Université catholique de Louvain

École polytechnique de Louvain

Institute of Information and Communication
Technologies, Electronics and Applied Mathematics

Design and Experimental Validation of Statistical Attacks Against Block Ciphers

Baudoin Collard

Thèse présentée en vue de l'obtention du grade de docteur en sciences de l'ingénieur

Composition du jury:

Pr. Jean-Jacques Quisquater (UCL/ICTEAM/ELEN) – Promoteur
Pr. François-Xavier Standaert (UCL/ICTEAM/ELEN) – Promoteur
Pr. Olivier Pereira (UCL/ICTEAM/ELEN)
Pr. Marine Minier (INSA Lyon – France)
Pr. Vincent Rijmen (K.U.Leuven - Belgium)
Pr. Eli Biham (Technion - Israel)
Pr. Michel Verleysen (UCL/ICTEAM/ELEN) – Président

Louvain-la-Neuve, Belgique
2011

ACKNOWLEDGEMENTS

It is a pleasure for me to thank the people who made this thesis possible.

First, I would like to express my deep gratitude to my supervisors, François-Xavier Standaert and Jean-Jacques Quisquater for their guidance, suggestions and encouragements during the achievement of this thesis. Collaborating with them was an enlightening experience.

I also would like to thank Marine Minier, Eli Biham, Vincent Rijmen and Olivier Pereira for accepting to be part of my jury. I am also thankful to Michel Verleysen for having agreed to preside this jury.

Working with the Crypto Group members was a great pleasure, and I would like to thank them all for their friendly discussions and help. In particular, I would like to thank Guerric Meurice de Dormale, Francois *French* Mace, Benoit *Blib* Libert, Mathieu Renauld and Michael Modrikamen for providing a stimulating and fun office. Thanks to Damien Giry and François Koeune for bringing me out to the cinema so many times. I would also like to thank Brigitte Dupont for introducing me to the intricacies of the network administration and Sylvie Baudine for her availability and patient proofreading of my English mistakes.

The research in this thesis was supported by the project Nanotic-Cosmos of the Walloon Region. Their financial contribution as well as the freedom of action they allowed me is gratefully acknowledged.

I am also indebted to my teachers at the Institut Saint-Louis Namur who introduced me to sciences, especially Mr Massart, Mr Lefèvre and Ms Van Assche.

I wish to thank Alexandre, Vincent, Ludovic, John, David, Pierre-Yves, Thibaut, Jerome, Cihan, Amidou and Arnaud among others for the many evenings spent together after a hard work day! Thanks also to Julien, Sylvain and Anne-Sophie for all the friendship and trust they provided me.

As always, my family has offered me continuous encouragements. Valerie, Nathalie, Bastien, Justin and Roxane have always been there to support me. Catherine deserves a special mention for helping me get through the difficult times with her love and happy mood. Last but not least, I wish to thank my parents, Dominique and Philippe. They raised me, supported me, taught me, and loved me. To them I dedicate this thesis.

CONTENTS

THESIS OVERVIEW

1. INTRODUCTION

1.1 Cryptography

The advent of mass communication has led to the ubiquituous presence of information. Secure treatment of these data has become a main concern not only for governments, but also for companies and individuals. Cryptography is the study of the mathematical techniques related to the security of the information. Classical textbooks on cryptography generally consider different objectives, e.g.:

- *data confidentiality*: the data should not be eavesdropped and interpreted by an unauthorized third party.
- *data integrity*: the legitimate recipient should be able to determine if the data has been altered.
- *authentication*: in a communication, the receiver should be able to identify the sender. Furthermore, they should be able to verify that the sender did actually send the message.
- *non-repudiation*: the sender should not be able to deny having sent the message.

To achieve these goals, cryptography makes an extended use of *primitives*, which are algorithms providing basic functionalities. Such primitives generally take some (possibly) secret input data and process an output result. The secret part of the input is generally denoted as the key. Cryptographic primitives can be classified in two categories:

- Symmetric key primitives are the class of algorithms that use trivially related (often identical) cryptographic keys for reciprocal operations (for example encryption and decryption). Consequently, the keys represent a shared secret between the parties involved.
- Public-key primitives are the class of algorithms that use asymmetric keys: a secret private key and a public key. The keys are related mathematically, but the private key cannot be derived from the public key.

Cryptographic primitives cover a wide range of functionalities, among which we find encryption, pseudo-random number generation, hash functions, digital signatures. They are often combined into protocols in order to achieve more complex functionalities involving two or more parties.

In this thesis, we mainly focused our attention on the way confidentiality can be reached through the use of a symmetric-key *encryption algorithm*. Encryption refers to the process of transforming some piece of information (referred to as the *plaintext*), using an algorithm (called the *cipher*), to make it "unreadable" to anyone except those possessing the cipher key. The result of the process is a piece of encrypted information (referred to as the *ciphertext*). The reverse process, denoted as *decryption*, allows making the encrypted information readable again.

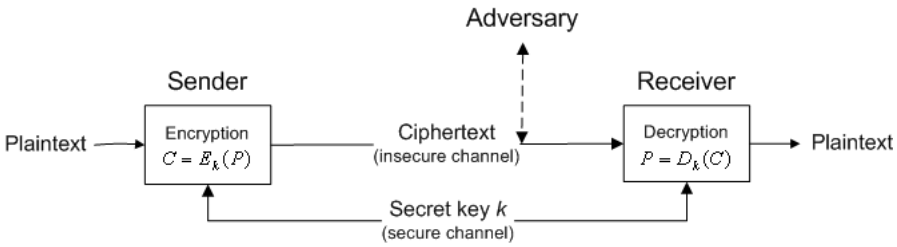

Fig. 1.1: Communication over an insecure channel

A simple model of a two-party communication using symmetric-key encryption is depicted in figure 1.1. It involves a sender and a receiver communicating through an *insecure channel* which can be eavesdropped by an adversary. For symmetric key, the model also requires the two parties to be able to exchange a secret key through a *secure channel* first. As such secure channel is usually very difficult to maintain, it cannot be used for the entire communication, but only to exchange the secret key. Once the parties have agreed on a shared secret key, they can start a secure communication on the insecure channel through the use of encryption. In some applications, the sender and the receiver can be the same person, for example to protect data on a storage device or data transiting through an insecure network.

1.2 Block ciphers

Primitives for symmetric encryption can be divided into *stream ciphers* and *block ciphers*. Whereas stream ciphers encrypt the bits of the message one at a time, block ciphers take a fixed number of bits and encrypt them as a single unit: if the message is longer than a single block, it is divided into several blocks and each one is encrypted individually.

The choice between these two primitives depends usually on the application specificities. Stream ciphers are often used for their speed and simplicity of implementation in hardware, and in applications where plaintext comes in quantities of variable length (for example, a secure wireless connection). However, they require an initialization vector to differenciate streams generated with the same key. Moreover, as they usually encrypt data one bit (or byte) at a time, an adversary can more easily alter the content of the message without knowing the key. Note finally that some mode of operation allows to turn a block cipher into a stream cipher [67].

In the following, we will concentrate more particularly on the analysis of block ciphers. Most block ciphers actually consist of two algorithms, one used for encryption, E, and the other for decryption, E^{-1}. Both algorithms accept two inputs: an input block of size n bits and a key of size k bits, yielding an n-bit output block such that:

$$E_K(P) = C \quad \text{and} \quad E_K^{-1}(C) = P \tag{1.1}$$

where P is the plaintext, C is the ciphertext and K is the secret key.

For practical reasons, most block ciphers are iterated by applying several times a simpler function called the *round function* [78]. This has the advantage to improve both hardware and software performance and to reduce the implementation costs. Typically, a round function applies a number of simple transformations which are weak if taken separately but strongly enhance the security of the cipher when combined. The basic transformations are usually:

- **Key mixing layer**: The key mixing layer incorporates the key material, usually through simple arithmetic operations such as bitwise *XOR* (denoted as $\oplus$) or modular additions. The round keys or *subkeys* are usually different from one round to another and derived from the *master key* through a *key scheduling algorithm*.

- **Confusion layer**: The confusion layer aims at making the relationship between the key and the ciphertext as complex and involved as possible. This is usually achieved by the use of *substitution box* or *S-box*. A S-box takes a fixed number of bits in input, and produces other bits as output in a non-linear way predefined by a *substitution table*. As the size of an S-box (typically 4 to 8 bits) is usually much smaller than the block size (typically 64 to 128 bits), a substitution layer is usually made of several parallel S-box applications.

- **Diffusion layer**: The diffusion layer aims at spreading redundancies from the input to the output such that each ciphertext bits depend on all the plaintext bits in a very complex way. This can be achieved by a simple permutation of the bits (as in the *DES* [72]) or by a more elaborated linear transformation (as in the *AES* [30]).

The idea to repeatedly combine confusion and diffusion to obtain a much stronger cipher comes from the seminal work of Shannon [82].

