUT). An n-input LUT is a multiplexer with n control bits which are used to select amongst the 2^n input bits the one which will be passed to the output. In the case of FPGAs, the 2^n input bits are stored in volatile memories and are thus programmable. Thus an n-input LUT can implement any function from n bit to 1 bit (the control bits being the input). The classical representation of a 2-input multiplexer is pictured in Figure 2 (at left), with c being the control bit and i_0, i_1 the input bits. A *flip-flop* storage element is represented at right of Figure 2.

Besides LUTs, fast access memories of large size are also frequently incorporated in FPGAs. They are called *RAM blocks*.

The implementation of `ICEBERG` described in section VII.5 was done using a Xilinx Virtex-II®. The Virtex-II®'s logic cell is made up of a 4-input LUT, but also a carry (that is very useful for arithmetics, but also permits XOR operations), and a flip-flop. The combination of two

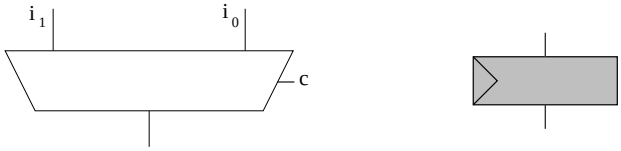


FIGURE 2. A 2-input multiplexer (at left) and a flip-flop (at right).

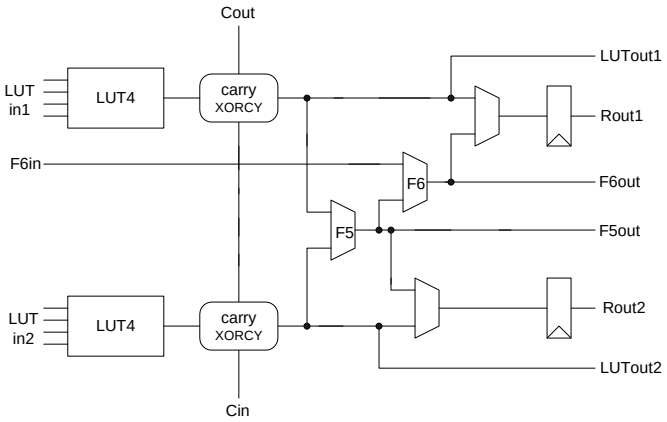


FIGURE 3. The Virtex® slice.

such cells forms a *slice*, and can implement any 5-input function. A Virtex® slice is pictured in Figure 3. Finally the combination of two slices forms a Configurable Logic Block (CLB); it can implement any 6-input function.

Different types of architectures exist regarding hardware implementation of block ciphers. One of them consists in implementing one only round; during the encryption, this round is executed the appropriate number of times. Flip-flops are used to store the intermediate result between two rounds. It is called a ***loop architecture***. A general view of this approach is given in Figure 4[1]. Its advantage is the small area needed to implement the whole cipher; its drawback is its relatively low speed. On the other hand, ***unrolled architectures*** have all their rounds independently implemented (see example in Figure 5). However as such this approach is not really interesting: the speed gain is small in comparison with loop architectures for a much more important area. Nevertheless by inserting registers after each round of the cipher[2], several plaintexts can be processed simultaneously by the hardware (see Figure 6). We speak of ***pipelining***. After a given clock cycle each of the intermediate registers

[1]Note that the figure assumes the first operation of the implemented cipher is a XOR with the key; thus this scheme is not quite general.

[2]Which represents no hardware cost as the logic cell contains such registers.

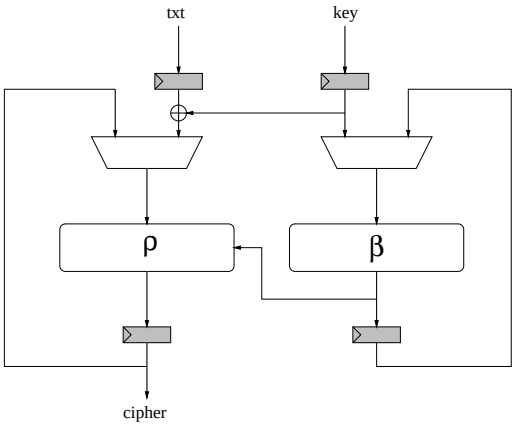


FIGURE 4. Loop architecture for a block cipher. The round is denoted by ρ and the key round by β.

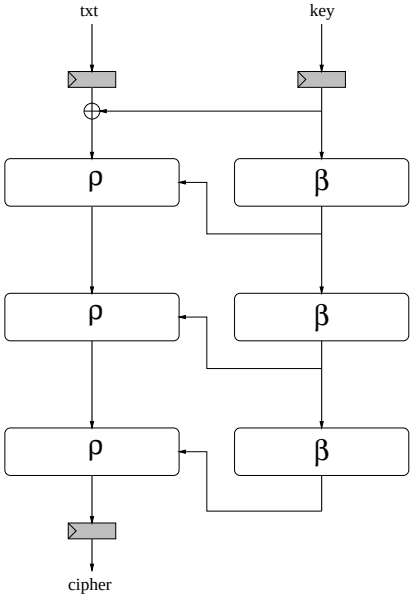


FIGURE 5. Unrolled architecture for a 3-round block cipher. The round is denoted by ρ and the key round by β.

contains data corresponding to a different plaintext. Thus the speed is roughly multiplied by the number of rounds in comparison with previous architectures. However a drawback is that this architecture cannot be used for mode of operations other than ECB. ***Sub-pipelining*** is similar to pipelining but also inserts registers inside the round functions.

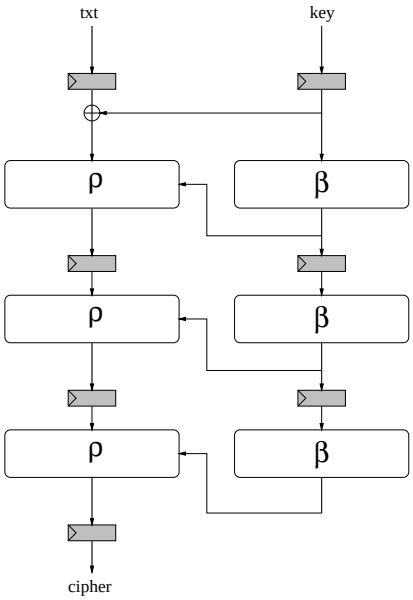


FIGURE 6. Unrolled architecture with pipelining for a 3-rounds block cipher. The round is denoted by ρ and the key round by β.

Remark that all these architectures imply that the key schedule is re-computed at each encryption, contrary to the case of software implementations. We speak of ***on-the-fly*** key derivation.

Taking into account the specificity of hardware implementation and the structure of the slice, we can define two notions of efficiency for the implementation of a block cipher:

(1) In terms of performances, the efficiency is the ratio *Throughput (Mbits/sec) / Area (# slices)*.
(2) In terms of resources, the efficiency is the ratio *Nbr of registers / Nbr of LUTs*. This ratio measures how optimally the hardware resources are used in the implementation. A ratio of 1 would mean that in each slice used, both the LUTs and the registers are in use. Thus the actual ratio should be close to 1.

VII.3. Design Rationale and Specifications of ICEBERG

Recent submissions to the NESSIE project [126], such as KHAZAD [8] or Misty [114], provide an improved hardware efficiency [162]. They also allow interesting comparisons between Feistel ciphers (e.g. Misty) and SP-networks, with respect to hardware efficiency [162]. Although KHAZAD is not a Feistel network, its structure is designed so that by choosing all components to be involutions, the inverse operation of the

cipher differs from the forward operation in the key scheduling only. However the flaw of KHAZAD regarding hardware implementation is that key derivation in decryption mode requires preliminary computation of the whole key schedule in encryption mode, and storage of the result, before beginning decryption. This is particularly damaging when the application requires frequent key changes or frequent switches between encryption and decryption.

ICEBERG is based on an involutional structure like KHAZAD but its key schedule allows direct on-the-fly key derivation in both encryption and decryption modes. This involves no storage requirements for the round keys. The design goals for ICEBERG were:

- Good security properties: ICEBERG's resistance to known attacks is comparable to recently published block ciphers (AES, NESSIE [126]).
- Easiness of hardware implementation.
- Hardware implementation efficiency, as defined in section VII.2.
- Hardware implementation versatility: ICEBERG is scalable for different architectures (loop, unrolled, pipeline) and FPGA technologies. ASIC implementations would be efficient as well.
- Resistance against side-channel attacks: Small substitution tables are used in order to allow efficient Boolean masking [40, 119]. Moreover, the key agility offers the opportunity to consider new encryption modes where the key is changed frequently in order to make the averaging of side-channel traces unpractical.

The complete specifications of ICEBERG follow.

VII.3.1. Block and Key Size

Let n be the block bit-size and k be the key bit-size. The state x is represented as a n-bit vector where $x(i)$ $(0 \leq i < n)$ represents the i^{th} bit from the right. Alternatively, x can be represented as an array of $\frac{n}{4}$ 4-bit blocks, where x_j is the j^{th} block from the right. ICEBERG operates on 64-bit blocks and uses a 128-bit key. It is an involutional iterative block cipher based on the repetition of R identical key-dependent round functions.

VII.3.2. The Non-Linear Layer γ

Function γ consists of the successive application of non-linear S-boxes and bit permutations (i.e. wire crossings).

VII.3.2.1. *Substitution layers S_0, S_1:* The substitution layers S_j consist in the parallel application of S-boxes s_j to the blocks of the state.

$$S_j : \mathbb{Z}_{2^4}^{16} \to \mathbb{Z}_{2^4}^{16} : x \to y = S_j(x) \Leftrightarrow y_i = s_j(x_i) \quad 0 \leq i \leq 15$$

The tables of S-boxes s_0, s_1 are given in Appendix D.

VII.3.2.2. *Bit permutation layer $P8$:* The permutation layer $P8$ consists in the parallel application of 8 permutations $p8$ to the state, where $p8$ consists in bit permutations on 8-bit blocks of data. The table of $p8$ is given in Appendix D.

$$P8 : \mathbb{Z}_{2^8}^{8} \to \mathbb{Z}_{2^8}^{8} : x \to y = P8(x) \Leftrightarrow y(8i+j) = x(8i + p8(j)) \\ 0 \leq i \leq 7,\ 0 \leq j \leq 7$$

Based on previous descriptions, the non-linear layer γ can be expressed as

$$\gamma : \mathbb{Z}_2^{64} \to \mathbb{Z}_2^{64} : \gamma = S_0 \circ P8 \circ S_1 \circ P8 \circ S_0.$$

For cryptanalytic and software implementation purposes, γ may also be viewed as a unique layer consisting in the application of 8 identical 8×8 S-boxes of which the table is given in Appendix D.

VII.3.3. The Key Addition Layer σ_K

The affine key addition σ_K consists in the bitwise exclusive or of a key vector K.

$$\sigma_K : \mathbb{Z}_2^{64} \to \mathbb{Z}_2^{64} : x \to y = \sigma_K(x) \Leftrightarrow y(i) = x(i) \oplus K(i) \quad 0 \leq i \leq 63$$

VII.3.4. The Linear Layer ϵ_K

Function ϵ_K consists in the successive application of binary matrix multiplications and wire crossing layers, combined with the key addition layer for efficiency purposes. It is defined as

$$\epsilon_K : \mathbb{Z}_2^{64} \to \mathbb{Z}_2^{64} : \epsilon_K = P64 \circ P4 \circ \sigma_K \circ M \circ P64,$$

where M, $P64$, and $P4$ are as follows:

VII.3.4.1. *Matrix multiplication layer M:* The matrix multiplication layer M is based on the parallel application of a simple involutional matrix multiplication. Let $V \in \mathbb{Z}_2^{4\times4}$ be a binary involutional (i.e. such that $V^2 = I_n$) matrix:

$$V = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

M is then defined as

$$M : \mathbb{Z}_{2^4}^{16} \to \mathbb{Z}_{2^4}^{16} : x \to y = M(x) \Leftrightarrow y_i = V \cdot x_i, \quad 0 \leq i \leq 15.$$

We define diffusion boxes D as performing multiplication by V. The table of D is given in Appendix D.

VII.3.4.2. *Bit permutation layer $P64$:* Permutation $P64$ performs bit permutations on 64-bit blocks of data.

$$P64 : \mathbb{Z}_2^{64} \to \mathbb{Z}_2^{64} : x \to y = P64(x) \Leftrightarrow y(i) = x(P64(i)) \quad 0 \leq i \leq 63$$

The table of permutation $P64$ is given in Appendix D.

VII.3.4.3. *Bit permutation layer $P4$:* The permutation layer $P4$ consists in the parallel application of 16 permutations $p4$ to the state. $p4$ consists in bit permutations on 4-bit blocks of data. The table of $p4$ is given in Appendix D.

$$P4 : \mathbb{Z}_{2^4}^{16} \to \mathbb{Z}_{2^4}^{16} : x \to y = P4(x) \Leftrightarrow y_i(j) = x_i(p4(j)) \quad 0 \leq i \leq 15,\ 0 \leq j \leq 3$$

The purpose of permutation $P4$ is to efficiently distinguish encryption from decryption. It will become clearer in section VII.3.8 and Appendix C.

VII.3.5. The Round Function ρ_K

Finally, the whole round function can be expressed as

$$\rho_K : \mathbb{Z}_2^{64} \to \mathbb{Z}_2^{64} : \rho_K = \epsilon_K \circ \gamma.$$

It is illustrated in Figure 7.

VII.3.6. The Key Schedule

The key scheduling process consists of key expansion and key selection.