Frequently considered block cipher architectures include:

- The Feistel network
- The Substitution-Permutation network

In a Feistel cipher, the n-bit plaintext is divided into two parts (L_0, R_0) of equal size $n/2$. For each round, the two parts are updated as follows:

$$L_{i+1} = R_i \quad \text{and} \quad R_{i+1} = L_i \oplus F(R_i, K_i) \tag{1.2}$$

where F is the round function that takes a subkey K_i and a $n/2$-bit string as input and outputs $n/2$ bits. One round of a generic Feistel cipher is illustrated on figure 1.2. The Feistel structure has the advantage that encryption and decryption operations are very similar, even identical in some cases, requiring only a reversal of the key schedule. Moreover, the Feistel construction has been analysed in the *Luby-Rackoff* model and some generic proofs of security are available [62]. Practical examples of Feistel ciphers are the *DES* (Data Encryption Standard [72]), *Blowfish* [80], *FEAL* [83] and many more. *Unbalanced* Feistel ciphers are a variant structure where L_i and R_i are not of equal lengths, as in the case of *Skipjack* [1].

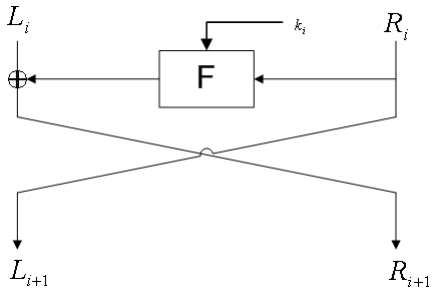

Fig. 1.2: One round of a Feistel network

In the Substitution-Permutation Network architecture, the full block of input is transformed by the iterative application of the successive transformations constituting the round function (see fig.1.3). Contrary to the Feistel construction, the round function F (and therefore its constitutive transformations) must be invertible in order to allow decryption. In addition to the round function, a key mixing is usually applied at the begining of the encryption, as otherwise the first few transformations before the key mixing could easily be undone. This process is called *key whitening* [49]. One of the main advantages of SPN ciphers are their simple and elegant design that facilitates their design and

analysis. Among others, examples of SPN ciphers include the *AES* (Advanced Encryption Standard [30]), Serpent [6], PRESENT [23].

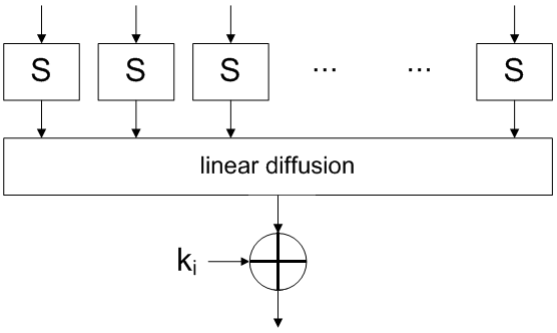

Fig. 1.3: One round of a Substitution-Permutation network

The previous discussion illustrates that several block ciphers exist in the literature, which raises the question of how to design or select them. Quite naturally, different criteria can be considered for this purpose. For example, following the Advanced Encryption Standard selection process [73], one could take into account:

- The availability of the algorihm in the public domain
- The security of its design
- The performances on hardware/software platforms
- The possibility to protect implementation against physical attacks
- ...

In general, different optimization criteria give rise to different trade-offs. For example, performance in terms of throughput can always be increased at the cost of a possible reduction of the security margins.

1.3 Cryptanalysis

Cryptanalysis is the science devoted to the analysis of algorithm security. In other words, the goal of a cryptanalyst is to scrutinize algorithms in order to detect possible weaknesses or attacks. Classically, three features characterize a cryptanalysis against a block cipher:

1. The objective of the attack in terms of information recovered
2. The context in which the adversary performs the attack

3. The complexity of the attack

According to the classification proposed by Lars Knudsen [52], attacks on block ciphers can be divided into five categories depending on the secret information recovered:

- Distinguishing algorithm: the attacker can distinguish the cipher from a random permutation.
- Information deduction: the attacker gains some Shannon information about plaintexts (or ciphertexts) not previously known.
- Instance deduction: the attacker discovers additional plaintexts (or ciphertexts) not previously known.
- Global deduction: the attacker discovers a functionally equivalent algorithm for encryption and decryption, but without learning the key.
- Total break: the attacker recovers the secret key.

Note that, depending on the design, some of these goals can be strongly related. For example, in iterated block ciphers, a distinguishing attack can usually be turned into a key recovery attack, by using a standard "divide and conquer" key guessing strategy [2].

In cryptanalysis, one generally assumes that the specifications of the algorithm are known to the attacker (i.e. only the key has to be kept secret): this is Kerckhoffs' principle of *the enemy knows the system* [86]. Historically, the opposite conception of *security through obscurity* has proved exceedingly difficult to maintain due to espionage, leakage and reverse engineering among others. In addition to this assumption, the adversarial scenario defines the types of attack employed according to the kind of acces given to the attacker:

- Ciphertext-only: the attacker has only access to a set of ciphertexts.
- Known-plaintext: the attacker has a set of pairs of plaintext and the corresponding ciphertext.
- Chosen-plaintext: the attacker can obtain the ciphertexts corresponding to an arbitrary set of plaintexts of his choice.
- Chosen-ciphertext: the attacker can obtain the plaintexts corresponding to an arbitrary set of ciphertexts of his choice.
- Adaptive chosen-plaintext: like a chosen-plaintext attack, but the attacker can choose subsequent plaintexts based on information learned from previous queries.

- Adaptive chosen-ciphertext: like a chosen-ciphertext attack, but the attacker can choose subsequent ciphertexts based on information learned from previous queries.
- Related-key attack: like a chosen-plaintext attack, except the attacker can obtain ciphertexts encrypted under two (or more) different keys. The keys are unknown, but the relationship between them is known.

The goal of a block cipher designer is to prevent the existence of successful attack, even in strong adversarial scenarios, e.g. with related key queries. Beside the type of data available, the evaluation of a cryptanalysis is also determined by three factors:

- The *data complexity* (the expected number of queries required)
- The *time complexity* (the expected number of operations required to process the data)
- The *memory complexity* (the expected number of storage units required to perform a successful attack)

In the case of block cipher, it is always possible to recover the secret key given one (or a few) pair of plaintex-ciphertext, by encrypting the plaintext under every possible k-bit key guess and then select the guess that gives the correct ciphertext. If more than one key is found, an additional pair of texts is usually sufficient to decide between those candidates. This generic attack is called *exhaustive search* or *brute-force attack* and requires at most 2^k encryptions for an k-bit key. In the context of cryptanalysis, one generally refers to attacks for any technique allowing to recover some previously unknown piece of information with smaller complexity than exhaustive key search. This again allows various tradeoffs between data, time and memory.

The design of a block cipher is intrinsically related to its cryptanalysis: a *certificational attack*, even if it does not endanger practical application of the cipher, often suggests weaknesses in its conception. That is why any new block cipher proposal is expected to demonstrate immunity against all known attacks.

This thesis investigates one important category of statistical attacks against block ciphers, denoted as *linear cryptanalysis*, and some of its possible extensions. For this purpose, the next chapter starts with a brief state-of-the-art on the cryptanalysis of block ciphers. Then, in the third chapter, we use this state-of-the-art as a motivation for introducing and summarizing the different contributions of this thesis.

2. STATE OF THE ART IN CRYPTANALYSIS

2.1 Introduction

In this section, we briefly present different cryptanalysis techniques that are frequently used in the evaluation of the security of modern block ciphers, namely:

- Differential cryptanalysis and its extensions:
  - Truncated differentials
  - Higher order differentials
  - Impossible differentials
- Linear cryptanalysis and its extensions:
  - Multiple linear approximations
  - Differential-linear cryptanalysis
  - Non-linear cryptanalysis
- Boomerang attacks,
- Interpolation attacks,
- Related-key attacks,
- Slide attacks,
- Saturation attacks,
- Partitioning cryptanalysis,
- Cube attacks,
- Algebraic cryptanalysis

2.2 Differential cryptanalysis

Historically, differential cryptanalysis was one of the first statistical cryptanalysis techniques efficient for a wide variety of block ciphers. It was presented by Biham and Shamir [15] at the Crypto'90 conference. Initially developed against the DES, the technique was successfully applied against a large number of block ciphers, stream ciphers and even hash functions.

Unlike other ciphers developed around the same period (e.g. FEAL [16]), the DES cipher proved to be surprisingly resistant against differential attacks. Since then, its designers at IBM have admitted the knowledge of this technique as early as in the seventies but the *National Security Agency* had requested confidentiality. At IBM, it was then known as the *tickling attack*, in reference to its principle which consisted to *tickle* the input in order to observe the effect on the output [27].

Differential Cryptanalysis is generally a chosen plaintext attack, even though some variants are known-plaintext or even ciphertext-only attacks [19]. In summary, the differential cryptanalysis evaluates how differences in plaintext pairs propagate through the encryption process to generate differences in the corresponding ciphertexts. If an attacker can find a difference pattern that holds with a high probability, then he can use this to recover several bits of information about the key.