VII.3.6.1. *The key expansion:* This process expands the cipher key $K \in \mathbb{Z}_2^{128}$ into a sequence of keys $K^0, K^1, ..., K^R$ also $\in \mathbb{Z}_2^{128}$. We set the initial key $K^0 = K$. Then we expand K^0 by a simple key round function β_C so that

$$K^{i+1} = \beta_C(K^i),$$

where $0 \leq i \leq R$ and $C \in \mathbb{Z}_2$ is a round constant discussed in section VII.3.9.

The **key round** β_C is pictured in Figure 8. It consists in the application of non-linear S-boxes, shift operations and bit permutations:

$$\beta_C : \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128} : \beta_C = \tau_C \circ P128 \circ S' \circ P128 \circ \tau_C,$$

where τ_C, S', and $P128$ are defined as follows.

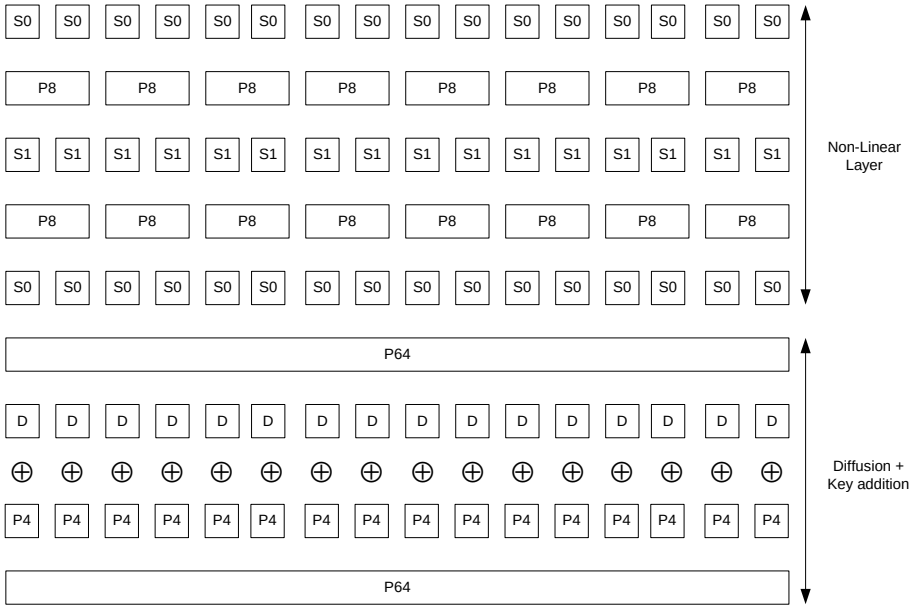


FIGURE 7. The round function ρ_K.

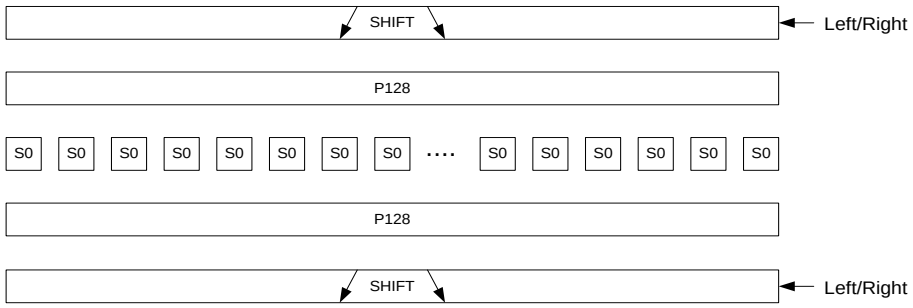


FIGURE 8. The key round β_C.

- **Shift layer τ_C:** The shift layer τ_C consists in the application of a variable shift operator to the bytes of the key : shift left if $C = 1$, shift right if $C = 0$.

  $\tau_C : \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128} : x \to y = \tau_C(x) \Leftrightarrow$

  if $C = 0$: $y(i) = x((i + 8) \mod 128) \quad 0 \le i \le 127$

  if $C = 1$: $y(i) = x((i - 8) \mod 128) \quad 0 \le i \le 127$

- **Substitution layer S':** The substitution layer S' consists in the parallel application of S-boxes s_0 to the blocks of the key.

  $S' : \mathbb{Z}_{2^4}^{32} \to \mathbb{Z}_{2^4}^{32} : x \to y = S'(x) \Leftrightarrow y_i = s_0(x_i) \quad 0 \le i \le 31$

  The table of S-box s_0 is given in Appendix D.

- **Bit permutations layer** $P128$**:** $P128$ performs bit permutation on 128-bit blocks of data.

$$P128 : \mathbb{Z}_2^{128} \to \mathbb{Z}_2^{128} : x \to y = P128(x) \Leftrightarrow y(i) = x(P128(i)) \quad 0 \le i \le 127$$

The table of $P128$ is given in Appendix D.

VII.3.6.2. *The key selection:* From every 128-bit vector K^i, we first apply a simple compression function E that selects 64 bits corresponding to key bytes of K^i having odd indices. We denote the resulting key as $K64^i$. Then we apply a **key selection layer** (ϕ) that consists in the parallel application of a selection function X to the blocks of the key.

$$\phi_{sel} : \mathbb{Z}_{2^4}^{16} \to \mathbb{Z}_{2^4}^{16} : K64^i \to RK_{sel}^i = \phi_{sel}(K64^i) \Leftrightarrow RK_{sel,j}^i = X_{sel}(K64_j^i) \quad 0 \le j \le 15$$

The **selection function** X takes 4-bit inputs and a selection bit sel:

$$X_{sel} : \mathbb{Z}_{2^4} \to \mathbb{Z}_{2^4} : x \to y = X_{sel}(x) \Leftrightarrow \begin{cases} y(0) = (x(0) \oplus x(1) \oplus x(2)) \cdot sel \vee (x(0) \oplus x(1)) \cdot \overline{sel} \\ y(1) = (x(1) \oplus x(2)) \cdot sel \vee x(1) \cdot \overline{sel} \\ y(2) = (x(2) \oplus x(3) \oplus x(0)) \cdot sel \vee (x(2) \oplus x(3)) \cdot \overline{sel} \\ y(3) = (x(3) \oplus x(0)) \cdot sel \vee x(3) \cdot \overline{sel} \end{cases} \quad (102)$$

This selection process is represented in Figure 9. As a result, we obtain a 64-bit round key denoted by RK_1^i if $sel = 1$ and RK_0^i if $sel = 0$.

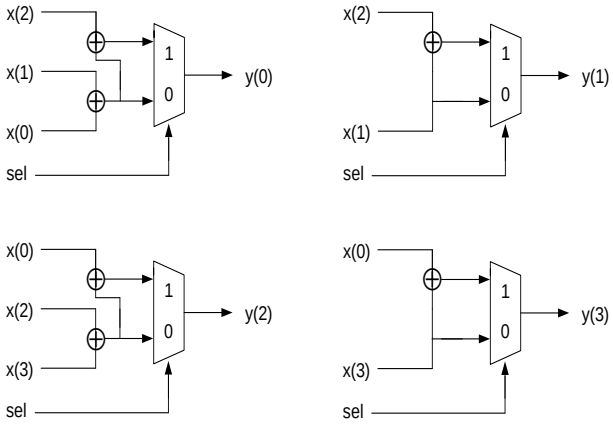


FIGURE 9. The key selection function X_{sel}.

VII.3.7. Encryption Process

ICEBERG is defined for the cipher key K, as the transformation $\texttt{ICEBERG}[K] = \alpha_R[RK_1^0, RK_1^1, .. ., RK_0^R]$ applied to the plaintext where

$$\alpha_R[RK_1^0, RK_1^1, ..., RK_0^R] = \sigma_{RK_0^R} \circ \gamma \circ (\bigcirc_{r=1}^{R-1} \rho_{RK_1^r}) \circ \sigma_{RK_1^0}.$$

The standard number of rounds is $R = 16$.

VII.3.8. Decryption Process

We now show that ICEBERG is an involutional cipher in the sense that the only difference between encryption and decryption is in the key schedule. We will need the following theorem, proved in Appendix C:

THEOREM 34. *For any* $K64 \in \mathbb{Z}_{2^4}^{16}$*:* $\epsilon_{RK_0}^{-1} = \epsilon_{RK_1}$*, where* $RK_0 = \phi_0(K64)$ *and* $RK_1 = \phi_1(K64)$*.*

Then, the decryption process can be obtained as follows. We start from the encryption process

$$\alpha_R[RK_1^0, RK_1^1, ..., RK_0^R] = \sigma_{RK_0^R} \circ \gamma \circ (\bigcirc_{r=1}^{R-1} \epsilon_{RK_1^r} \circ \gamma) \circ \sigma_{RK_1^0}.$$

Then we have for decryption

$$\alpha_R^{-1}[RK_1^0, RK_1^1, ..., RK_0^R] = \sigma_{RK_1^0} \circ (\bigcirc_{r=R-1}^{1} \gamma \circ \epsilon_{RK_1^r}^{-1}) \circ \gamma \circ \sigma_{RK_0^R}$$
$$\Leftrightarrow \alpha_R^{-1}[RK_1^0, RK_1^1, ..., RK_0^R] = \sigma_{RK_1^0} \circ \gamma \circ (\bigcirc_{r=R-1}^{1} \epsilon_{RK_1^r}^{-1} \circ \gamma) \circ \sigma_{RK_0^R}.$$

Finally the above theorem leads to

$$\alpha_R^{-1}[RK_1^0, RK_0^1, ..., RK_0^R] = \sigma_{RK_1^0} \circ \gamma \circ (\bigcirc_{r=R-1}^{1} \epsilon_{RK_0^r} \circ \gamma) \circ \sigma_{RK_0^R}.$$

VII.3.9. Round Constants

ICEBERG is an involutional cipher in the sense that the only difference between encryption and decryption is in the key schedule. Moreover, if properly chosen, the round constants allow us to compute keys "on-the-fly" in encryption and decryption modes. Basically, we would like round keys to satisfy:

$$K^0 = K^R, \quad K^1 = K^{R-1}, \quad K^2 = K^{R-2}, \quad ... \tag{103}$$

This implies that R is even. Then, if the first half of round constants (i.e. until round 8) is 0 (shift left) and the second half is 1 (shift right), the resulting round keys will satisfy Equation (103).

As a consequence, the only difference between encryption and decryption is the selection function ϕ of the key bits, as $\epsilon_{RK_1}^{-1} = \epsilon_{RK_0}$.

VII.4. Security Analysis

VII.4.1. Design Properties of the Components

VII.4.1.1. *S-boxes:* The non-linear layer γ may be viewed as made out of the parallel application of 8 copies of the same 8×8 S-box. We designed this S-box such that it has the following properties:

- It is an involution.
- Its δ-parameter is 2^{-5}.
- Its λ-parameter is 2^{-2}.
- Its nonlinear order ν (see [96]) is maximum, namely 7.

For efficiency purposes, the S-box was generated from a fixed permutation $p8$ and small 4×4 S-boxes s_0 and s_1 that perfectly fit into 4-input LUTs. The generation of the S-boxes is detailed in Appendix E.

VII.4.1.2. *The bit permutations:* $P64$ and $P128$ were designed such as to disturb the bit alignment inside bytes as much as possible, in order to provide resistance against some attacks. A remarkable property of $P64$ and $P128$ is that 2 bits from the same byte are always mapped to 2 bits belonging to different bytes. $p8$ is involutional and allows generating good S-boxes. Finally, $p4$ allows the selection function X_{sel} to be implemented in one LUT.

VII.4.1.3. *The diffusion layer:* Due to the fact that we attached much importance to hardware implementation aspects in the design of the diffusion layer, it is not optimal. More precisely, it is easy to see that its byte branch number is 4, as the bit branch number of $p4 \circ \sigma_K \circ D$ is 4, and because of the remarkable property of $P64$ we have just mentioned. The diffusion boxes D were designed so that their combination with the key addition layer σ_K can be done inside one LUT.

VII.4.1.4. *The key round:* The key round has been chosen for its efficiency properties, as well as in order to provide resistance against key schedule cryptanalysis and slide attacks:

(1) Non periodicity is provided by the shift operation τ_C.
(2) Non linearity is provided by non-linear S-boxes.
(3) Good diffusion properties are provided by the combination of shifts, S-boxes and bit permutations.

Moreover, the shift layer τ_C is used in order to allow the property (103) to be satisfied. The selection function X_{sel} is necessary to prove the property of Appendix C and is designed such that it fits into a single LUT.

VII.4.2. Strength Against Known Attacks

VII.4.2.1. *Linear and differential cryptanalysis:* From the properties of the S-box and the diffusion layer, we can compute that a differential characteristic over 2 rounds of ICEBERG has probability at most $(2^{-5})^4 = 2^{-20}$. Also, a linear characteristic over these 2 rounds has input-output correlation at most $(2^{-2})^4 = 2^{-8}$. Therefore loose bounds can be computed for the full cipher (16 rounds):

- The probability of the best differential characteristic is smaller than 2^{-160}.
- The input-output correlation of the best linear characteristic is smaller than 2^{-64}.

The security margin is very likely big enough to prevent variants of differential and linear attacks, such as boomerang [169] and rectangle [18] attacks, multiple linear cryptanalysis [85], non-linear approximations of outer rounds [93], partitioning cryptanalysis [69],... Note also that the security margin of ICEBERG against linear and differential cryptanalysis is comparable to the one of KHAZAD. This is probably more than necessary, as resistance against structural attacks [44] was probably more determinant in the choice of the number of rounds of KHAZAD (8), than security margins against linear and differential cryptanalysis.

VII.4.2.2. *Truncated and impossible differentials:* Truncated differentials were introduced in [96], and impossible differentials in [17, 16]. They typically apply to ciphers operating on well-aligned data blocks (often bytes), such as KHAZAD or the AES (and many others). But our cipher does not enter in this category because of the $P64$ layer, which makes it very difficult to attack this way. Therefore we do not expect such distinguisher to be found on more than 5 or 6 rounds of ICEBERG.