A differential cryptanalysis can be decomposed into three steps:

First, to efficiently find good differentials the attacker generally starts by analyzing separately the components of the cipher. For linear transformations such as permutations and *XOR* additions, a fixed input difference affects the difference in output in a deterministic way. For non-linear components such as S-boxes however, differences in input can lead to several differences in output, each one with some probability p. For a n-bit input, m-bit output S-box, there are 2^n possible input differences and 2^m output differences are potentially possible for any input difference. As the size of a S-box is usually small, it is possible to exhaustively investigate every possible pair $(\Delta I, \Delta O)$ of (input,output) difference and compute the probability of occurence of each combination for the S-box denoted as:

$$\Delta I \xrightarrow{p} \Delta O \tag{2.1}$$

Such a pair is called an *S-box differential* and the corresponding S-box is said to be *active*.

Secondly, S-box differentials can be extended from one round to the next through the linear components in order to define a *differential characteristic* that spans on several rounds. If we assume that occurences of S-box differentials are independent for different active S-boxes, the differential characteristic

probability can be computed as:

$$p = \prod_{n}^{i=1} p_i \tag{2.2}$$

where n is the number of active S-boxes and p_i is the probability of the S-box differential for the i^{th} active S-box.

The third and last step exploits an R-round differential characteristic to recover some information about the cipher. It can be used to distinguish R rounds of the cipher from a random permutation by observing that, for a sufficient number of plaintext pairs with the given input difference, a significant number (about 1 every $1/p$ pairs) of ciphertext pairs have the corresponding output difference, which suggests a non-ideal behavior for a cipher. Then, such distinguisher can be extended to a key recovery attack on $R + 1$ rounds using a "divide and conquer" strategy illustrated in figure 2.1. In this setting, the R-round characteristic should ideally connect to a limited number of active S-boxes in the round $R+1$ so that, by guessing only a few bits of the last subkey, it is possible to perform a partial decryption of the ciphetext pairs for these S-boxes. For every key hypothesis, the attacker then evaluates the probability that the R-round differential characteristic holds and select the key with the highest probability.

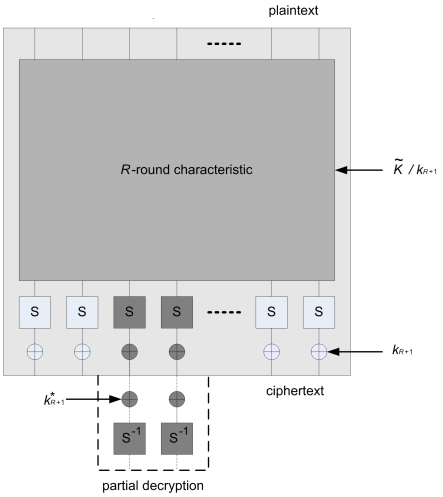

Fig. 2.1: The "divide and conquer" approach

The data complexity of the attack depends crucially on the probability p of the differential characteristic $(\Delta I, \Delta O)$ used. Let us define as a *right pair* any pair of plaintexts with difference ΔI leading to a difference ΔO after R rounds (thus after partial decryption with a key guess). For the correct key guess, a

right pair should occur with probability p, while for a wrong candidate, the probability of occurence should be close to 2^{-n} since every output difference is expected to be equiprobable in this case. Consequently, to distinguish the correct key, an estimation of the number of pairs required is [15]:

$$N \approx c/p \tag{2.3}$$

where c is a small constant and p should be significantly higher than 2^{-n}.

However, the precise evaluation of this probability is not straightforward. To develop this point, we need to introduce two specific notions related to the concept of differential characteristic:

- A *differential* on R rounds is pair constituted by an input difference Δ_0 and an output difference Δ_R.
- A *characteristic* on R rounds is a sequence of R+1 differences $(\Delta_0, \Delta_1, ..., \Delta_R)$ where two consecutive values describe the differential on one round.

Many characteristics usually correspond to the same differential, each characteristic contributing to its global probability of occurence. Therefore, the data complexity of differential cryptanalysis derived from the probability of one characterisic found on the cipher is an upper bound. In order to prove security of a block cipher against differential cryptanalysis, it is necessary to show that any differential remains below some acceptable threshold. However, the exact evaluation of the probability of a differential remains an open problem.

Finally, as a countermeasure against differential cryptanalysis, a block cipher designer can decrease the probability of every differential characteristic. Among others, this can be done in two ways:

1. Select S-boxes and other non-linear components so that difference propagations have a probability as low as possible.
2. Increase the diffusion in the cipher so that the number of active S-boxes and other non-linear components in any differential is as high as possible.

This approach was successfully employed in the *wide trail strategy* [31] during the conception of the Advance Encryption Standard *Rijndael* [30].

Truncated differentials

Truncated differentials were proposed by Knudsen at FSE 1994 [51]. In this paper, Knudsen introduces the concept of differentials where only part of the difference in the ciphertexts can be predicted. The advantage is that some functions are secure against differential cryptanalysis, but not against such truncated differentials.

Among others, this technique has been applied against SAFER [54], IDEA [24], Skipjack [57]. Truncated differentials are more efficient against ciphers where transformations operate on aligned blocks of bytes rather than on bits. Binary permutations can be used as an efficient countermeasure against this kind of attack [30, 78].

Higher order differentials

In the same paper, Knudsen also introduces the notion of higher order differentials as a generalisation of the concept of differentials. For a boolean function f defined on the point a, we define a *first order derivative* as:

$$\Delta_a f(x) = f(x+a) - f(x) \tag{2.4}$$

Then, the i^{th} derivative of f at the points $a_1, a_2, a_3, ...a_i$ can be defined recursively as:

$$\Delta^i_{a_1,...,a_i} f(x) = \Delta a_i(\Delta^{i-1}_{a_1,...a_{i-1}} f(x)) \tag{2.5}$$

A one round differential of order i is an $(i+1)$-tuple $(a_1, ..., a_i, b)$ such that $\Delta^i_{a_1,...,a_i} f(x) = b$.

Knudsen showed in [51] that there exists ciphers provably secure against first order differentials (i.e. against differential cryptanalysis) but not against second order differentials. However, the attack seems limited to a reduced number of rounds because iterating higher order differentials for more than 2 rounds has proved to be challenging.

Impossible differentials

Instead of exploiting differential characteristics with probability higher than average, impossible differential cryptanalysis takes advantage of differentials with zero probability of occurrence.

The technique used to generate such impossible differentials is based on the *Miss in the Middle* approach presented by Biham, Biryukov, and Shamir at the CRYPTO 1998 rump session. It consists in finding two differentials of probability one, $(\Delta_{r_0}, \Delta_{r_1})$ and $(\Delta^*_{r_1}, \Delta_{r_2})$, that cannot be combined together. Usually, the attacker finds a first differential (or truncated differential) that has probability 1 for the R_1 first round of the cipher, followed by a second differential with probability 1 that start at the end of round R_1 and finishes at round R_2. If the differences Δ_{r_1} and $\Delta^*_{r_1}$ at the meeting place are not compatible, then the complete differential $(\Delta_{r_0}, \Delta_{r_2})$ is impossible, hence the name *miss in the middle*.

Once an attacker has found an impossible differential, he can use it in two ways:

1. *Distinguishing attack*: in this case, the attacker can distinguish the cipher from a random permutation by requesting a large number of ciphertext pairs corresponding to plaintext pairs with the fixed difference Δ_{r_0} and checking if the impossible difference occurs.

2. *Sieving attack*: in this case, the attacker adds some rounds to the impossible differential, and starts to encrypt pairs of plaintexts with the given input difference. For all the pairs of corresponding ciphertexts, the attack performs a partial decryption and rejects all the keys that lead to the impossible differential. Therefore, the correct key is found by eliminating all the other keys.

This technique was used to break many reduced-round variants of block ciphers, including 4.5 out of 8.5 rounds of IDEA [8] and 31 out of 32 rounds of the cipher Skipjack [7].

2.3 Linear cryptanalysis

Linear cryptanalysis is one of the most powerful attacks against block ciphers. It was proposed by Mitsuru Matsui at Eurocrypt 1992 [66]. The development of linear cryptanalysis led to the first successful attack against the complete DES block cipher [64, 65]. In its basic version, linear cryptanalysis is a known-plaintext attack that exploits a linear approximation of the block cipher of the following form:

$$P[\chi_P] \oplus C[\chi_C] = K[\chi_K], \tag{2.6}$$

where P, C and K respectively denote the plaintext, ciphertext and the (expanded) secret key while $A[\chi]$ stands for $A_{a_1} \oplus A_{a_2} \oplus ... \oplus A_{a_n}$, with $A_{a_1}, ..., A_{a_n}$ representing particular bits of A in positions $a_1, ..., a_n$ (χ is usually denoted as a mask). To be effective, such approximation must hold with probability significantly different from 1/2.

The principle of linear cryptanalysis is similar to differential cryptanalysis and can be equally decomposed in three steps:

The first step of the attack is to generate efficient linear approximations through a careful analysis of the components of the cipher. Specifically, the non-linear parts of a cipher generally constituted by layers of S-boxes are the critical factor for the attack. The basic principle is to approximate these S-boxes by linear expressions: for a n-bit input, m-bit output S-box, such linear expression can be defined by a pair of (n, m)-bit masks that specifies which bits are XORed together. Consequently, there are $(2^n - 1) * (2^m - 1)$ possible linear approximations of the S-box (masks with no input bit and/or no output bit are not considered). The cryptanalyst exhaustively evaluates the probability that these approximations hold by checking the 2^n possible inputs. If α is

the number of times a linear approximation holds, the resulting probability is $p = \alpha/2^n$ and the corresponding *bias* is defined as $\epsilon = p - 1/2$. The S-boxes involved in the linear approximation are referred to as *active* and have a non-zero bias.

Once linear approximations of the S-boxes have been generated, the second step consists in combining and propagating them from the input to the output through the whole cipher so that the global linear approximation only involves bits of the plaintext, the ciphertext and the key. Provided that the S-box approximations hold independently, the bias of a linear approximation constituted of several active S-boxes can then be estimated thanks to the *piling-up lemma* [64]:

$$\epsilon = 2^{n-1} \prod_{i=1}^{n} \epsilon_i \tag{2.7}$$

where n is the number of active S-boxes and ϵ_i is the bias of the S-box approximation for the i^{th} active S-box.

Equation 2.7 underlines that in order to have a high bias (in absolute value), one must reduce the number of active S-boxes and select S-box approximations with the highest bias. In order to find the best linear approximations of a block cipher, Matsui proposed a branch-and-bound algorithm that works by induction: knowing the value of maximum bias on the $R-1$ first rounds, it manages to find the maximum bias on R rounds as well as the corresponding input and output masks.

The last step of the cryptanalysis exploits the linear approximation coupled with a large set of plaintext-ciphertext pairs in order to recover some secret information. In his original paper, Matsui described two methods: a distinguishing attack denoted as *algorithm 1*, and a key recovery attack denoted as *algorithm 2*:

- Given an R-round linear approximation with sufficient bias, the algorithm 1 simply counts the number of times T that the left side of equation 2.6 is equal to zero for N (plaintext, ciphertext) pairs. If $T > N/2$, then it assumes either $K[\chi_K] = 0$ if $\epsilon > 0$ or $K[\chi_K] = 1$ if $\epsilon < 0$ so that the experimental value $(T - N/2)/N$ matches the theoretical bias. If $T > N/2$, an opposite reasoning holds. For the attack to be successful, it is shown in [66] that the number of available pairs must be proportional to $\frac{1}{\epsilon^2}$.
- The second algorithm is based on the "divide and conquer" approach illustrated in figure 2.1 for differential cryptanalysis. In this case, the attacker targets $R+1$ rounds of the cipher and performs a partial decryption of the last round in order to evaluate the R-round linear approximation for each key guess. As a consequence, all the guessed key bits can be recovered rather than the parity $K[\chi_K]$ which yields much more efficient

attacks in practice. In this case, it is more difficult to derive bounds for the complexity of the attack. For this reason, the same complexity order as for *algorithm 1* is usually assumed in the literature.

As in the differential case, multiple linear approximations sharing the same input and output masks but different intermediate S-box approximations also coexist in linear cryptanalysis. The notion of linear hull was introduced by Kaisa Nyberg in [74] to describe and analyse this phenomenon. The resulting *linear hull effect* further increases the difficulty to evaluate the data complexity of a cryptanalysis based on linear approximations. Indeed, the efficiency of linear cryptanalysis (and other techniques!) in terms of complexity and probability of success is generally theoretically evaluated under some basic assumptions. Among the usual hypothesis used, we can mainly cite:

- The *key equivalence hypothesis* [42] assumes that the linear bias will be close to its average value for the vast majority of keys.
- The *key independace hypothesis* [55] assumes that round keys can be considered as independent provided that the key schedule alorithm is sufficiently complex.
- The *wrong key randomization hypothesis* [42] states that partial decryption with a wrong key guess is equivalent to an additional round encryption and consequently decreases the (key-dependent) bias.

Interestingly, the impact of the linear hull on these hypotheses is not well understood and may be computationally difficult to assess.

Finally, in order to increase the resistance of a block cipher against linear cryptanalysis, a designer can decrease the probability of every linear approximations. This can be done in two ways:

1. Select S-boxes and other non-linear components so that linear probabilities are as low as possible.
2. Increase the diffusion in the cipher so that the number of active S-boxes and other non-linear components in any linear approximation is as high as possible.

This is again the approach employed in the *wide trail strategy* [31].

Multiple linear approximations

Among the various proposals to improve the linear cryptanalysis of block ciphers, Kaliski and Robshaw proposed an algorithm using several linear approximations [46] in 1994.

Their idea was further improved by Biryukov, De Cannière and Quisquater in 2004 [17]. Let us suppose that one has access to m approximations on R block cipher rounds of the form:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \ (1 \leq i \leq m), \tag{2.8}$$

and wishes to determine the value of the binary vector:

$$\mathbf{Z} = (z_1, z_2, ..., z_m) = (K[\chi_K^1], K[\chi_K^2], ..., K[\chi_K^m]) \tag{2.9}$$

The improved algorithm associates a counter T_i with each approximation that is incremented each time the corresponding linear approximation is verified for a particular pair (plaintext-ciphertext). As for algorithm 1, the values of $K[\chi_K^i]$ are determined from the experimental biases $(T^i - N/2)/N$ and the theoretical bias ϵ_i by means of a maximum likelihood rule. The extension of Matsui's algorithm 2 to the multiple approximations case can be similarly described.

Another extension called *multidimensional cryptanalysis* [25] evaluates the distribution of the approximations in a multidimensional way for improved evaluation. Additionally, the distances between distributions are measured using Kullback-Leibler distance instead of the sum of the squared correlations. An adaptation of the technique for Matsui's algorithm 2 is also proposed in [44].

The use of multiple approximations aims at decreasing the data complexity of linear cryptanalysis by exploiting more thoroughly the statistical information brought by each plaintext ciphertext pair. However this approach raises several new open questions as the evaluation of the data complexity in this case is not well understood. In particular, it is not exactly known how linear dependencies between the approximations used affects the efficiency of the attack.

Differential-linear cryptanalysis

As the name suggests, this attack combines the *differential* and *linear* techniques together [60]. The original attack against 8 rounds of DES performs significatively better than any other attack, requiring only 512 chosen plaintexts for 80% success rate. It uses a 3-round linear approximation on top of a 3-round differential characteristic which holds with probability 1. One additional round is added at the begining and another is added at the end. The attack exploits the fact that, for any plaintext pair with the correct difference in input, the corresponding bits at the input of the linear approximation are the same for both plaintexts. Consequently, the linear approximation is more likely either to hold for both texts or for none than to hold for one text only. This property is used in conjunction with a partial decryption of the first and last rounds to recover 10 bits of the key.

In [10], Biham *et al.* generalized this attack for differential characteristics with probability lower than 1. This allowed a differential-linear attack against

9 rounds of the DES. In this case however, the data complexity is much higher.

Non-linear cryptanalysis

Non-linear cryptanalysis decreases the number of plaintexts required to cryptanalyse a block cipher by using *non-linear* approximations of the S-boxes with higher bias [56]. For example, there exists a simple non-linear approximation of a DES S-box with bias 28/64 instead of 20/64 for the best linear bias.

The difficulty however consists in joining non-linear approximations between consecutive rounds. Practically, to extend the approximations across an S-box to the entire round, additional key bits need to be guessed. To illustrate this, we can observe that before an S-box layer, any input bit x_i is combined with a subkey bit k_i to give z_i which is the input to the S-box. For example, the non-linear combination of bits $x_i x_j$ becomes $(z_i \oplus k_i)(z_j \oplus k_j) = z_i z_j \oplus z_i k_j \oplus z_j k_i \oplus k_i k_j$ and is therefore key dependent contrary to the linear case where $x_i \oplus x_j = (z_i \oplus k_i) \oplus (z_j \oplus k_j) = z_i \oplus z_j \oplus k_i \oplus k_j$. Consequently, in non-linear cryptanalysis not only the sign of the bias depends on the key but also its value.

Despite this difficulty, non-linear approximations can still be used in three ways:

1. The inputs or outputs to an approximation in the first or last round need not to be combined with other approximations. Consequently, approximations of these rounds can be non-linear.

2. In the case of cryptanalysis with partial decryption in the first and/or last round, some bits in the second and/or penultimate round are available to the attacker. Non-linear approximations can also be used in these rounds.

3. Key dependency of non-linear approximations can be used to perform another kind of key guess in the last (or first) rounds of block ciphers. It allows more flexibility than classical cryptanalysis.

2.4 Boomerang attack

The Boomerang attack is another attack based on differentials that was proposed by David Wagner at *FSE 1999* [87]. The attack performs better than differential cryptanalysis whenever the best differential on the whole cipher has a probability much lower than differentials on a reduced-round version.