VII.4.2.3. *Square attacks:* We have seen in Chapter III (and more specifically, in section III.10.2) that square attacks [44] only apply to ciphers operating on well-aligned data blocks. Therefore the $P64$ layer should prevent them efficiently, at least on more than a few rounds.

VII.4.2.4. *Interpolation attacks:* Interpolation attacks [79] are made possible when the S-box has a simple algebraic structure, allowing us to express the cipher as a sufficiently simple polynomial or rational expression. The diffusion layer also has a role with this respect. As the S-box of ICEBERG has no simple algebraic expression, it prevents interpolation attacks for more than a few rounds of our cipher.

VII.4.2.5. *Higher order differential cryptanalysis:* It was introduced by L. Knudsen in [96], and relies on finding high order differentials being a constant for the whole cipher. But as the nonlinear order of the S-box we use is maximal, namely 7, we can expect that the maximal value of

63 for the non-linear order of the cipher is reached after a few rounds of ICEBERG.

VII.4.2.6. *Slide attacks:* Slide attacks [25, 26][3] work against ciphers using a periodic key schedule. Although the sequence of subkeys produced by the key schedule of ICEBERG is not periodic, it has a particular structure, namely

$$(K^0, K^1, ..., K^7, K^8, K^7, ..., K^0).$$

The key schedule of the GOST cipher has some similarities with the one of ICEBERG. Vulnerability of some variants and reduced-round versions of GOST against slide attacks are examined in [26]. However none of the attacks presented there seems to be applicable to our cipher.

VII.4.2.7. *Related-key attacks:* The first related-key attack has been described in [13][3], and is the related-key counterpart of the slide attack. Let us examine a slightly simplified version of ICEBERG, where the initial key addition $\sigma_{RK_1^0}$ is replaced by a normal round $\rho_{RK_1^0}$, and the final $\sigma_{RK_0^R} \circ \gamma$ is also replaced by a normal round $\rho_{RK_0^R}$. Then if 2 keys K and K^* are such that $K^1{=}K^{*0}$, and 2 plaintexts P and P^* are such that $P^* = \rho_{RK_1^0}(P)$, encryption of P under K and of P^* under K^* will process the same way (with a shift of 1 round) during 8 rounds. However the round keys, and hence the computation, will then differ. Therefore such a related-key attack does not work against our key schedule. Forgetting the simplification we made on the first and last round of ICEBERG, a related-key attack becomes even more difficult. Differential related-key attacks [90] are also very unlikely to be applicable to ICEBERG, due to the good diffusion and nonlinearity of its key schedule.

VII.4.2.8. *Weak keys:* The design properties of the key round prevent ICEBERG from having weak keys. The only remarkable property of the key round is in the selection function X_{sel} where some symbols are independent of the selection bit. Namely, hexadecimal input symbols $0, 2, 8, A$ become $0, C, 3, F$ regardless of $sel = 0$ or $sel = 1$. However, this point is very unlikely to be an exploitable weakness.

VII.4.2.9. *Algebraic Attacks:* In [42] N. Courtois and J. Pieprzyk described a new attack technique on block ciphers based on describing them as overdefined systems of algebraic equations (holding with probability 1), more specifically quadratic equations. Such system is solved using algorithms such as XL [41] or XSL [42]. This attack requires very few plaintext-ciphertext pairs. [42] discusses its applicability to the AES [47] and Serpent [2]. However XL and XSL are heuristic algorithms, thus their complexity when dealing with large systems is difficult to figure out; it is why there has been a controversy on the real efficiency of this technique.

[3]See also Chapter IV of this thesis.

Analysis of ICEBERG's 8×8 S-box showed that there is no quadratic equation that (partially) describes it [33]. Thus applying an algebraic attack to ICEBERG is probably not feasible.

VII.4.2.10. *Biryukov's observations on involutional ciphers:* Observations of A. Biryukov on KHAZAD and Anubis [21] remain valid for ICEBERG. However this study could at best threaten 5 rounds of our cipher, while it is made out of 16 rounds.

VII.4.2.11. *Side-channel analysis:* Cryptosystem designers frequently assume that secret parameters will be manipulated in closed reliable computing environments. However as already mentioned in Chapter V, P. Kocher et al. stressed in 1998 [101] that actual computers and microchips leak information correlated to the data handled. Side-channel attacks based on time, power and electromagnetic measurements were successfully applied to smart card implementations of block ciphers.

Protecting implementations against side-channel attacks is usually difficult and expensive. Masking all the data with random Boolean values is suggested in several papers [66, 34] and the use of small substitution tables allows implementing this efficiently, although it is still an expensive solution.

Moreover, the key agility provided by ICEBERG (changing the key at every plaintext block is for free) also offers interesting opportunities to prevent most side-channel attacks by defining new encryption modes where the key is changed sufficiently often. As most side-channel attacks need to collect several leakage traces to remove the noise from useful information, changing the key frequently, even in a well chosen deterministic way (e.g. LFSR[4]-based), would make most attacks somewhat unpractical. Actually, only template attacks [35] allow the extraction of information from a single sample but the context is also more specific as they require that an adversary has access to an experimental device (identical to the device attacked) that he can program to his choosing.

VII.5. Performance Analysis

As a consequence of the criteria enumerated at the beginning of section VII.3, ICEBERG has the following properties:

- All its components easily fit in 4-input LUTs. Practically, ICEBERG is made out of the parallel application of 4-input-bit transforms combined with bit permutations and shifts.
- All its components are involutional so that encryption and decryption can be made with the same hardware. The only difference between them is in the selection bit ϕ_{sel}.

[4]LFSR : Linear Feedback Shift Register.

TABLE 1. Number of LUTs required to implement each of the ICEBERG components.

Round Components	HW cost (LUTs)	**Keyround Components**	HW cost (LUTs)
S_0, S_1 layers	64	Shift layer	128
Non-linear layer	$64 \times 3 = 192$	S_0 layer	128
Linear diffusion layer	64	Keyround	384
Round	256	Selection layer	64

- The key expansion allows us to derive keys "on-the-fly" in encryption and decryption modes. There is no need to store the round keys and the key can be changed in one clock cycle.
- The algorithm and its key scheduling are balanced, which means that the round and key round perform in the same number of clock cycles.
- The non-linear layer could be efficiently implemented into the RAM blocks available in most modern FPGAs.

As all components easily fit in 4-input LUTs, we can directly evaluate the combinatorial cost of individual ICEBERG components in the Xilinx Virtex-II® family of devices. The result is shown in Table 1.

Remark that if the maximum pipeline is not inserted, the shift layers can be efficiently implemented inside the Virtex® slice, using additional multiplexers $F5$ and $F6$ available next to the LUT (see section VII.2).

In the next sections, we present the practical implementation of different architectures for ICEBERG. All the architectures proposed allow the choice of the key and E/D mode for every plaintext. The area and frequency estimations presented result from an implementation with Xilinx ISE 6.1 on the Xilinx Virtex-II®technology.

The results from this section come from [160].

VII.5.1. Unrolled Architectures

For high throughput applications, we describe an unrolled implementation with the 16 rounds implemented. Two pipelining strategies were applied. If a maximum throughput is required, a full pipe implementation is provided, with the maximum pipeline inserted. However, for large designs, the implementation (and specially the routing task) may become the bottleneck, with routing delays larger than logic delays. Therefore, for an optimized efficiency, a half pipe architecture is preferable. In addition to a better tradeoff between logic and routing delays, it also allows an efficient implementation of the shift layer, using the

additional multiplexers available inside the Virtex® slice. Both architectures are pictured in Figure 10. Finally, if the half pipe architecture is considered, we can also implement the round S-box inside the FPGA RAM blocks. The implementation results for these three proposals are in Table 2.

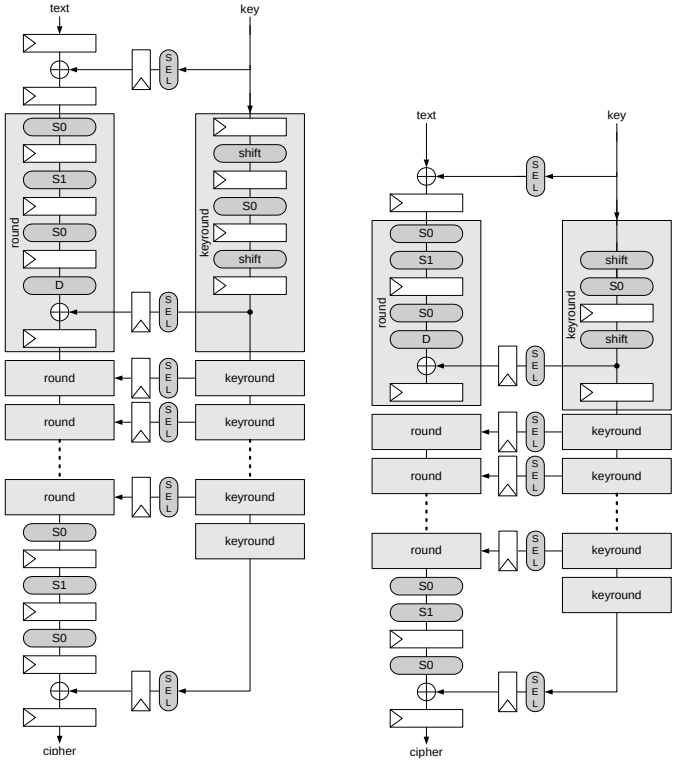


FIGURE 10. Unrolled architectures : full pipe and half pipe.

TABLE 2. Unrolled architectures results on Virtex-II®.

Type	# of slices	# of RAM blocks	Latency (cycles)	Output every (cycles)	Freq. (Mhz)	Throughput (Mbits/sec)
Full Pipe	6808	0	66	1	297	19008
Half Pipe	4946	0	33	1	271	17344
Half Pipe RAM	3132	64	33	1	210	13440

VII.5.2. Loop Architectures

For applications requiring minimum area, a loop architecture with only one round implemented is possible. In order to decrease the area requirements, a half pipe strategy is considered. In addition to the efficiency advantages already mentioned, half pipe structures are specially convenient for loop architectures because they allow the combination of the

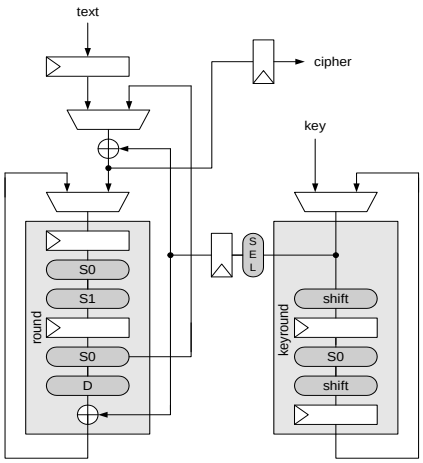


FIGURE 11. Loop architecture.

loop multiplexer with the round and key round logic. The result is pictured in Figure 11. As for unrolled architectures, it is possible to use the FPGA RAM blocks to implement the round S-box. The implementation results for these loop architectures are provided in Table 3.

TABLE 3. Loop architecture results on Virtex-II®.

Type	# of slices	# of RAM blocks	Latency (cycles)	Output every (cycles)	Freq. (Mhz)	Throughput (Mbits/sec)
Loop	631	0	34	2/32	254	1016
Loop RAM	526	4	34	2/32	227	908

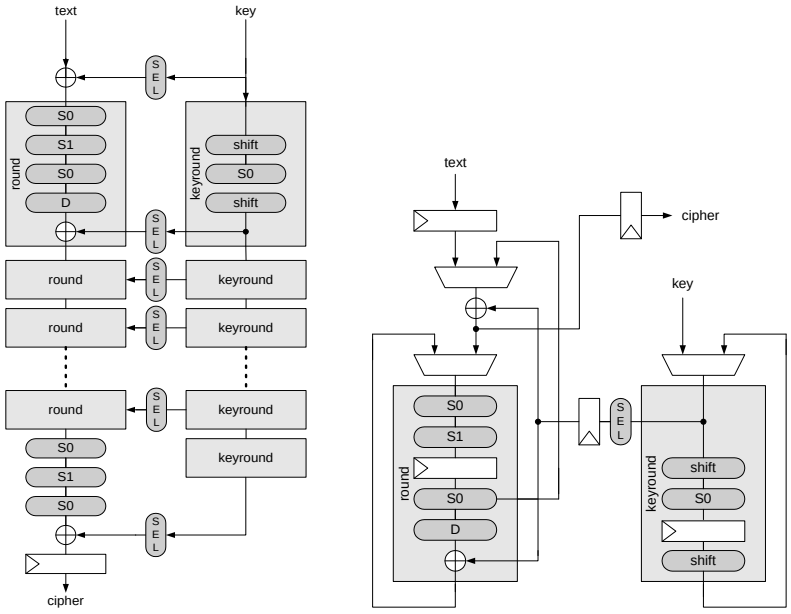


FIGURE 12. Feedback mode : unrolled and loop architectures.

VII.5.3. Feedback Modes

It is mentioned in section VII.2 that as soon as a feedback mode is used, pipelining techniques are not relevant for block cipher implementations. Although we do not recommend the use of feedback modes in FPGA implementations of block ciphers because they do not allow us to take full advantage of hardware efficiency, the following designs are interesting for comparison purposes. An unrolled architecture without pipelining and a minimum latency loop architecture are represented in Figure 12. The implementation results of these designs are in Table 4.

TABLE 4. Feedback mode results on Virtex-II®.