The basic attack works as follows: suppose that the block cipher encryption operation $E(\cdot)$ can be decomposed into two parts such that $E(\cdot) = E_1 \circ E_0 = E_1(E_0(\cdot))$, where E_0 represents the first half of the cipher and E_1 represents the last half. Suppose also that there exist efficient differentials $\Delta \rightarrow \Delta^*$ for E_0 with probability p and $\nabla \rightarrow \nabla^*$ for E_1^{-1} with probability q. We start with 2 plaintexts P_1 and $P_2 = P_1 \oplus \Delta$, and request their corresponding ciphers C_1 and C_2. Then, we take the two ciphertexts $C_3 = C_1 \oplus \nabla$ and $C_4 = C_2 \oplus \nabla$

and request their decrypted plaintexts P_3 and P_4. If the difference patterns $\Delta \xrightarrow{p} \Delta^*$ and $\nabla \xrightarrow{q} \nabla^*$ happened as expected between E_0 and E_1, then:

$$\begin{aligned} E_0(P_3) \oplus E_0(P_4) &= E_0(P_1) \oplus E_0(P_2) \oplus E_0(P_1) \oplus E_0(P_3) \oplus E_0(P_2) \oplus E_0(P_4) \\ &= E_0(P_1) \oplus E_0(P_2) \oplus E_1^{-1}(C_1) \oplus E_1^{-1}(C_3) \oplus E_1^{-1}(C_2) \oplus E_1^{-1}(C_4) \\ &= \Delta^* \oplus \nabla^* \oplus \nabla^* \\ &= \Delta^* \end{aligned}$$

which in turn holds with probability p^2q^2 if the four differentials hold independently. Figure 2.2 illustrates this property.

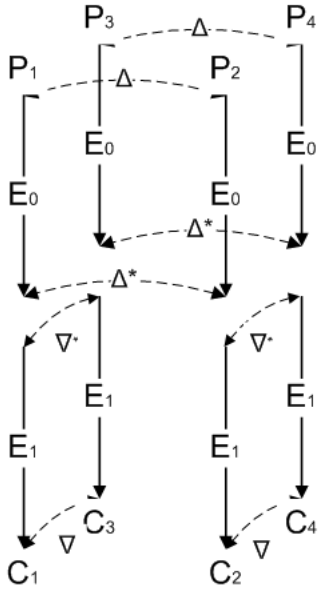

Fig. 2.2: Propagation of differentials in a Boomerang attack

If the best differential on the block cipher has a probability $m >> 2^{-n}$, a differential cryptanalysis of data complexity $O(m^{-1})$ can be successful against the cipher. On the other hand, if the best differentials for the two halfs of the cipher have a probability p and q, then a boomerang attack requiring $O(p^{-2}q^{-2})$ can be implemented. Consequently, if $p^{-2}q^{-2} << m^{-1}$, the boomerang attack is more efficient than differential cryptanalysis.

The boomerang attack was first demonstrated against the block cipher COCONUT98 [84]. This attack is specially significant since this cipher was designed to be provably secure against differential attacks. Despite the inexistence of good differentials for the whole cipher, Wagner observed that there were differentials with probability $\approx 2^{-4.3}$ for half of the cipher. These differentials led to a key recovery attack against the full cipher requiring 2^{16} adaptive chosen-plaintexts and ciphertexts and a time complexity of 2^{38}.

At *FSE 2000*, Kelsey, Kohno and Schneier proposed an extension of boomerang attacks dubbed as *amplified boomerang attacks* [47] that requires only chosen plaintext queries. This extension takes advantage of collisions that occur after E_0 if a sufficient number of plaintexts is encrypted. According to the birthday paradox, by requesting $2^{n/2}$ pair of plaintexts $(P_{2i},\ P_{2i+1})$ such that $P_{2i} \oplus P_{2i+1} = \Delta$ it is possible to find some indices (i, j) such that $E_0(P_{2i}) \oplus E_0(P_{2j}) = \Delta$ and $E_0(P_{2i+1}) \oplus E_0(P_{2j+1}) = \Delta$. These differences then propagate through E_1 and $E_1(E_0(P_i)) \oplus E_1(E_0(P_j)) = \Delta^*$ with high probability. Biham, Dunkelman and Keller showed in the *rectangle attack* [9] that additional improvements could significantly increase the proportion of right quartets.

2.5 Interpolation attacks

The interpolation attack [45] can be applied against ciphers for which the round function can be written as a simple algebraic expression. It is based on the *Lagrange Interpolation Formula* that can be stated as follows:

Let R be a field. Given $2n$ elements $x_l, \ldots, x_n, y_1, \ldots, yn \in R$, where the x_i are distinct. Define

$$f(x) = \sum_{i=1}^{n} y_i \prod_{1 \le j \le n, j \neq n} \frac{(x - x_j)}{x_i - x_j} \tag{2.10}$$

Then $f(x)$ is the only polynomial over R of degree at most $n - 1$ such that $f(x_i) = y_i$ for $i = 1, \ldots, n$.

In the interpolation attack, the cryptanalyst constructs a polynomial f mimicking the cipher with the help of a subset of plaintext-ciphertext pairs. It is a good example of *global deduction* attack, which means that an equivalent for the cipher is established, but the secret key is not recovered. However, the attack can also be converted into a key recovery attack using the "divide and conquer" approach presented previously. If the number of coefficients in the polynomial is m and the block size of the cipher is n, then for $m < 2^n$ there exists an interpolation attack of time complexity m requiring m plaintext-ciphertext pairs

By combining the attack with a meet-in-the middle approach, it is also possible to reduce the number of terms in the polynomial, thus the data complexity.

2.6 Related key cryptanalysis

A related key attack is a kind of attack where an adversary can request encryptions of plaintexts under different (unknown) keys that are connected by

a (known) mathematical relationship. At first glance such a technique may look unrealistic in practice, however cryptographic algorithms are often embedded in complex protocols where such relationships between secret keys can inadvertantly occur. A significant example of protocol that failed because of related key attacks is the *Wired Equivalent Privacy* (WEP) protocol used in wifi wireless networks. The attack exploits weaknesses in the keyschedule of the stream cipher *RC4* which is used to encrypt the data [39].

In the seminal paper on related key attacks [4], Biham targets specific key schedule algorithms where 2 master keys K, K^* are said to be related if their subkeys K_i, K_i^* are such that $K_i^* = K_{i+1}$ for several rounds. For two such related keys, if the data before the first round under the key K^* equals the data before the second round under the key K, then both data and subkeys are the same with a difference of one round. By using relations holding between P, P^*, C and C^*, and the related keys K, K^*, it is then possible to recover some bits of the secret keys.

Related key attacks have been applied against a broad variety of ciphers including *LOKI* [4], *3-WAY*, *RC2*, *TEA* [48], *SHACAL-1* [14], *KASUMI* [12] and even the *AES* [18]. In addition, related key attacks can be combined with other statistical attacks such as *impossible differential cryptanalysis* [13], *boomerang* and *rectangle attacks* [11]

Related key attacks remain mostly theoretical, but they can also have an impact on the security of hash functions based on block ciphers. In order to avoid such attacks, it is important to carefully design the key scheduling algorithm of the block cipher. First of all, designers should avoid linear key schedules. More generally, the use of round constants efficiently thwarts the application of sliding techniques by making the key schedule different in every round.

2.7 Slide attacks

Slide attacks targets iterated block ciphers using the same round function and a periodic key schedule. The idea for the attack comes from a paper by Grossman and Tuckerman at IBM in 1977 [41]. It was then refined by Biryukov and Wagner who introduced the name of the attack at FSE 1999 [21]. Slide attacks are similar to Biham's related-key attacks but in a more realistic adversarial scenario.

For many iterative block ciphers with weak round function, it is relatively easy to break one round of the cipher given one pair of plaintext-ciphertext for the round. If in addition the cipher uses a periodic key schedule where the subkeys are periodicaly repeated, then the cipher is prone to slide attacks using a *slid pair*: *Let F be the round function of an iterated block cipher. If two pairs of known plaintext $(P, C), (P^*, C^*)$ satisfy $P^* = F(P)$, then due to the self-similarity of both the rounds and the key schedule, the corresponding*

ciphertexts also satisfy $C^* = F(C)$. *Such a pair is called a* slid pair.

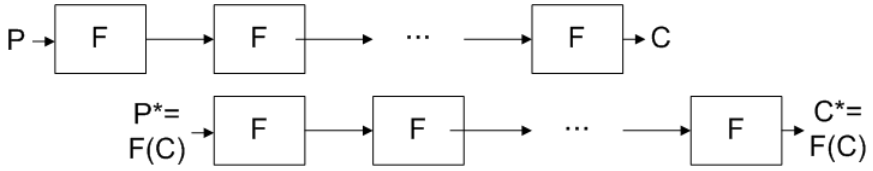

Fig. 2.3: a slid pair

The basic attack works as follows (see fig. 2.3): first, an attacker generates $O(2^{n/2})$ plaintext-ciphertext pairs. By the birthday paradox, there is a slid pair $((P, C), (P^*, C^*))$ among all the possible combinations with high probability. Such a slid pair can be detected easily if the function F is weak, by checking that $P^* = F(P)$ and $C^* = F(C)$. Once a slid pair is available, recovering the key is as difficult as breaking one round of the cipher. For Feistel ciphers in particular, the round function modifies only half of the input. Thus, a slid pair can be recognized simply by comparing the left half of P with the right half of P^*. In this case, the time complexity can be reduced from 2^n to $2^{n/2}$ and the data complexity can be reduced to $2^{n/4}$ chosen plaintexts.