Type	# of slices	# of RAM blocks	Latency (cycles)	Output every (cycles)	Freq. (Mhz)	Throughput (Mbits/sec)
Unrolled	3174	0	1	1	14	896
Loop	571	0	17	1/16	147	588
Loop RAM	467	4	17	1/16	145	580

VII.6. Comparisons with Other Block Ciphers

Comparing the performances of block cipher hardware implementations is generally a delicate task. This is due to the high dependency of these implementation results on the design methodology, but also to the various commercial FPGAs that may be chosen for evaluation. In the case of

ICEBERG, it is even more critical as our implementations provide key **and** E/D agility: two properties that are never combined in other block cipher implementations, except Triple-DES. The following considerations must therefore be taken with care and should be considered as general guidelines more than as a strict comparison.
We tried to find the best results for various block ciphers in non feedback modes, if possible in the most recent technology (Virtex-II®). Then, we provided the area and throughput results. If no RAMBs are used, the ratio Throughput/Area is given in order to estimate the hardware efficiency. We also specified the architecture used (loop or unrolled) and its basic features (encryption only, encryption/decryption, key agility). These results are listed in Tables 5 and 6.
In general, ICEBERG implementations exhibit a significant improvement of the hardware efficiency, even when compared to encryption only designs. It is clear that the most relevant implementation schemes for ICEBERG do not use RAMBs because they considerably increase the S-box memory requirements[5]. LUT-only implementations are also the best estimators for ASIC performances and underline the excellent potentialities of ICEBERG for hardware implementations in general. More specifically, only Rijndael [152] and the Triple-DES have an efficiency comparable to ICEBERG with an E/D structure. However, the specified Rijndael implementation does not provide key and E/D agility, uses RAM blocks and shares resources between the round and key round.

For Triple-DES, it is well known to allow very efficient implementation opportunities. It is thus an excellent result that ICEBERG's efficiency is proved better than the one of Triple-DES. In [151] a fully unrolled and pipelined implementation of DES is given. It achieves throughput up to 21.3 Gbps, and a ratio Throughput/Area up to 7.18. If we transpose these results to Triple-DES (by iterating three such implementations), the throughput is about the same (only the latency changes), but the ratio Throughput/Area drops to about 2.4, which is smaller than the one of our ICEBERG unrolled implementation. Even if we reduce Triple-DES to 40 rounds, in order to have security bounds against linear and differential cryptanalysis comparable to those of ICEBERG[6], the ratio is still smaller.

Note that measuring the throughput is not that significative. As a matter of fact, it is always possible to increase the throughput provided we have enough hardware resources, by implementing the cipher in parallel several times.

---

[5]The ICEBERG S-box memory requirements are : $(2^4 \times 4) \times 6 = 384$ bits. If RAMBs are used, it becomes $2^8 \times 8 = 2048$ bits.

[6]This reasoning is questionable, as a better way of attacking 3-key Triple-DES is to use a meet-in-the-middle attack to retrieve the key in 2^{112} operations. So a better key schedule for Triple-DES, with a similar hardware efficiency, would be necessary.

TABLE 5. Basic features of compared block ciphers implementations.

Algorithm	**Device**	**Enc.**	**Dec.**	**Key ag.**	**Loop/Unr.**
0.22 μm					
Twofish [59]	Virtex®	•		•	U
Serpent [59]	Virtex®	•		•	U
0.18 μm					
Rijndael [163]	Virtex-E®	•		•	U
Camelia [75]	Virtex-E®	•		•	U
KHAZAD [162]	Virtex-E®	•		•	U
Misty1 [162]	Virtex-E®	•		•	U
Rijndael [163]	Virtex-E®	•		•	L
0.15 μm					
RC6 [11]	Virtex-II®	•		•	U
IDEA [12]	Virtex-II®	•		•	U
SHACAL-1 [116]	Virtex-II®	•		•	U
Triple-DES [151]	Virtex-II®	•	•	•	U
ICEBERG	Virtex-II®	•	•	•	**U**
Triple-DES [151]	Virtex-II®	•	•	•	L
ICEBERG	Virtex-II®	•	•	•	**L**
0.15 μm + RAMBs					
Rijndael [152]	Virtex-II®	•	•		L
ICEBERG	Virtex-II®	•	•	•	**L**
Rijndael [70]	Virtex-II®	•	•		L
ICEBERG	Virtex-II®	•	•	•	**U**

TABLE 6. Performances of compared block ciphers implementations.

Algorithm	**# Slices**	**# RAMBs**	**Throughput** (Mbits/sec)	**Thr./Area** (Mbits/sec / #slices)
0.22 μm				
Twofish [59]	21000	0	15200	0.72
Serpent [59]	19700	0	16800	0.85
0.18 μm				
Rijndael [163]	2784	100	11776	-
Camelia [75]	9692	0	6750	0.7
KHAZAD [162]	7175	0	7872	1.10
Misty1 [162]	6322	0	10176	1.61
Rijndael [163]	2524	0	2085	1.17
0.15 μm				
RC6 [11]	7456	0	4800	0.64
IDEA [12]	9793	0	6800	0.69
SHACAL-1 [116]	13729	0	17021	1.24
Triple-DES [151]	604	0	917	1.51
ICEBERG	**4946**	**0**	**17344**	**3.51**
Triple-DES [151]	227	0	326	1.44
ICEBERG	**631**	**0**	**1016**	**1.61**
0.15 μm + RAMBs				
Rijndael [152]	146	3	358	-
ICEBERG	**526**	**4**	**908**	-
Rijndael [70]	$\approx$1125	18	1408	-
ICEBERG	**3132**	**64**	**13440**	-

VII.7. ICEBERG Software Implementations

Software efficiency was not a design goal of ICEBERG. Nevertheless we think it interesting to observe how costly the hardware efficiency goal is with respect to software efficiency. Indeed, although (or because) some components of the cipher were chosen for their very good performance in hardware, they cannot be really efficiently implemented in software. A typical example is bit permutation layers $P64$ and $P128$. Thus we compared software performance of ICEBERG with the one of Rijndael (128-bit block, 128-bit key) and KHAZAD (64-bit block, 128-bit key).

We implemented ICEBERG in C language on a 32-bit processor. Implementation of the round function was done using a table-lookup approach as the one described in [137]: for each of the eight 8×8 S-boxes and every possible input value to it, the resulting output of the round is given in a

TABLE 7. Software performance comparison of AES, KHAZAD, and ICEBERG.

	AES	KHAZAD	ICEBERG
Encryption (MBytes/s)	6.64	4.04	1.46
Decryption (MBytes/s)	6.30	4.04	1.46
Key Sch. (Enc.)(#KS/s)	$526 \cdot 10^3$	$405 \cdot 10^3$	$8 \cdot 10^3$
Key Sch. (Dec.)(#KS/s)	$167 \cdot 10^3$	$199 \cdot 10^3$	$8 \cdot 10^3$

big table (the key addition layer being moved to the end of the round). Table lookups are also used in the key expansion and key selection layers. Note that ICEBERG is also susceptible to be implemented in bitslice mode as suggested in [14].

The machine used for test was a Sun Ultra SPARC III workstation, with a 750 Mhz processor and 1024MB of RAM. Comparison was performed with optimized C-coded versions of KHAZAD and AES. Results are presented in Table 7.

Regarding encryption/decryption, we can see that ICEBERG is about three times slower than KHAZAD, and four time slower than AES. Regarding the key schedule, its performances are near from disastrous[7]. One reason for this is that implementing it using a lookup table based approach requires separate tables for transforms $P128$ and $P128 \circ S'$.

Although no experiment was made, we guess the performance on a smart card with 8-bit processor would be even worse. However, our aim was absolutely not to devise a multi-platform cipher, but rather a dedicated one, devoted to specific applications. Therefore its low performance on other platforms is not really a surprise.

If a software-efficient key schedule is wanted, an alternative key round based on a small Feistel structure can be used, illustrated in Figure 13. We just use a conditional switch of the two 64-bit vectors so that we can

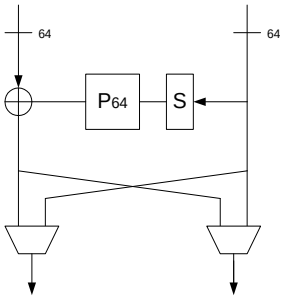


FIGURE 13. A modified ρ_K.

[7]Note however that the code used was not fully optimized.

encrypt during half the rounds and decrypt afterwards in order to satisfy Equation (103). This will only slightly affect hardware performances (an additional multiplexor is necessary to select the round keys).

VII.8. Conclusions

This chapter presented the platform-specific encryption algorithm `ICEBERG` and the rationale behind its design. `ICEBERG` is based on a fast involutional structure in order to provide very efficient hardware implementation opportunities. We showed the specificity of this type of platform in block cipher design. We underlined that the overall structure of a cipher is important for efficiency purposes (for example, in designing rounds and key rounds that can be made in the same number of clock cycles, or in allowing "on-the-fly" key derivation in both encryption and decryption modes). Also, it appeared that optimizing for hardware induces a complete loss of software efficiency; nevertheless, we believe that some applications require such specialized cipher. We believe `ICEBERG` to be as secure as AES and NESSIE [126] candidates and much more efficient for reconfigurable hardware implementations. `ICEBERG` also offers free opportunities to defeat most side-channel attacks by using adequate encryption modes.

Conclusion and Open Problems

In this thesis we contributed to various topics regarding block cipher analysis:

In Chapter II we proved security bounds on the structure underlying the Misty block cipher, where the round functions are involutions without fixed points. To the best of our knowledge, it is the first time involutions are considered in the framework of the Luby-Rackoff model.
Several open problems exist regarding this model: on the one hand, existing bounds on the already studied structures can be improved, possibly by adding more rounds. On the other hand, many structures remain to be studied. But paradoxically the fact that the model only deals with distinguishers with unbounded computation capabilities is definitely a limitation. Thus, despite other attempts such as those of Vaudenay [167, 168], proving the security of block ciphers remains an important open problem, which is probably very difficult. Therefore security analysis of block ciphers is essentially about proving security against known attacks. Chapters III and IV of the thesis were devoted to analyzing classes of attacks.

In Chapter III we dealt with the square attack [44]. Although this attack is known since 1997, little theory exists about it. We tried to give a good formalization of it, and showed a link with another attack: the truncated differential cryptanalysis. Applications to Skipjack and SAFER++ were also given.
It is often the case that a general concept of attack (such as linear and differential cryptanalysis) gives rise to several variations. It is why we suggested possible directions to extend the square attack; they do not seem to be successful, and we do believe that most block ciphers cannot be threatened by them. However we point out that we only have a strong feeling about this, but no security proof...

In Chapter IV we discussed attacks focusing on the key schedule of a block cipher. Among them, related-key attacks work under the particular hypothesis that plaintext-ciphertext pairs are available under several different keys. While at first sight this hypothesis may look strange, there are practical contexts in which it can be satisfied. Among other things, we pointed out mistakes in attacks of the literature. We also discussed application of this type of attacks to multiple encryption modes.

In Chapter V we dealt with a completely different way of attacking block ciphers, which requires ability to tamper with the correct running of the algorithm: fault attacks. We showed how devastating a fault attack can be against unprotected implementations of block ciphers: it may require very few plaintext-cipher pairs.
Our attack exploited faults occurring during the last two rounds of the algorithm. The possibility and the efficiency of an attack exploiting faults occurring during the middle rounds of the cipher is not clear. A proof of such security result is probably bound to resistance against differential cryptanalysis. This is an open problem which is of practical importance, as its solution would indicate which rounds of the cipher should be protected.

Scrambling functions are intended to protect against another type of side-channel attacks, namely probing attacks. In Chapter VI we analyzed the security of the scrambling function $DeKaRT$ using linear cryptanalysis. The conclusion was that the design paradigm behind it is questionable. Our feeling is that classical paradigms of block cipher design are more promising; designing a block cipher efficient in hardware in accordance with these paradigms is the topic of Chapter VII. More importantly, an open problem regarding scrambling functions is to define adequate security criteria for them, knowing that these are much lighter than for classical block ciphers.

In Chapter VII we designed a block cipher which is deliberately oriented towards efficiency in hardware, while most block ciphers are intended to perform equally well on all platforms. Among other things, it allows encryption and decryption using exactly the same hardware and "on-the-fly" key derivation in both encryption and decryption modes. This last characteristic could be used to define encryption modes in which the key is changed very frequently in order to counter side-channel attacks. Two tendencies exist regarding block cipher design: either using strong rounds iterated a small number of times (a typical example is the KHAZAD cipher [8]), or using weak ones iterated a big number of times (an example is Serpent [2]). Aiming at hardware efficiency naturally implied to use the second approach, as the linear transform must be relatively weak in order to be efficient.

Designing and implementing a secure and efficient block cipher requires to consider many issues. Security of the primitive itself is not sufficient: practical implementations remain vulnerable to side-channel attacks. Moreover, the current state of the art regarding cryptanalysis allows a designer to prove security against some known attacks only. It is sometimes the case that one cannot prove security against a given (known) attack, except by trying to apply the attack and argue that it does not seem to work; an example is the square attack. Thus, while

a competent design strongly reduces the possibility of attacks, the fact that an algorithm has been subject to intensive cryptanalytic effort is surely an additional guarantee of its strength. It is why we expect additional cryptanalytic effort against our `ICEBERG` cipher to take place.

The current state of the art regarding block cipher cryptanalysis is a bit particular: on the one hand, there has not been any really new attack discovered during the last five years, if we except the controversial attack of Courtois [42]. Moreover, any serious block cipher designer knows how to protect against the known attacks. On the other hand, there has not been any significant progress regarding security proofs of block ciphers either: while research is still going regarding Luby-Rackoff security proofs, this model has limitations and is therefore not sufficient for complete security proofs of block ciphers. Whether the cryptographic community will reach a real understanding of block cipher security within the next ten years is an interesting as well as difficult question.