One of the major strength of slide attacks is that the number of rounds in the cipher becomes irrelevant. However, as this kind of cryptanalysis exploits similarity in the round function for different rounds of the cipher, it is easy to provide resistance by using round constants and by avoiding periodic key scheduling algorithms.

Advanced techniques can be used to attack a broader range of Feistel ciphers, namely *p-round self-similar* block ciphers, where a generalized round consisting of p rounds of the original cipher is iterated. Two techniques are proposed in [22] when $p = 2$:

1. *Complementation slide*: A possibility to deal with 2-round self-similarity would be to slide by 2 rounds. In this case however, the generalised round function made of two rounds may be difficult to detect and break. The complementation slide technique consists in sliding by only one round, then choosing a slid pair so that the plaintexts difference cancels the difference between the two different subkeys k_1 and k_2. Thus, the slid pair is chosen according to the equations: $P^* = F(P) + \Delta$ and $C^* = F(C) + \Delta$ where $\Delta = k_1 \oplus k_2$.

2. *Sliding with a twist*: If the final swap is ignored, then the decryption of a Feistel Cipher under key k_1, k_2 is the same as encryption with key k_2, k_1. Therefore, it is possible to slide by one round a decryption process against an encryption process.

Finally, by combining the two approaches, it is possible to target 4-round self-similar Feistel ciphers.

2.8 Saturation attacks

The first use of saturation attacks appeared in the paper introducing the block cipher SQUARE [29], where it was called *Square attack* for obvious reasons. It is also called *integral* or *multiset attack* in some subsequent papers [58, 75, 90, 20, 71]. The saturation attack is a chosen plaintext attack that exploits the structure of the cipher independently of the choice of the S-boxes and key schedule. It works by analysing the propagation of plaintexts through the cipher when some plaintext bytes are fixed while others are *saturated*, i.e. vary throughout all possible values.

The saturation attack dedicated to the cipher SQUARE is a good illustration of the attack principles. SQUARE is a 128-bit iterated block cipher with a round function composed of 4 invertible transformations that operate on a 4×4 array of bytes called the *state*:

1. A linear transformation θ that operates separately on each of the four rows of the state (except in the first round).
2. An S-box γ that applies individually to each byte of the state.
3. A byte permutation π transposing rows and column of the state.
4. A bitwise subkey addition σ.

The i^{th} round function $\rho[k^i]$ is obtained by the following composition of the building blocks: $\rho[k^i] = \sigma[k^i] \circ \pi \circ \gamma \circ \theta$. The key schedule algorithm is not relevant for the attack.

The saturation attack applied on SQUARE makes use of a set of 256 plaintexts which are all different for a particular byte (called *active* or *saturated*) in position λ of the state, and all equal for the 15 other bytes of the state (called *passive*). The designers of SQUARE call such a set a $\Lambda - set$. Figure 2.4 illustrates the propagation of the active bytes (in light gray) in the state during the encryption. Starting with a $\Lambda - set$, the attacker observes the propagation of the active bytes through the state encryption and uses subsequent properties to distinguish the cipher from a random permutation. The 1st round contains no θ transformation, hence there is still only one byte active at the beginning of the 2nd round. Having a $\Lambda - set$ in the input of the second round (fig. 2.4(a)), the only active byte is converted into a row of 4 active bytes after the θ transformation, which in turn is transformed into a column of active bytes after the π transposition (fig. 2.4(b)). At the begining of the third round, this column of active bytes is propagated to the whole state after the θ transform, and all the bytes remain active until the begining of the fourth round (fig. 2.4(c)). After the θ transformation in the fourth round, the bytes of the states are not necessarily active anymore. However, as they are linear combinations of active bytes, the sum over the 256 value of any byte of the state is equal to zero (i.e.

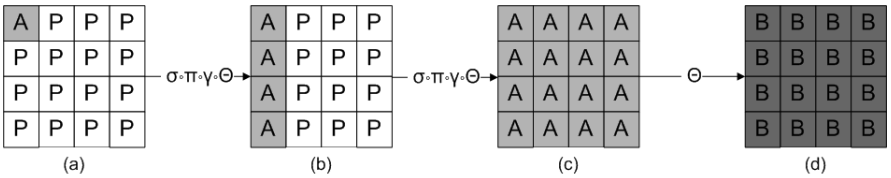

Fig. 2.4: Propagation of the active byte in the Square attack. A denotes the bytes that are *active*, P denotes the bytes that are *passive* and B denotes the bytes that are *balanced*.

the bytes are *balanced*, see fig. 2.4(d)). This particular property is lost after the application of the following γ transform.

As a consequence of the previous observation, it is possible to distinguish 4 rounds of SQUARE from a random permutation, by performing a partial decryption with one byte of subkey key in the fourth round and then testing for balancedness. By guessing more bytes, it is possible to target up to six rounds of the block cipher SQUARE with 2^{32} plaintexts and 2^{72} operations. This time complexity was further reduced to 2^{44} in [38]. As the AES cipher is inspired by the SQUARE cipher and inherits many of its properties, a very similar attack can also target six rounds of the AES [30]. Gilbert and Minier presented an improved version of the attack against 7 rounds which exploits collisions between some partial functions induced by the cipher. The attack requires 2^{32} chosen-plaintexts and is marginally faster than an exhaustive search [40].

Forms of saturation cryptanalysis have been applied to a variety of ciphers, including IDEA [33], Camellia [89], Skipjack [32], SAFER++ [75], KHAZAD [59] and others. The *SASAS* cryptanalysis from Byriukov and Shamir [20] considers the security of block ciphers which contain alternate layers of invertible S-boxes S and affine mappings A using structural cryptanalysis combined with linear algebra.

Although saturation cryptanlysis is best suited against block ciphers operating on blocks of data (like SQUARE), it has also been used with some success against bit-based ciphers such as Twofish [63]. At *FSE 2008*, Muhammad Reza Z'aba *et al.* proposed an extension of the saturation attack called the *Bit-Pattern Based Integral Attack* [90] which also aims at bit-based ciphers.

2.9 Partitioning cryptanalysis

The following cryptanalysis technique is based on the idea that for some partitions of the plaintexts, ciphertexts can be distributed non uniformly during the encryption process, hence the name *partitioning cryptanalysis* [43]. As a trivial example, for a Feistel cipher, the left output half after one round is simply equal to the right input half: $L_{i+1} = R_i$. Therefore, if the partition of

the plaintexts is performed according to the value of the right part R_i, then for a partition of the ciphertexts according to the left part L_{i+1}, there will be a strong non-uniform distribution from the input to the output: all the plaintexts from a given subset of the input partition will be directed to the same subset of the ciphertext partition. Partitioning cryptanalysis can be seen as a generalisation of linear cryptanalysis [42].

The attack makes use of the notion of *partition pair*:

Let $F = F_0, F_1, ..., F_{l-1}$ and $G = G_0, G_1, ..., G_{m-1}$ be partitions of the input set and the output set, respectively, of a keyed function Φ_Z. The pair (F, G) is a partition-pair *for Φ_Z if all subsets of F contain the same number (at least two) of elements, as also do all subsets of G, and if both l and m are at least two.*

In partitioning cryptanalysis, only plaintexts that belong to a fixed subset of a partition of the plaintexts are considered. An attacker then observes how the corresponding ciphertexts are distributed according to the subsets of the output partition. It is possible to distinguish a block cipher from a true random permutation if this distribution is sufficiently non-uniform. The effectiveness of a *partition pair* can be evaluated through different *imbalance* measurements. The original paper by Harpes and Massey [43] proposes two such metrics:

1. *The peak imbalance*:

$$I_p(X) = \frac{m}{m-1}\left(\max_{0 \le i < m} P\left[X = i\right] - 1/m\right)$$

2. *The squared Euclidian imbalance*:

$$I_2^2(X) = \frac{m}{m-1}\sum_{i=0}^{m-1}\left(P\left[X = i\right] - 1/m\right)^2$$

where m is the number of subset output partitions and X is the random variable mapping plaintexts to the index i of the corresponding output partition subset. In other words, the peak imbalance measures the maximal deviation w.r.t a uniform distribution while the squared Euclidian imbalance measures its distance.

As usual, the distinguisher attack can be extended to a key recovery attack on one more round by performing a partial last round decryption. In this case, the attacker guesses some bits of the last round subkey, evaluates the distribution of the ciphertexts before the last round and computes the imbalance according to the key hypothese. The key leading to the highest imbalance (i.e. the most non-uniform distribution of the ciphertexts according to the output partition) is considered as the correct key. of course, the guessed bits must be chosen so as to allow the evaluation of the index of the output partition subset for each plaintext-ciphertext pair.

The attack was applied against 6 rounds of the *DES* with the two metric imbalances and 256 chosen plaintext-ciphertext pairs. The authors observed that the success rate of the attack was better for the squared Euclidian imbalance approach than for the peak imbalance approach. The results were similar to the ones of differential cryptanalysis.

The stochastic cryptanalysis of Crypton [68] is another example of attack partially inspired by the partitioning cryptanalysis. In the case of Crypton, the partitions of the plaintexts and plaintext differences are based on some invariance properties in the linear part of the round function which involve four bytes out of 16. The resulting attack targets up to eight rounds of Crypton faster than exhaustive search.

2.10 Cube attacks

The cube attack is a general technique developed by Vielhaber [85] and Dinur and Shamir [35] that can be applied against block ciphers, stream ciphers and even keyed hash functions. It makes use of *tweakable polynomials* which are polynomials describing the outputs of the encryption process. These polynomials contain both secret variables (the key bits) and public variables (the input bits). The attacker sets some public variables to all their values and obtains several (related) polynomials. By summing up the resulting polynomials, the attacker can obtain equations with much lower degree and even linear equations in the private variables.

As the polynomial representation can be very complex, it can be considered as a black box and evaluated by multiple query only. In this case, efficient linear test can be used in order to check the linearity of the sums of polynomials during a preproccessing phase where the secret variables can also be set.

According to the authors, cube attacks are provably successful when applied to random polynomials of degree d over n secret variables whenever the number m of public variables exceeds $d+log_d n$. Their time complexity is $2^{d-1}n+n^2$ bit operations. Even though the attack is applicable to block ciphers, the degree of their equivalent polynomials is too high for most practical block ciphers, including the *AES*. The attack was applied on a version of Trivium reduced to 735 initialization rounds (out of 1152) with complexity 2^{30}, and it is conjectured that the technique may extend to breaking 1100 initialization rounds faster than exhaustive search [35].

2.11 Algebraic cryptanalysis

The principle of algebraic cryptanalysis is to write the encryption proccess as a system of multivariate algebraic equations where the key bits are the unknown and then (try to) solve this system. In [82] Shannon stated that breaking a

cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type. This expectation is motivated by the fact that solving a large system of quadratic multivariate polynomial equations is an NP-hard problem over any field. When the number of equations m is equal to the number n of unknowns, the best known algorithm is exhaustive search for small fields and Gröbner base algorithms (with exponential complexity) in the case of larger fields [36, 37].

However, if the number of equations m in the system is higher than the number of variables n, then faster algorithms become possible. In the *linearization* technique, every product of variables $x_i \cdot x_j$ is replaced by a new variable y_{ij}. If the original system is sufficiently overdefined, it is easy to solve the linearized system in the new variables by Gaussian elimination. In [50], Kipnis and Shamir introduced a new algorithm called *relinearization*. The basic principle of relinearization is to exploit additional nonlinear equations expressing relations between the new linearized variables y_i. These methods were extended in the *XL-eXtended Linearization* technique [28] which expands the original system by multiplying the equations with small monomials. When the number of equations m is proportional to the square of the number of variables n in the original system, the XL algorithm is expected to run in polynomial time.

The XL attack was further refined in the *XSL* technique by Courtois and Pieprzyk which takes advantage of sparsity and specific structures in the equations describing the S-boxes of some block ciphers. However, the time complexity and the success rate of this attack is not well understood. In particular, speculations over the implications of the XSL attack on the security of AES and *Serpent* have been the subject of an uncommonly controversial debate in the cryptographic community [69, 61, 26].

Another research direction in algebraic cryptanalysis consists in using of *SAT-solvers* to resolve the system after its conversion to an equivalent satisfiability problem [3]. Providing the system of equations to solve with additional physical information is another interesting approach denoted *algebraic side-channel attacks* [77].

One of the major advantages of algebraic attacks over statistical attacks is their extremely reduced data complexity: a few plaintext-ciphertext pairs are generally sufficient. On the other hand, these attacks are often based on heuristic techniques for which little is known concerning the time complexity.

3. CONTRIBUTION OF THE THESIS

So far, we have introduced the basic principles of cryptography and cryptanalysis, as well as the main techniques developed in the cryptanalysis of block ciphers. Building from this state-of-the-art, research in symmetric cryptography generally extends in two main directions. On the one hand, the evaluation of cryptanalytic attacks usually relies on a number of working assumptions. Hence, improving our understanding of these assumptions and the resulting evaluations is an important scope of research. As a typical example, in linear cryptanalysis, the impact of the linear hull effect on the security of actual block cipher constructions, as well as its possible interference with other assumptions used in block cipher design, are important issues to tackle. On the other hand, the development of new cryptanalytic techniques, possibly dedicated to particular cipher designs, is always needed to trigger interactions in the field.

In this thesis, we aimed to contribute on both issues, with a particular emphasis on the experimental evaluation of different assumptions and attacks. This approach was first motivated by the fact that, in view of the large complexities that are usually considered in the cryptanalysis of actual block ciphers, the amount of experimental works to confirm the relevance of attacks and countermeasures in symmetric cryptography is relatively small (in particular if we compare it with the large amount of more theoretical works). As a consequence, we paid a particular attention to the possibility to confirm and/or discuss our analyzes experimentally, most of the times by tweaking the target ciphers (e.g. reducing their number of rounds). Starting from this general point of view, the rest of this manuscript is structured in 6 technical chapters, each of them corresponding to previously published works, re-organized in three parts as we now detail.

- Part I: Tools for Cryptanalysis
  - *Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent* (Chapter 4)
  - *Improving the Time Complexity of Matsui's Linear Cryptanalysis* (Chapter 5)
- Part II: Dedicated cryptanalysis of the block cipher PRESENT

  - *A Statistical Saturation Attack on the Block Cipher PRESENT* (Chapter 6)
  - *Multi-Trail Statistical Saturation Attacks* (Chapter 7)

- Part III: Reviewing the basic assumptions
  - *Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent* (Chapter 8)
  - *Experimenting Linear Cryptanalysis* (Chapter 9)

In a **first part**, we present two tools that were developed to facilitate the application of linear cryptanalysis and possibly other techniques. The first tool, presented in chapter 4, is a heuristic version of Matsui's Branch-and-Bound algorithm for generating good approximations on block ciphers when strong diffusion in the cipher makes it impossible to generate the best approximations exhaustively. The same approach can also be used to generate good differentials. In this chapter, we also investigate the possibility to take advantage of multiple linear approximations generated by our branch-and-bound to improve the linear cryptanalysis of the block cipher Serpent.

The second tool, described in chapter 5, is based on a mathematical trick that allows decreasing the time complexity of Matsui's algorithm 2 for linear cryptanalysis. The improvement is based on the fact that partial decryptions with many keyguesses imply redundant computations. Using an approach based on the *Fast Fourier Transform*, it is possible to decrease the time complexity from $2^k * 2^k$ to $k * 2^k$ where k is the number of subkey bits retrieved during partial decryption. As the time complexity of the attack sometimes prevent the exploitation of certain linear approximations, this tool can also involve a reduction of the data complexity of the attack in certain contexts (by allowing the use of approximations with better biases). For illustration, the method is applied against the block cipher Serpent and the speed-up is given for exemplary attacks.

In the **second part** of this thesis, we take advantage of the previously introduced tools, in order to investigate the cryptanalysis of PRESENT [23], a 64-bit block cipher specially designed for constrained environments and inspired by the AES candidate Serpent [6].

In chapter 6, we introduce a dedicated attack against PRESENT, that can be related to previous works in the lines of partitioning and saturation cryptanalysis. We denote it as a *Statistical Saturation Attack* and provide an analysis of its expected complexity. The attack is based on the existence of a particular trail in the cipher, in which the diffusion achieved is suboptimal. By combining a distinguishing attack exploiting this low diffusion property with a standard divide-and-conquer approach, it allows one to recover key material in the last cipher round(s). The attack is chosen-plaintext in its basic version but can be easily extended to a known-plaintext scenario.

Next, Chapter 7 analyzes an extension of the previous attack where more trails are taken into account and combined in order to decrease the data complexity of the attack. For this purpose, we provide a theoretical evaluation of the trail distributions using probability transition matrices. Since the exhaustive evaluation of all possible distributions turned out to be computationally hard, we additionally provide a heuristic branch-and-bound algorithm that allows us to generate a large number of good trails. Based on independence assumptions, we extrapolate complexity bounds for this extension of the attack.

Finally, one important open question raised by the previous attacks against PRESENT is to determine how our complexity evaluations hold as the number of rounds increases. More precisely, as such evaluations rely on certain independence assumptions, it is expected that a statistical hull effect (i.e. a generalization of the linear hull effect in the context of multi-bit distributions) starts to play an important role in this context. This question generally relates to the tradeoff between data complexity and time complexity in statistical attacks against block ciphers. It also relates to the recent comments of Sean Murphy about the linear hull effect [70]. Motivated by these observations, the **third part** of this thesis aims at assessing the basic assumptions used in linear cryptanalysis and its extensions. Surprisingly, and as previously mentioned, only a limited number of practical experiments have been conducted for this purposes. And probably the most extensive ones were dedicated to the DES cipher (e.g. in [65]), which has a quite different structure than modern ciphers such as the AES candidates.

As a consequence, we first present practical experiments on the cryptanalysis of the block cipher Serpent, in chapter 8. The goal was to use practical data in order to assess the accuracy of the theoretical model introduced by Biryukov et al. for the analysis of linear cryptanalysis using multiple approximations. Our results illustrate that the hypotheses stated at Crypto 2004 [17] about the possible influence of dependencies between approximations hold (at least) as long as the number of approximations exploited is computationally tractable. They also underline the limits and specificities of Matsui's *algorithm 1* and *algorithm 2* for the exploitation of such approximations. In particular, they show that the application of *algorithm 2* requires good theoretical estimations of the approximation biases, while *algorithm 1* is effective as long as the bias is significantly different from zero.