APPENDIX A

Publication List

(1) With NESSIE Members. Comments by the NESSIE Project on the AES Finalists. NESSIE Tech. Rep., May 2000.

(2) M. Ciet, F. Koeune, G. Piret, J.-J. Quisquater, and F. Sica. Methodology for comparing the performances of primitives on a fair and equal basis. NESSIE Tech. Rep., October 2000.

(3) M. Ciet, G. Piret, and J.-J. Quisquater. Several Optimizations for Elliptic Curves Implementation on Smart Card. Technical Report CG-2001/1, UCL Crypto Group, 2001. Available at `http://www.uclcrypto.org/tech_reports/CG2001_1.ps.gz`.

(4) G. Piret and J.-J. Quisquater. Impossible differential and square attacks: Cryptanalytic link and application to Skipjack. Technical Report CG-2001/4, UCL Crypto Group, 2001. Available at `http://www.uclcrypto.org/tech_reports/CG2001_4.ps.gz`.

(5) M. Ciet, G. Piret, and J.-J. Quisquater. A Survey of Key Schedule Cryptanalysis. Technical Report CG-2002/1, UCL Crypto Group, 2002. Available at `http://www.uclcrypto.org/tech_reports/CG2002_1.zip`.

(6) G. Piret, M. Ciet, and J.-J. Quisquater. Related Key and Slide Attacks: Analysis, Connections, and Improvements. In *Proceedings of the 23rd Symposium on IT in Benelux*, 2002.

(7) G. Piret and J.-J. Quisquater. Integral Cryptanalysis on reduced-round Safer++ . Technical Report 033, IACR eprint archive, 2003. Available at `http://eprint.iacr.org/2003/033/`.

(8) M. Ciet, G. Piret, and J.-J. Quisquater. A Structure of Block Ciphers achieving some Resistance against Fault Attacks. In *Proceedings of the 24th Symposium on IT in Benelux*, 2003.

(9) F.-X. Standaert, G. Rouvroy, G. Piret, J.-J. Quisquater, and J.-D. Legat. Key-Dependent Approximations in Linear Cryptanalysis. In *Proceedings of the 24th Symposium on IT in Benelux*, 2003.

(10) F.-X. Standaert, G. Piret, and J.-J. Quisquater. Cryptanalysis of Block Ciphers: A Survey. Technical Report CG-2003/2, UCL Crypto Group, 2003. Available at `http://www.uclcrypto.org/tech_reports/CG-2003-2.pdf`.

(11) G. Piret and J.-J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In C.D. Walter, Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, Cologne, Germany, September 8-10, 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer-Verlag, 2003.

(12) F.-X. Standaert, G. Piret, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. ICEBERG : an Involutional Cipher Efficient for Block Encryption on Reconfigurable Hardware. In B.K. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 279–299. Springer-Verlag, 2004.

(13) G. Piret, F.-X. Standaert, G. Rouvroy, and J.-J. Quisquater. On the Security of the *DeKaRT* Primitive. In J.-J. Quisquater, P. Paradinas, Y. Deswarte, and A.A. El Kalam, editors, *Smart Card Research and Advanced Applications VI - 18th IFIP World Computer Congress*, pages 241–254. Kluwer Academic Publishers, August 2004.

(14) G. Piret and J.-J. Quisquater. Security of the MISTY Structure in the Luby-Rackoff Model: Improved Results. In H. Handschuh and A. Hasan, editors, *Selected Areas in Cryptography, 11th Annual International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

(15) F.-X. Standaert, G. Piret, G. Rouvroy, and J.-J. Quisquater. FPGA Implementations of the `ICEBERG` Block Cipher. Accepted at ITCC 2005.

APPENDIX B

Description of Algorithms

In this section we describe the AES and DES algorithms. The aim is to facilitate understanding of chapters which deal with them, rather than to give a complete specification. So some details and a few tables are omitted, as well as the key schedule of the AES. Complete specifications can be found in [46, 47] for the AES, and in [117, 164] for DES.

B.1. AES

The AES [47] (previously Rijndael) is a Substitution-Permutation Network. It means that its round function consists of three different layers:

- A non-linear layer γ. It is made out of the parallel application of identical 8×8 S-boxes.
- A round key addition layer $\sigma[k]$ performed using XOR ($\oplus$): $\sigma[k](x) = x \oplus k$.
- A diffusion layer θ, linear with respect to $\oplus$.

Its block size is 128 bits[1], while its key size can be 128, 192, or 256 bits. The number of rounds R is 10, 12, or 14 depending on the key size. The AES algorithm can be globally represented as

$$\sigma[k^R] \circ \texttt{ShiftRows} \circ \gamma_R \circ \left(\bigcirc_{r=1}^{R-1} \sigma[k^r] \circ \theta_r \circ \gamma_r \right) \circ \sigma[k^0].$$

The `ShiftRows` transform is described below.

TABLE 1. The state during AES encryption.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

[1]Note that the original Rijndael submission allows block sizes of 192 and 256 bits as well, but only the 128-bit version is standardized.

TABLE 2. The `ShiftRows` transformation.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$\xrightarrow{\texttt{ShiftRows}}$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

In the remaining of this description, we deal with the 128-bit block 128-bit key version only. Other versions are quite similar. In this case, the intermediate computation results, called *state*, are usually represented by a 4×4 square, each cell of which is a byte (see Table 1); note that the round key bytes are indexed similarly: thus $k^r_{i,j}$ denotes the byte of k^r that will be XORed with the current $a_{i,j}$. The θ layer (identical for all rounds) is the composition of two transformations of the state:

(1) First, the `ShiftRows` transformation consists in shifting cyclically the rows of the state. Row 0 is not shifted, row 1 is shifted by 1 byte, row 2 is shifted by 2 bytes, and row 3 by 3 bytes. It is pictured in Table 2.
(2) Then, the `MixColumns` transformation applies a linear transformation with optimal byte branch number (i.e. 5) to each column of the state. More precisely, application of `MixColumns` to the first column of the state (for example) is computed by

$$\begin{bmatrix} b_{0,0} \\ b_{1,0} \\ b_{2,0} \\ b_{3,0} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} a_{0,0} \\ a_{1,0} \\ a_{2,0} \\ a_{3,0} \end{bmatrix}$$

where multiplication is performed in $\mathrm{GF}(2^8)$, via definition of an irreducible polynomial of degree 8 over $\mathrm{GF}(2)$ (see [47] for details).

It is easy to see that the byte branch number of θ is 5. Note that `ShiftRows` is applied during the last round, while `MixColumns` is not. As a matter of fact, applying a linear transform just before the last key addition layer does not make sense. However `ShiftRows` was maintained for implementation reasons.

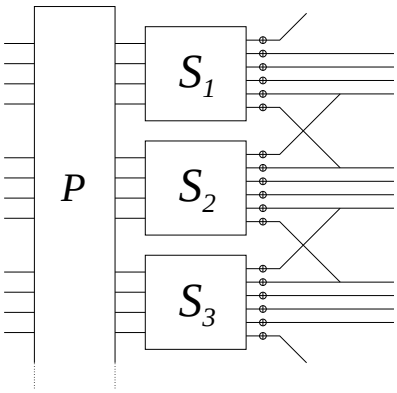


FIGURE 1. The round function of DES.

B.2. DES

DES is a 64-bit block 56-bit key block cipher. It is essentially a 16-round Feistel network, preceded and followed by bit permutation layers IP (Initial Permutation) and FP (Final Permutation), which have no cryptographic significance; moreover the last swap of the left and right parts is omitted. The round function of DES is pictured in Figure 1 (limited to its first 12 bits). It is made out of four layers:

- The expansion layer E transforms a 32-bit input into a 48-bit output by duplicating some of the input bits. Figure 1 permits to figure out how it is designed.
- The key addition layer adds the 48-bit round key to the 48 bits outputting E.
- The non-linear layer is made out of the parallel application of 8 different 6×4 S-boxes. They have the property that each of their 16 possible outputs can be caused by 4 different inputs.
- The last layer is a bit permutation layer P.

The key schedule of DES actually takes a 64-bit key as input. But 8 of them are parity bits which are immediately discarded by the *Permuted Choice* function $PC-1$. Moreover $PC-1$ applies a bit permutation to the remaining 56 bits. The key schedule of DES is pictured in Figure 2. The state during the application of the algorithm is divided in two parts $\langle C_i, D_i \rangle$ of 28 bits each. At each stage, each part is rotated left ($\lll$) by 1 or 2 bits (1 only at stages 1, 2, 9, and 16). The result passes through another *Permuted Choice* function $PC-2$ which selects 48 out of the 56 bits and applies a bit permutation to them. The result after stage i is the i^{th} round key k_i. Note that in order to generate the round keys in reverse order for decryption, one only has to replace the left rotates $\lll$ by right rotates $\ggg$, and to cancel the rotate of the first round.

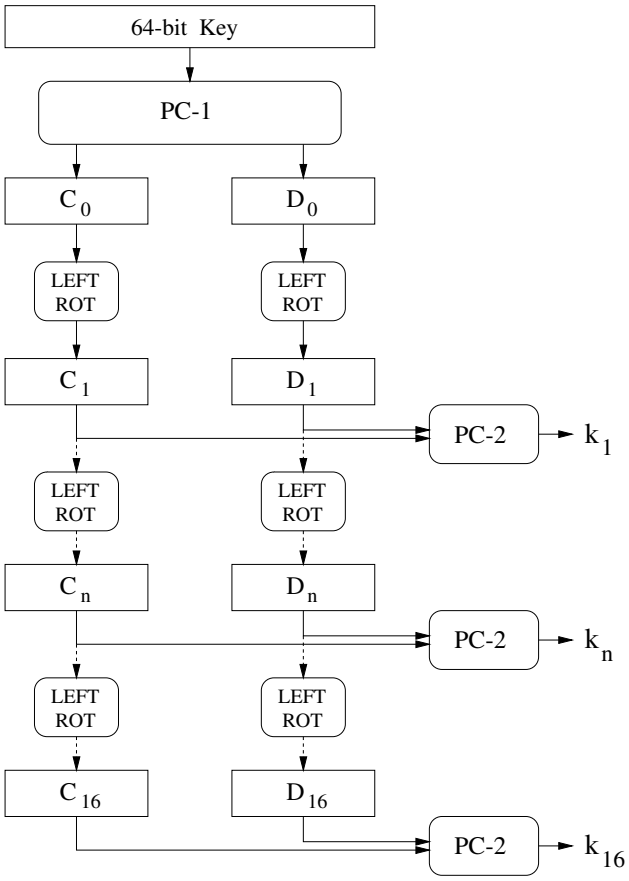


FIGURE 2. The key schedule of DES.

APPENDIX C

ICEBERG: Proof of Theorem 34

We have to prove that $P4 \circ \sigma_{RK_1} \circ M \equiv M \circ \sigma_{RK_0} \circ P4$. Inputs and outputs of every transform are represented in Figure 1. We simply write down relations between them. First we encrypt with RK_1:

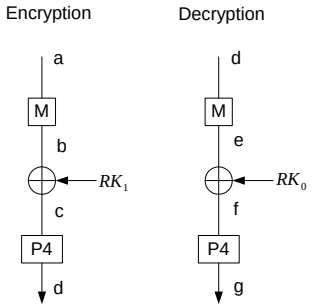


FIGURE 1. Theorem 1.

$$
\begin{aligned}
b_0 &= a_1 \oplus a_2 \oplus a_3 \\
b_1 &= a_0 \oplus a_2 \oplus a_3 \\
b_2 &= a_0 \oplus a_1 \oplus a_3 \\
b_3 &= a_0 \oplus a_1 \oplus a_2
\end{aligned}
$$

$$
\begin{aligned}
c_0 &= a_1 \oplus a_2 \oplus a_3 \oplus k_0 \oplus k_1 \oplus k_2 \\
c_1 &= a_0 \oplus a_2 \oplus a_3 \oplus k_1 \oplus k_2 \\
c_2 &= a_0 \oplus a_1 \oplus a_3 \oplus k_2 \oplus k_3 \oplus k_0 \\
c_3 &= a_0 \oplus a_1 \oplus a_2 \oplus k_3 \oplus k_0
\end{aligned}
$$

$$
\begin{aligned}
d_0 &= a_0 \oplus a_2 \oplus a_3 \oplus k_1 \oplus k_2 \\
d_1 &= a_1 \oplus a_2 \oplus a_3 \oplus k_0 \oplus k_1 \oplus k_2 \\
d_2 &= a_0 \oplus a_1 \oplus a_2 \oplus k_3 \oplus k_0 \\
d_3 &= a_0 \oplus a_1 \oplus a_3 \oplus k_2 \oplus k_3 \oplus k_0
\end{aligned}
$$

Then, when we decrypt with RK_0:

$$\begin{aligned} e_0 &= d_1 \oplus d_2 \oplus d_3 = a_1 \oplus k_0 \oplus k_1 \\ e_1 &= d_0 \oplus d_2 \oplus d_3 = a_0 \oplus k_1 \\ e_2 &= d_0 \oplus d_1 \oplus d_3 = a_3 \oplus k_2 \oplus k_3 \\ e_3 &= d_0 \oplus d_1 \oplus d_2 = a_2 \oplus k_3 \end{aligned}$$

From Equation (102), we have:

$$\begin{aligned} f_0 &= a_1 \\ f_1 &= a_0 \\ f_2 &= a_3 \\ f_3 &= a_2 \end{aligned}$$

And finally, permutation $P4$ finishes the decryption:

$$\begin{aligned} g_0 &= a_0 \\ g_1 &= a_1 \\ g_2 &= a_2 \\ g_3 &= a_3 \end{aligned}$$

Remark that permutation $P4$ allows the selection function X_{sel} to be efficiently implemented in LUTs as it has at most 4 inputs: 3 key bits and a selection bit.

APPENDIX D

ICEBERG Tables

0	1	2	3
1	0	3	2

TABLE 1. p4.

0	1	2	3	4	5	6	7
0	1	4	5	2	3	6	7

TABLE 2. p8.

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

TABLE 3. s_0.

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

TABLE 4. s_1.

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

TABLE 5. D.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	24	c1	38	30	e7	57	df	20	3e	99	1a	34	ca	d6	52	fd
10	40	6c	d3	3d	4a	59	f8	77	fb	61	0a	56	b9	d2	fc	f1
20	07	f5	93	cd	00	b6	62	a7	63	fe	44	bd	5f	92	6b	68
30	03	4e	a2	97	0b	60	83	a3	02	e5	45	67	f4	13	08	8b
40	10	ce	be	b4	2a	3a	96	84	c8	9f	14	c0	c4	6f	31	d9
50	ab	ae	0e	64	7c	da	1b	05	a8	15	a5	90	94	85	71	2c
60	35	19	26	28	53	e2	7f	3b	2f	a9	cc	2e	11	76	ed	4d
70	87	5e	c2	c7	80	b0	6d	17	b2	ff	e4	b7	54	9d	b8	66
80	74	9c	db	36	47	5d	de	70	d5	91	aa	3f	c9	d8	f3	f2
90	5b	89	2d	22	5c	e1	46	33	e6	09	bc	e8	81	7d	e9	49
a0	e0	b1	32	37	ea	5a	f6	27	58	69	8a	50	ba	dd	51	f9
b0	75	a1	78	d0	43	f7	25	7b	7e	1c	ac	d4	9a	2b	42	e3
c0	4b	01	72	d7	4c	fa	eb	73	48	8c	0c	f0	6a	23	41	ec
d0	b3	ef	1d	12	bb	88	0d	c3	8d	4f	55	82	ee	ad	86	06
e0	a0	95	65	bf	7a	39	98	04	9b	9e	a4	c6	cf	6e	dc	d1
f0	cb	1f	8f	8e	3c	21	a6	b5	16	af	c5	18	1e	0f	29	79

TABLE 6. 8 x 8 substitution box.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	23	25	38	42	53	59	22	9	26	32	1	47	51	61
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
24	37	18	41	55	58	8	2	16	3	10	27	33	46	48	62
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
11	28	60	49	36	17	4	43	50	19	5	39	56	45	29	13
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
30	35	40	14	57	6	54	20	44	52	21	7	34	15	31	63

TABLE 7. P64.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
76	110	83	127	67	114	92	97	98	65	121	106	78	112	91	82
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
71	101	89	126	72	107	81	118	90	124	73	88	64	104	100	85
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
109	87	75	113	120	66	103	115	122	108	95	69	74	116	80	102
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
84	96	125	68	93	105	119	79	123	86	70	117	111	77	99	94
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
28	9	37	4	51	43	58	16	20	26	44	34	0	61	12	55
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
46	22	15	2	48	31	57	33	27	18	24	14	6	52	63	42
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
49	7	8	62	30	17	47	38	29	53	11	21	41	32	1	60
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
13	35	5	39	45	59	23	54	36	10	40	56	25	50	19	3

TABLE 8. P128.

APPENDIX E

Generation of the ICEBERG S-box

As $P8$ is fixed, the only part of the ICEBERG S-box structure that remains unspecified consists of the s_0 and s_1 involutions, which are generated pseudo-randomly in a verifiable way.

The searching algorithm [8] starts with two copies of a simple involution without fixed points (namely, the negation mapping $u \mapsto \bar{u} = u \oplus \texttt{0xF}$), and pseudo-randomly derives from each of them a sequence of 4×4 substitution boxes ("mini-boxes") with the optimal values $\delta = 1/4$, $\lambda = 1/2$, and $\nu = 3$. At each step, in alternation, only one of the sequences is extended with a new mini-box. The most recently generated mini-box from each sequence is taken, and the pair is combined according to the ICEBERG S-box shuffle structure. Finally, the resulting 8×8 S-box, if free of fixed points, is tested for the design criteria regarding δ, λ, and ν.

Given a mini-box at any point during the search, a new one is derived from it by choosing two pairs of mutually inverse values and swapping them, keeping the result an involution without fixed points. This is repeated until the running mini-box has optimal values of δ, λ, and ν.

The pseudo-random number generator is implemented using the AES cipher Rijndael in counter mode, with a fixed key consisting of 128 zero bits and an initial counter value consisting of 128 zero bits.

The following pseudo-code fragment illustrates the computation of the chains of mini-boxes and the resulting S-box:

```
procedure ShuffleStructure(s_0, s_1)
    for w ← 0 to 255 do
        u_0 ← s_0[w ≫ 4];   v_0 ← s_0[w & 0x0F];
        u_1 ← (u_0 & 0xC) | ((v_0 & 0xC) ≫ 2);   v_1 ← (v_0 & 0x3) | ((u_0 & 0x3) ≪ 2);
        u_0 ← s_1[u_1];   v_0 ← s_1[v_1];
        u_1 ← (u_0 & 0xC) | ((v_0 & 0xC) ≫ 2);   v_1 ← (v_0 & 0x3) | ((u_0 & 0x3) ≪ 2);
        S[w] ← (s_0[u_1] ≪ 4) | s_0[v_1];
    end for
    return S;
end procedure

procedure SearchRandomSBox()
    // initialize mini-boxes to the negation involution:
    for u ← 0 to 255 do
        s_0[u] ← ū; s_1[u] ← ū;
    end for
```

```
    // look for S-box conforming to the design criteria:
    repeat
        // swap mini-boxes (update the "older" one only)
        s_0 ↔ s_1;
        // randomly generate a "good" Z_2^4 involution free of fixed points:
        repeat
            repeat
                // randomly select x and y such that
                // x ≠ y and s_1[x] ≠ y (this implies s_1[y] ≠ x):
                z ← RandomByte(); x ← z ≫ 4; y ← z & 0x0F;
            until x ≠ y ∧ s_1[x] ≠ y;
            // swap entries:
            u ← s_1[x]; v ← s_1[y];
            s_1[x] ← v; s_1[u] ← y;
            s_1[y] ← u; s_1[v] ← x;
        until δ(s_1) = 1/4 ∧ λ(s_1) = 1/2 ∧ ν(s_1) = 3;
        // build S-box from the mini-boxes:
        S ← ShuffleStructure(s_0, s_1);
        // test the design criteria:
    until #FixedPoints(S) = 0 ∨ δ(S) ⩽ 2^-5 ∧ λ(S) ⩽ 2^-2 ∧ ν(S) = 7;
    return S;
end procedure
```

Bibliography

[1] 3G TS 35.201. Specification of the 3gpp confidentiality and integrity algorithm; document 1: f8 and f9 specifications. Available at `http://www.3gpp.org`.

[2] R. Anderson, E. Biham, and L.R. Knudsen. Serpent: A Proposal for the Advanced Encryption Standard. NIST AES Proposal, 1998. Available from `http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/serpent.pdf`.

[3] R. Anderson and M. Kuhn. Tamper Resistance – a Cautionary Note. In *Proc. of the second USENIX workshop on electronic commerce, Oakland, USA, Nov. 18-21, 1996*, pages 1–11, 1996. Available at `http://www.cl.cam.ac.uk/~mgk25/tamper.pdf`.

[4] R. Anderson and M. Kuhn. Low Cost Attacks on Tamper Resistant Devices. In B. Christianson, B. Crispo, M. Lomas, and M. Roe, editors, *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer-Verlag, 1997. Available at `http://www.cl.cam.ac.uk/~mgk25/tamper2.pdf`.

[5] K. Aoki, T. Ichikawa, and al. Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis. In D.R. Stinson and S.E. Tavares, editor, *Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Canada, August 14-15, 2000*, volume 2012 of *Lecture Notes in Computer Science*, pages 39–56. Springer-Verlag, 2001.

[6] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The Sorcerer´s Apprentice Guide to Fault Attacks. Cryptology ePrint Archive, 2004/100, 2004. `http://eprint.iacr.org/2004/100.pdf`.

[7] P.S.L.M. Barreto and V. Rijmen. The Anubis Block Cipher. Submitted as a NESSIE Candidate Algorithm. Available at `http://www.cryptonessie.org`.

[8] P.S.L.M. Barreto and V. Rijmen. The Khazad Legacy-Level Block Cipher. Submitted as a NESSIE Candidate Algorithm. Available at `http://www.cryptonessie.org`.

[9] P.S.L.M. Barreto, V. Rijmen, J. Nakahara, B. Preneel, and al. Improved SQUARE Attacks against Reduced-Round HIEROCRYPT. In Mitsuru Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 165–173. Springer-Verlag, 2002.

[10] 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 - FOCS '97, Miami Beach, USA, October 19-22, 1997*, pages 394–403. IEEE Computer Society, 1997. Full paper available at `http://www.cs.ucdavis.edu/~rogaway/papers/sym-enc-abstract.html`.

[11] J.-L. Beuchat. FPGA Implementations of the RC6 Block Cipher. In P. Y. K. Cheung, G. A. Constantinides, and J. T. de Sousa, editors, *Field Programmable Logic and Application, 13th International Conference, FPL 2003, Lisbon, Portugal, September 1-3, 2003*, volume 2778 of *Lecture Notes in Computer Science*, pages 101–110. Springer-Verlag, 2003.

[12] J.-L. Beuchat. Modular Multiplication for FPGA Implementation of the IDEA Block Cipher. In E. Deprettere, S. Bhattacharyya, J. Cavallaro, A. Darte, and L. Thiele, editors, *Proceedings of the 14th IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAC 2003)*, pages 412–422. IEEE Computer Society, 2003.

[13] E. Biham. New Types of Cryptanalytic Attacks Using Related Keys (Extended Abstract). In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Lofthus, Norway, May 23-27, 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 398–409. Springer-Verlag, 1994.

[14] E. Biham. A Fast New DES Implementation in Software. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. Springer-Verlag, 1997.

[15] E. Biham, A. Biryukov, N. Ferguson, L.R. Knudsen, B. Schneier, and A. Shamir. Cryptanalysis of Magenta. In *Proc. of the 2nd AES Candidate Conference*, pages 182–183, 1999. Available from `http://csrc.nist.gov/CryptoToolkit/aes/round1/conf2/papers/biham1.pdf`.

[16] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, Prague, Czech Republic, May 2-6, 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 12–23, Berlin, 1999. Springer-Verlag.

[17] E. Biham, A. Biryukov, and A. Shamir. Miss in the middle attacks on IDEA, Khufu, and Khafre. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999*, volume 1636 of *Lecture Notes in Computer Science*, pages 124–138, Berlin, 1999. Springer-Verlag.

[18] E. Biham, O. Dunkelman, and N. Keller. The Rectangle Attack - Rectangling the Serpent. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, Innsbruck, Austria, May 6-10, 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 340–357. Springer-Verlag, 2001.

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

[20] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In Burton S. Kaliski, editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997.

[21] A. Biryukov. Analysis of Involutional Ciphers: Khazad and Anubis. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 45–53. Springer-Verlag, 2003.

[22] A. Biryukov, C. De Cannière, and G. Dellkrantz. Cryptanalysis of SAFER++. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, Santa Barbara, USA, August 17-21, 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 195–211. Springer-Verlag, 2003.

[23] A. Biryukov, C. De Cannière, and M. Quisquater. On Multiple Linear Approximations. In *Proceedings of CRYPTO 2004*, 2004. Also available at `http://eprint.iacr.org/2004/057.pdf`.

[24] A. Biryukov and A. Shamir. Structural Cryptanalysis of SASAS. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, Innsbruck, Austria, May 6-10, 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 394–405. Springer-Verlag, 2001.

[25] A. Biryukov and D. Wagner. Slide Attacks. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259, Berlin, 1999. Springer-Verlag.

[26] A. Biryukov and D. Wagner. Advanced Slide Attacks. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, Bruges, Belgium, May 14-18, 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 589–606, Berlin, 2000. Springer-Verlag.

[27] J. Blömer and J.-P. Seifert. Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In Rebecca N. Wright, editor, *Financial Cryptography, 7th International Conference, FC 2003, Guadeloupe, January 27-30, 2003*, Lecture Notes in Computer Science, pages 162–181. Springer-Verlag, 2003. Also available at http://eprint.iacr.org/, 2002/075.

[28] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, Konstanz, Germany, May 11-15, 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.

[29] E. Brier, H. Handschuh, and C. Tymen. Fast Primitives for Internal Data Scrambling in Tamper Resistant Hardware. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Paris, France, May 14-16, 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 16–27. Springer-Verlag, 2001.

[30] L. Brown, M. Kwan, J. Pieprzyk, and J. Seberry. Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI. In H. Imai, R.L. Rivest, and T. Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91, Fujiyoshida, Japan, November 11-14, 1991*, volume 739 of *Lecture Notes in Computer Science*, pages 36–50. Springer-Verlag, 1993.

[31] L. Brown, J. Pieprzyk, and J. Seberry. LOKI - A Cryptographic Primitive for Authentication and Secrecy Applications. In J. Seberry and J. Pieprzyk, editor, *Advances in Cryptology - AUSCRYPT '90, International Conference on Cryptology, Sydney, Australia, January 8-11, 1990*, volume 453 of *Lecture Notes in Computer Science*, pages 229–236. Springer-Verlag, 1990.

[32] C. Burwick, D. Coppersmith, and al. MARS - a candidate cipher for AES. NIST AES Proposal. Available from `http://www.research.ibm.com/security/mars.pdf`, September 1999.

[33] A. Canteaut and M. Minier. Personal communication. October 2004.

[34] S. Chari, C.S. Jutla, J.R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, Santa Barbara, USA, August 15-19, 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag, 1999.

[35] S. Chari, J.R. Rao, and P. Rohatgi. Template Attacks. In B.S. Kaliski, Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, Redwood Shores, USA, August 13-15, 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer-Verlag, 2003.

[36] C.-N. Chen and S.-M. Yen. Differential Fault Analysis on AES Key Schedule and Some Countermeasures. In R. Safavi-Naini and J. Seberry, editors, *Information Security and Privacy, 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 118–129. Springer-Verlag, 2003.

[37] R. Chung-Wei Phan. Related-Key Attacks on Triple-DES and DESX Variants. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, San Francisco, USA, February 23-27, 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 15–24. Springer-Verlag, 2004.

[38] R. Chung-Wei Phan and S. Furuya. Sliding Properties of the DES Key Schedule and Potential Extensions to the Slide Attacks. In P.J. Lee and C.H. Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference, Seoul, Korea, November 28-29, 2002*, volume 2587 of *Lecture Notes in Computer Science*, pages 138–148. Springer-Verlag, 2003.

[39] R. Chung-Wei Phan and H. Handschuh. On Related-Key and Collision Attacks: The Case for the IBM 4758 Cryptoprocessor. In K. Zhang and Y. Zheng, editors, *Information Security, 7th International Conference, ISC 2004, Palo Alto, USA, September 27-29, 2004*, volume 3225 of *Lecture Notes in Computer Science*, pages 111–122. Springer-Verlag, 2004.

[40] J.-S. Coron and L. Goubin. On Boolean and Arithmetic Masking against Differential Power Analysis. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Worcester, USA, August 17-18, 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 231–237. Springer-Verlag, 2000.

[41] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, Bruges, Belgium, May 14-18, 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407, Berlin, 2000. Springer-Verlag.

[42] N. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, Queenstown, New Zealand, December 1-5, 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer-Verlag, 2002.

[43] J. Daemen. *Cipher and Hash Function Design.* PhD thesis, KULeuven, March 1995.

[44] J. Daemen, L.R. Knudsen, and V. Rijmen. The Block Cipher SQUARE. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer-Verlag, 1997.

[45] J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen. Nessie proposal: NOEKEON. Submitted as a NESSIE Candidate Algorithm, 2000. Available from `http://www.cryptonessie.org`.

[46] J. Daemen and V. Rijmen. *The Design of Rijndael.* Information Security and Cryptography - Texts and Monographs. Springer-Verlag, Berlin, 2002.

[47] J. Daemen and V. Rijmen. AES proposal: Rijndael. In *Proc. first AES conference*, August 1998. Available on-line from the official AES page: `http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf`.

[48] I.B. Damgärd and L.R. Knudsen. Two-Key Triple Encryption. *Journal of Cryptology*, 11(3):209–218, 1998.

[49] A. Desai and S.K. Miner. Concrete Security Characterizations of PRFs and PRPs: Reductions and Applications. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, Kyoto, Japan, December 3-7, 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 503–516. Springer-Verlag, 2000.

[50] C. D'Halluin, G. Bijnens, V. Rijmen, and B. Preneel. Attack on Six Rounds of CRYPTON. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999*, volume 1636 of *Lecture Notes in Computer Science*, pages 46–59, Berlin, 1999. Springer-Verlag.

[51] P. Dusart, G. Letourneux, and O. Vivolo. Differential Fault Analysis on A.E.S. In J. Zhou, M. Yung, and Y. Han, editors, *Applied Cryptography and Network Security, First International Conference, ACNS 2003. Kunming, China, October 16-19, 2003*, volume 2846 of *Lecture Notes in Computer Science*, pages 293–306. Springer-Verlag, 2003.

[52] S. Even and Y. Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *Journal of Cryptology*, 10(3):151–162, 1997.

[53] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC 87), New York City, 25-27 May 1987*, pages 210–217, 1987.

[54] H. Feistel. Cryptography and Computer Privacy. *Scientific American*, 228(5):15–23, 1973.

[55] N. Ferguson, J. Kelsey, and al. Improved Cryptanalysis of Rijndael. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, USA, April 10-12, 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer-Verlag, 2000.

[56] S. Fernandez-Gomez, J.J. Rodriguez-Andina, and E. Mandado. Concurrent error detection in block ciphers. In *International Test Conference (ITC)*, IEEE, 2000.

[57] W.W. Fung and J.W. Gray. Protection Against EEPROM Modification Attacks. In C. Boyd and E. Dawson, editors, *Information Security and Privacy, Third Australasian Conference, ACISP'98, Brisbane, Australia, July 1998*, volume 1438 of *Lecture Notes in Computer Science*, pages 250–260. Springer-Verlag, 1998.

[58] S. Furuya. Slide Attacks with a Known-Plaintext Cryptanalysis. In Kwangjo Kim, editor, *Information Security and Cryptology - ICISC 2001, 4th International Conference, Seoul, Korea, December 6-7, 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 214–225. Springer-Verlag, 2002.

[59] K. Gaj and P. Chodowiec. Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard Using Field Programmable Gate Arrays. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, San Francisco, USA, April 8-12, 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 84–99. Springer-Verlag, 2001.

[60] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Paris, France, May 14-16, 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer-Verlag, 2001.

[61] H. Gilbert and M. Minier. A Collision Attack on 7 Rounds of Rijndael. In *Proc. of the 3rd AES Candidate Conference*, pages 230–241, 2000. Available from `http://csrc.nist.gov/CryptoToolkit/aes/round2/conf3/aes3conf.htm`.

[62] H. Gilbert and A. Tardy-Corfdir. A Known Plaintext Attack of FEAL-4 and FEAL-6. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, USA, August 11-15, 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 172–181. Springer-Verlag, 1992.

[63] C. Giraud. DFA on AES. Technical Report 2003/008, IACR eprint archive, 2003. Available at http://eprint.iacr.org/2003/008.ps.

[64] C. Giraud and H. Thiebeauld. A Survey on Fault Attacks. In J.-J. Quisquater, P. Paradinas, Y. Deswarte, and A.A. El Kalam, editors, *Smart Card Research and Advanced Applications VI - 18th IFIP World Computer Congress*, pages 159–176. Kluwer Academic Publishers, August 2004.

[65] J.D. Golić. DeKaRT: A New Paradigm for Key-Dependent Reversible Circuits. In C.D. Walter, Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, Cologne, Germany, September 8-10, 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 98–112. Springer-Verlag, 2003.

[66] L. Goubin and J. Patarin. DES and Differential Power Analysis (The "Duplication" Method). In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES '99, Worcester, USA, August 12-13, 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer-Verlag, 1999.

[67] L. Granboulan. Flaws in Differential Cryptanalysis of Skipjack. In Mitsuru Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 328–335. Springer-Verlag, 2002.

[68] S. Halevi and P. Rogaway. A Tweakable Enciphering Mode. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, Santa Barbara, USA, August 17-21, 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer-Verlag, 2003.

[69] C. Harpes and J.L. Massey. Partitioning Cryptanalysis. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 13–27. Springer-Verlag, 1997.

[70] Helion Technology. High Performance AES (Rijndael) Cores for XILINX FPGA. Available at `http://www.heliontech.com`.

[71] L. Hemme. A Differential Fault Attack Against Early Rounds of (Triple-)DES. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004, Cambridge, USA, August 11-13, 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 254–267. Springer-Verlag, 2004.

[72] H.M. Heys and S.E. Tavares. Known Plaintext Cryptanalysis of Tree-Structured Block Ciphers. *Electronic Letters*, 31(10):784–785, May 1995.

[73] J.J. Hoch and A. Shamir. Fault Analysis of Stream Ciphers. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004, Cambridge, USA, August 11-13, 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 240–253. Springer-Verlag, 2004.

[74] K. Hwang, W. Lee, S. Lee, S. Lee, and J. Lim. Saturation Attacks on Reduced Round Skipjack. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 100–111, Berlin, 2002. Springer-Verlag.

[75] T. Ichikawa, T. Sorimachi, T. Kasuya, and M. Matsui. On the Criteria of Hardware Evaluation of Block Ciphers. Technical report, IEICE, ISEC, 2001.

[76] T. Iwata, T. Yoshino, and K. Kurosawa. Non-cryptographic Primitive for Pseudorandom Permutation. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 149–163. Springer-Verlag, 2002.

[77] T. Iwata, T. Yoshino, T. Yuasa, and K. Kurosawa. Round Security and Super-Pseudorandomness of MISTY Type Structure. In Mitsuru Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 233–247. Springer-Verlag, 2002.

[78] G. Jakimoski and Y. Desmedt. Related-Key Differential Cryptanalysis of 192-bit Key AES Variants. In M. Matsui and R.J. Zuccherato, editor, *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 208–221. Springer-Verlag, 2004.

[79] T. Jakobsen and L.R. Knudsen. The Interpolation Attack on Block Ciphers. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer-Verlag, 1997.

[80] N. Joshi, K. Wu, and R. Karri. Concurrent Error Detection Schemes for Involution Ciphers. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004, Cambridge, USA, August 11-13, 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 400–412. Springer-Verlag, 2004.

[81] M. Joye, P. Paillier, and S.M. Yen. Secure Evaluation of Modular Functions. In R.J. Hwang and C.K. Wu, editors, *International Workshop on Cryptology and Network Security, Taipei, Taiwan, September 26-28, 2001*, pages 227–229, 2001.

[82] P. Junod. On the Complexity of Matsui's Attack. In S. Vaudenay and A.M. Youssef, editors, *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001, Toronto, Ontario, Canada, August 16-17, 2001*, volume 2259 of *Lecture Notes in Computer Science*, pages 199–211. Springer-Verlag, 2001.

[83] P. Junod. On the Optimality of Linear, Differential, and Sequential Distinguishers. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, Warsaw, Poland, May 4-8, 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, 2003.

[84] P. Junod and S. Vaudenay. Optimal Key Ranking Procedures in a Statistical Cryptanalysis. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 235–246. Springer-Verlag, 2003.

[85] B. S. Kaliski and M. J. B. Robshaw. Linear Cryptanalysis Using Multiple Approximations. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, Santa Barbara, USA, August 21-25, 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 26–39, Berlin, 1994. Springer-Verlag.

[86] M. Karpovsky, K. Kulikowski, and A. Taubin. Differential Fault Analysis Attack Resistant Architectures for the Advanced Encryption Standard. In J.-J. Quisquater, P. Paradinas, Y. Deswarte, and A.A. El Kalam, editors, *Smart Card Research and Advanced Applications VI - 18th IFIP World Computer Congress*, pages 177–192. Kluwer Academic Publishers, August 2004.

[87] R. Karri, G. Kuznetsov, and M. Goessel. Parity-Based Concurrent Error Detection of Substitution-Permutation Network Block Ciphers. In C.D. Walter, Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, Cologne, Germany, September 8-10, 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 113–124. Springer-Verlag, 2003.

[88] R. Karri, K. Wu, P. Mishra, and Y. Kim. Concurrent error detection of fault-based side-channel cryptanalysis of 128-bit symmetric block ciphers. In *DAC 2001*, ACM 1-58113-297-2/01/0006, 2001.

[89] J. Kelsey, B. Schneier, and D. Wagner. Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, Santa Barbara, USA, August 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 237–251. Springer-Verlag, 1996.

[90] J. Kelsey, B. Schneier, and D. Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Y. Han, T. Okamoto, and S. Qing, editors, *Information and Communication Security, First International Conference, ICICS'97, Beijing, China, November 11-14, 1997*, volume 1334 of *Lecture Notes in Computer Science*, pages 233–246. Springer-Verlag, 1997.

[91] J. Kilian and P. Rogaway. How to Protect DES Against Exhaustive Key Search. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, Santa Barbara, USA, August 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 252–267. Springer-Verlag, 1996.

[92] J. Kim, G. Kim, S. Hong, S. Lee, and D. Hong. The Related-Key Rectangle Attack - Application to SHACAL-1. In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004*, volume 3108 of *Lecture Notes in Computer Science*, pages 123–136. Springer-Verlag, 2004.

[93] L. R. Knudsen and M. J. B. Robshaw. Non-Linear Approximations in Linear Cryptoanalysis. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, Saragossa, Spain, May 12-16, 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 224–236, Berlin, 1996. Springer-Verlag.

[94] L.R. Knudsen. Cryptanalysis of LOKI91. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology - ASIACRYPT '92, Gold Coast, Australia, December 13-16, 1992*, volume 718 of *Lecture Notes in Computer Science*, pages 196–208. Springer-Verlag, 1993.

[95] L.R. Knudsen. New Potentially 'Weak' Keys for DES and LOKI (Extended Abstract). In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Perugia, Italy, May 9-12, 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 419–424. Springer-Verlag, 1995.

[96] L.R. Knudsen. Truncated and Higher Order Differentials. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1995.

[97] L.R. Knudsen and J.E. Mathiassen. A Chosen-Plaintext Linear Attack on DES. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, USA, April 10-12, 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 262–272. Springer-Verlag, 2000.

[98] L.R. Knudsen, M.J.B. Robshaw, and D. Wagner. Truncated Differentials and Skipjack. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, Santa Barbara, USA, August 15-19, 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 165–180, Berlin, 1999. Springer-Verlag.

[99] L.R. Knudsen and D. Wagner. Integral Cryptanalysis. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer-Verlag, 2002.

[100] P. Kocher. Timing Attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, Santa Barbara, USA, August 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[101] P. Kocher, Jaffe J., and B. Jub. Differential Power Analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, Santa Barbara, USA, August 15-19, 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[102] F. Koeune and J.-J. Quisquater. A timing attack against Rijndael. Technical report, available at http://www.dice.ucl.ac.be/crypto/techreports.html, 1999.

[103] F. Koeune and J.-J. Quisquater. Side channel attacks - State of the Art, October 2002. Available at `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf`.

[104] O. Kömmerling and M. Kuhn. Design principles for Tamper-Resistant Smartcard Processors. In *USENIX Workshop on Smartcard Technology, Chicago, USA*, May 1999. Available at `http://www.cl.cam.ac.uk/~mgk25/sc99-tamper.pdf`.

[105] X. Lai and J. L. Massey. A Proposal for a New Block Encryption Standard. In Ivan Damgärd, editor, *Advances in Cryptology - EUROCRYPT '90, Aarhus, Denmark, May 21-24, 1990*, volume 473 of *Lecture Notes in Computer Science*, pages 389–404. Springer-Verlag, 1990.

[106] X. Lai and J. L. Massey. Markov ciphers and differential cryptanalysis. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Brighton, UK, April 8-11, 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38, Berlin, 1991. Springer-Verlag.

[107] C.H. Lim. Crypton : A New 128-bit Block Cipher. In *The First Advanced Encryption Standard Candidate Conference*. NIST, 1998.

[108] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[109] S. Lucks. Faster Luby-Rackoff Ciphers. In Dieter Gollmann, editor, *Fast Software Encryption, Cambridge, UK, February 21-23, 1996*, volume 1039 of *Lecture Notes in Computer Science*, pages 189–203. Springer-Verlag, 1996.

[110] S. Lucks. The Saturation Attack - a Bait for Twofish. In Mitsuru Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2002.

[111] S. Lucks. Ciphers Secure against Related-Key Attacks. In B.K. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 359–370. Springer-Verlag, 2004.

[112] J.L. Massey, G.H. Khachatrian, and Kuregian M.K. Nomination of SAFER++ as Candidate Algorithm for NESSIE. Available at `http://www.cryptonessie.org`.

[113] M. Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Lofthus, Norway, May 23-27, 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1993.

[114] M. Matsui. New Block Encryption Algorithm MISTY. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, 1997.

[115] U. Maurer. A Simplified and Generalized Treatment of Luby-Rackoff Pseudorandom Permutation Generators. In Rainer A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92, Balatonfüred, Hungary, May 24-28, 1992*, volume 658 of *Lecture Notes in Computer Science*, pages 239–255. Springer-Verlag, 1993.

[116] M. McLoone and J.V. McCanny. Very High Speed 17 Gbps SHACAL Encryption Architecture. In P. Y. K. Cheung, G. A. Constantinides, and J. T. de Sousa, editors, *Field Programmable Logic and Application, 13th International Conference, FPL 2003, Lisbon, Portugal, September 1-3, 2003*, volume 2778 of *Lecture Notes in Computer Science*, pages 111–120. Springer-Verlag, 2003.

[117] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[118] R.C. Merkle and M.E. Hellman. On the security of multiple encryption. *Communications of the ACM*, 24(7):465–467, July 1981.

[119] T. Messerges. Securing the AES Finalists Against Power Analysis Attacks. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, USA, April 10-12, 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer-Verlag, 2000.

[120] M. Minier. *Preuves d'Analyse et de Sécurité en Cryptologie à Clé Secrète*. PhD thesis, LACO, Université de Limoges, September 2002.

[121] M. Minier and H. Gilbert. New Results on the Pseudorandomness of Some Blockcipher Constructions. In Mitsuru Matsui, editor, *Fast Software Encryption, 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 248–266. Springer-Verlag, 2002.

[122] F. Muller. A New Attack against Khazad. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003, Taipei, Taiwan, November 30 - December 4, 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 347–358. Springer-Verlag, 2003.

[123] J. Nakahara, P.S.L.M. Barreto, B. Preneel, H.Y. Kim, and al. SQUARE Attacks on Reduced-Round PES and IDEA Block Ciphers. In B. Macq and J.-J. Quisquater, editors, *Proceedings of the 23rd Symposium on Information Theory in the Benelux*, pages 187–195. Werkgemeenshap voor Informatie- and Communicatietheorie, 2002.

[124] J. Nakahara, B. Preneel, and al. Linear cryptanalysis of reduced-round safer++. In *Proceedings of the second NESSIE Workshop*, 2001.

[125] M. Naor and O. Reingold. On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited. *Journal of Cryptology*, 12(1):29–66, 1999.

[126] NESSIE (New European Schemes for Signatures, Integrity, and Encryption). Project funded by the European Community, IST-1999-12324. See `http://www.cryptonessie.org/`.

[127] K. Nyberg and L.R. Knudsen. Provable Security Against Differential Cryptanalysis. *Journal of Cryptology*, 8(1):27–37, 1995.

[128] P. Onions. On the Strength of Simply-Iterated Feistel Ciphers with Whitening Keys. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, San Francisco, USA, April 8-12, 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 63–69. Springer-Verlag, 2001.

[129] P. Paillier. Evaluating Differential Fault Analysis of Unknown Cryptosystems. In H. Imai and Y. Zheng, editors, *Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999*, volume 1560 of *Lecture Notes in Computer Science*, pages 235–244. Springer-Verlag, 1999.

[130] J. Patarin. *Etude des Générateurs de Permutations Basés sur le Schéma du DES*. PhD thesis, Université Paris VI, November 1991.

[131] J. Patarin. How to Construct Pseudorandom and Super Pseudorandom Permutations from one Single Pseudorandom Function. In Rainer A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92, Balatonfüred, Hungary, May 24-28, 1992*, volume 658 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, 1993.

[132] J. Patarin. About Feistel Schemes with Six (or More) Rounds. In Serge Vaudenay, editor, *Fast Software Encryption, Paris, France, March 23-25, 1998*, volume 1372 of *Lecture Notes in Computer Science*, pages 103–121. Springer-Verlag, 1998.

[133] J. Patarin. Generic Attacks on Feistel Schemes. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, Gold Coast, Australia, December 9-13, 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 222–238. Springer-Verlag, 2001.

[134] J. Patarin. Luby-Rackoff: 7 Rounds Are Enough for $2^{n(1-\epsilon)}$ Security. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, Santa Barbara, USA, August 17-21, 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 513–529. Springer-Verlag, 2003.

[135] J. Patarin. Security of Random Feistel Schemes with 5 or More Rounds. In Matt Franklin, editor, *Advances in Cryptology - CRYPTO 2004, Santa Barbara, USA, August 15-19, 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 106–122. Springer-Verlag, 2004.

[136] S. Patel, Z. Ramzan, and G. Sundaram. Luby-Rackoff Ciphers: Why XOR Is Not So Exclusive. In K. Nyberg and H.M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 271–290. Springer-Verlag, 2003.

[137] A. Pfitzmann and R. Aßmann. More Efficient Software Implementations of (Generalized) DES. *Computers & Security*, 12(5):477–500, August 1993.

[138] D.H. Phan and D. Pointcheval. About the Security of Ciphers (Semantic Security and Pseudo-Random Permutations). In H. Handschuh and A. Hasan, editors, *Selected Areas in Cryptography, 11th Annual International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[139] G. Piret and J.-J. Quisquater. Impossible differential and square attacks: Cryptanalytic link and application to Skipjack. Technical Report CG-2001/4, UCL Crypto Group, 2001. Available at `http://www.dice.ucl.ac.be/crypto/tech\_reports/CG2001\_4.ps.gz`.

[140] G. Piret and J.-J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In C.D. Walter, Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, Cologne, Germany, September 8-10, 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer-Verlag, 2003.

[141] G. Piret and J.-J. Quisquater. Integral Cryptanalysis on reduced-round Safer++ . Technical Report 033, IACR eprint archive, 2003. Available at `http://eprint.iacr.org/2003/033/`.

[142] G. Piret and J.-J. Quisquater. Security of the MISTY Structure in the Luby-Rackoff Model: Improved Results. In H. Handschuh and A. Hasan, editors, *Selected Areas in Cryptography, 11th Annual International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[143] G. Piret, F.-X. Standaert, G. Rouvroy, and J.-J. Quisquater. On the Security of the *DeKaRT* Primitive. In J.-J. Quisquater, P. Paradinas, Y. Deswarte, and A.A. El Kalam, editors, *Smart Card Research and Advanced Applications VI - 18th IFIP World Computer Congress*, pages 241–254. Kluwer Academic Publishers, August 2004.

[144] J.-J. Quisquater and D. Samyde. A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions: the SEMA and DEMA methods. Presented at the rump session of EUROCRYPT '00, 14-18th May 2000, Bruges, Belgium.

[145] Z. Ramzan and L. Reyzin. On the Round Security of Symmetric-Key Cryptographic Primitives. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO*

2000, Santa Barbara, USA, August 20-24, 2000, volume 1880 of *Lecture Notes in Computer Science*, pages 376–393. Springer-Verlag, 2000.

[146] V. Rijmen. *Cryptanalysis and Design of Iterated Block Ciphers*. PhD thesis, KULeuven, October 1997.

[147] V. Rijmen, B. Preneel, and E. De Win. On Weaknesses of Non-surjective Round Functions. *Designs, Codes and Cryptography*, 12(3):253–266, 1997.

[148] R. Rivest. The RC5 Encryption Algorithm. In Bart Preneel, editor, *Fast Software Encryption: 2nd International Workshop, Leuven, Belgium, December 14-16, 1994*, volume 1008 of *Lecture Notes in Computer Science*, pages 86–96. Springer-Verlag, 1995.

[149] R. Rivest, M. Robshaw, R. Sidney, and Y. Yin. The rc6 block cipher. NIST AES Proposal. Available from `ftp://ftp.rsasecurity.com/pub/rsalabs/rc6/rc6v11.pdf`, August 1998.

[150] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[151] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat. Design Strategies and Modified Descriptions to Optimize Cipher FPGA Implementations: Fast and Compact Results for DES and Triple-DES. In P. Y. K. Cheung, G. A. Constantinides, and J. T. de Sousa, editors, *Field Programmable Logic and Application, 13th International Conference, FPL 2003, Lisbon, Portugal, September 1-3, 2003*, volume 2778 of *Lecture Notes in Computer Science*, pages 181–193. Springer-Verlag, 2003.

[152] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat. Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications. In *Proceedings of ITCC 2004, Las Vegas, April 5-7, 2004*, pages 583–587, 2004.

[153] K. Sakurai and Y. Zheng. On Non-Pseudorandomness from Block Ciphers with Provable Immunity Against Linear Cryptanalysis. *IEICE Trans. Fundamentals*, E80-A(1), January 1997.

[154] B. Schneier, J. Kelsey, and al. Twofish: A 128-bit block cipher. NIST AES Proposal. Available from `http://www.counterpane.com/twofish.html`, June 1998.

[155] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[156] A. Shamir. How to check modular exponentiation. Presented at the rump session of EUROCRYPT '97, 11-15th May 1997, Konstanz, Germany.

[157] C. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, October 1949.

[158] S. Skorobogatov and R. Anderson. Optical fault induction attacks. In B.S. Kaliski, Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, Redwood Shores, USA, August 13-15, 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer-Verlag, 2003.

[159] F.-X. Standaert. *Secure and Efficient Use of Reconfigurable Hardware Devices in Symmetric Cryptography*. PhD thesis, Université Catholique de Louvain, June 2004.

[160] F.-X. Standaert, G. Piret, G. Rouvroy, and J.-J. Quisquater. FPGA Implementations of the `ICEBERG` Block Cipher. Accepted at ITCC 2005.

[161] F.-X. Standaert, G. Piret, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. ICEBERG : an Involutional Cipher Efficient for Block Encryption on Reconfigurable Hardware. In B.K. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004*, volume

3017 of *Lecture Notes in Computer Science*, pages 279–299. Springer-Verlag, 2004.

[162] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. Efficient FPGA Implementations of Block Ciphers KHAZAD and MISTY1. In *Proceedings of the Third NESSIE Workshop, November 6-7 2002, Munich, Germany*, 2002.

[163] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs. In C.D. Walter, Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, Cologne, Germany, September 8-10, 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 334–350. Springer-Verlag, 2003.

[164] "Data Encryption Standard". Federal Information Processing Standard PUB 46. National Tech. Info. Service, Springfield, USA, 1977. Available from `http://www.csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`.

[165] P.C. van Oorschot and M.J. Wiener. A Known Plaintext Attack on Two-Key Triple Encryption. In Ivan Damgärd, editor, *Advances in Cryptology - EUROCRYPT '90, Aarhus, Denmark, May 21-24, 1990*, volume 473 of *Lecture Notes in Computer Science*, pages 318–325. Springer-Verlag, 1990.

[166] P.C. van Oorschot and M.J. Wiener. Improving Implementable Meet-in-the-Middle Attacks by Orders of Magnitude. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, Santa Barbara, USA, August 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 229–236. Springer-Verlag, 1996.

[167] 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*, volume 1373 of *Lecture Notes in Computer Science*, pages 249–275. Springer-Verlag, 1998.

[168] S. Vaudenay. Decorrelation: A Theory for Block Cipher Security. *Journal of Cryptology*, 16(4):249–286, 2003.

[169] D. Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer-Verlag, 1999.

[170] S.-M. Yen, S. Kim, S. Lim, and S.-J. Moon. RSA Speedup with Residue Number System Immune against Hardware Fault Cryptanalysis. In Kwangjo Kim, editor, *Information Security and Cryptology - ICISC 2001, 4th International Conference, Seoul, Korea, December 6-7, 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 397–413. Springer-Verlag, 2002.

[171] Y. Yeom, S. Park, and I. Kim. On the Security of CAMELLIA against the Square Attack. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 89–99. Springer-Verlag, 2002.

[172] A.M. Youssef, S.E. Tavares, and H.M. Heys. A New Class of Substitution-Permutation Networks. In *proceedings of SAC '96 - Workshop on Selected Areas in Cryptography, Kingston, Canada, Aug. 1996*, pages 132–147, 1996.