Then, in chapter 9, we apply the tools developed in the first part of the thesis to different versions of a small block cipher, in which the different statistical parameters implied in linear cryptanalysis can be exhaustively computed. Based on this concrete instance of block cipher, we first propose a number of illustrative experiments in order to evaluate the distance between the so-called practical and provable security approaches for designing block ciphers. Then, we challenge the assumptions of key independence and key equivalence that are frequently used in linear cryptanalysis. Third, we put forward the difficulty of

obtaining precise estimations of the distributions within a secure block cipher when the number of rounds increases. We also provide systematic experiments of linear cryptanalysis using single and multiple approximations in order to confirm a number of intuitive views that can be found in former papers. This final chapter directly raises open problems for the design and analysis of block ciphers that we detail, with other open research problems, in a short conclusion chapter.

BIBLIOGRAPHY

[1] National Security Agency. Skipjack and kea algorithm specifications, version 2.0, May 1998. Available at the National Institute of Standards and Technology's web page.

[2] Thomas Baignères. *Quantitative security of block ciphers : designs and cryptanalysis tools*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2008.

[3] Gregory V. Bard, Nicolas T. Courtois, and Chris Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over gf(2) via sat-solvers. Cryptology ePrint Archive, Report 2007/024, 2007. `http://eprint.iacr.org/`.

[4] Eli Biham. New types of cryptanalytic attacks using related keys. *J. Cryptology*, 7(4):229–246, 1994.

[5] Eli Biham, editor. *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*. Springer, 1997.

[6] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In Serge Vaudenay, editor, *FSE*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.

[7] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In *EUROCRYPT*, pages 12–23, 1999.

[8] Eli Biham, Alex Biryukov, and Adi Shamir. Miss in the middle attacks on idea and khufu. In Knudsen [53], pages 124–138.

[9] Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack - rectangling the serpent. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2001.

[10] Eli Biham, Orr Dunkelman, and Nathan Keller. Enhancing differential-linear cryptanalysis. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 2002.

[11] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 507–525. Springer, 2005.

[12] Eli Biham, Orr Dunkelman, and Nathan Keller. A related-key rectangle attack on the full kasumi. In Roy [79], pages 443–461.

[13] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key impossible differential attacks on 8-round aes-192. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 21–33. Springer, 2006.

[14] Eli Biham, Orr Dunkelman, and Nathan Keller. A simple related-key attack on the full shacal-1. In Masayuki Abe, editor, *CT-RSA*, volume 4377 of *Lecture Notes in Computer Science*, pages 20–30. Springer, 2007.

[15] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[16] Eli Biham and Adi Shamir. Differential cryptoanalysis of feal and n-hash. In *EUROCRYPT*, pages 1–16, 1991.

[17] Alex Biryukov, Christophe De Cannière, and Michaël Quisquater. On multiple linear approximations. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2004.

[18] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[19] Alex Biryukov and Eyal Kushilevitz. From differential cryptoanalysis to ciphertext-only attacks. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 72–88. Springer, 1998.

[20] Alex Biryukov and Adi Shamir. Structural cryptanalysis of sasas. *J. Cryptology*, 23(4):505–518, 2010.

[21] Alex Biryukov and David Wagner. Slide attacks. In Knudsen [53], pages 245–259.

[22] Alex Biryukov and David Wagner. Advanced slide attacks. In Preneel [76], pages 589–606.

[23] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid

Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[24] Johan Borst, Lars R. Knudsen, and Vincent Rijmen. Two attacks on reduced idea. In *EUROCRYPT*, pages 1–13, 1997.

[25] Joo Yeon Cho, Miia Hermelin, and Kaisa Nyberg. A new technique for multidimensional linear cryptanalysis with applications on reduced round serpent. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC*, volume 5461 of *Lecture Notes in Computer Science*, pages 383–398. Springer, 2008.

[26] Carlos Cid and Gaëtan Leurent. An analysis of the xsl algorithm. In Roy [79], pages 333–352.

[27] Don Coppersmith. The data encryption standard (des) and its strength against attacks. *IBM Journal of Research and Development*, 38(3):243–250, 1994.

[28] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Preneel [76], pages 392–407.

[29] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher square. In Biham [5], pages 149–165.

[30] Joan Daemen and Vincent Rijmen. Rijndael for aes. In *AES Candidate Conference*, pages 343–348, 2000.

[31] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.

[32] Joan Daemen and Vincent Rijmen, editors. *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*. Springer, 2002.

[33] Hüseyin Demirci. Square-like attacks on reduced rounds of idea. In Kaisa Nyberg and Howard Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 147–159. Springer Berlin / Heidelberg, 2003.

[34] Yvo Desmedt, editor. *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*. Springer, 1994.

[35] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.

[36] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, June 1999.

[37] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. ACM.

[38] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of rijndael. In Schneier [81], pages 213–230.

[39] Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of rc4. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2001.

[40] Henri Gilbert and Marine Minier. A collision attack on 7 rounds of rijndael. In *AES Candidate Conference*, pages 230–241, 2000.

[41] E.K. Grossman and B. Tuckerman. Analysis of a feistel-like cipher weakened by having no rotating key. *IBM Thomas J. Watson Research Report*, RC 6375, 1977.

[42] Carlo Harpes, Gerhard G. Kramer, and James L. Massey. A generalization of linear cryptanalysis and the applicability of matsui's piling-up lemma. In *EUROCRYPT*, pages 24–38, 1995.

[43] Carlo Harpes and James L. Massey. Partitioning cryptanalysis. In Biham [5], pages 13–27.

[44] Miia Hermelin, Joo Yeon Cho, and Kaisa Nyberg. Multidimensional extension of matsui's algorithm 2. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 209–227. Springer, 2009.

[45] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In Biham [5], pages 28–40.

[46] Burton S. Kaliski Jr. and Matthew J. B. Robshaw. Linear cryptanalysis using multiple approximations. In Desmedt [34], pages 26–39.

[47] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified boomerang attacks against reduced-round mars and serpent. In Schneier [81], pages 75–93.

[48] John Kelsey, Bruce Schneier, and David Wagner. Related-key cryptanalysis of 3-way, biham-des, cast, des-x, newdes, rc2, and tea. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *ICICS*, volume 1334 of *Lecture Notes in Computer Science*, pages 233–246. Springer, 1997.

[49] Joe Kilian and Phillip Rogaway. How to protect des against exhaustive key search. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 252–267. Springer, 1996.

[50] Aviad Kipnis and Adi Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In Wiener [88], pages 19–30.

[51] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *FSE*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 1994.

[52] Lars R. Knudsen. Contemporary block ciphers. In Ivan Damgård, editor, *Lectures on Data Security*, volume 1561 of *Lecture Notes in Computer Science*, pages 105–126. Springer, 1998.

[53] Lars R. Knudsen, editor. *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*. Springer, 1999.

[54] Lars R. Knudsen and Thomas A. Berson. Truncated differentials of safer. In Dieter Gollmann, editor, *FSE*, volume 1039 of *Lecture Notes in Computer Science*, pages 15–26. Springer, 1996.

[55] Lars R. Knudsen and John Erik Mathiassen. On the role of key schedules in attacks on iterated ciphers. In Pierangela Samarati, Peter Y. A. Ryan, Dieter Gollmann, and Refik Molva, editors, *ESORICS*, volume 3193 of *Lecture Notes in Computer Science*, pages 322–334. Springer, 2004.

[56] Lars R. Knudsen and Matthew J. B. Robshaw. Non-linear approximations in linear cryptoanalysis. In *EUROCRYPT*, pages 224–236, 1996.

[57] Lars R. Knudsen, Matthew J. B. Robshaw, and David Wagner. Truncated differentials and skipjack. In Wiener [88], pages 165–180.

[58] Lars R. Knudsen and David Wagner. Integral cryptanalysis. In Daemen and Rijmen [32], pages 112–127.

[59] Chi-Sung Laih, editor. *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, volume 2894 of *Lecture Notes in Computer Science*. Springer, 2003.

[60] Susan K. Langford and Martin E. Hellman. Differential-linear cryptanalysis. In Desmedt [34], pages 17–25.

[61] Chu-Wee Lim and Khoongming Khoo. An analysis of xsl applied to bes. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 242–253. Springer, 2007.

[62] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.

[63] Stefan Lucks. The saturation attack - a bait for twofish. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2001.

[64] Mitsuru Matsui. Linear cryptoanalysis method for des cipher. In *EUROCRYPT*, pages 386–397, 1993.

[65] Mitsuru Matsui. The first experimental cryptanalysis of the data encryption standard. In Desmedt [34], pages 1–11.

[66] Mitsuru Matsui and Atsuhiro Yamagishi. A new method for known plaintext attack of feal cipher. In *EUROCRYPT*, pages 81–91, 1992.

[67] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[68] Marine Minier and Henri Gilbert. Stochastic cryptanalysis of crypton. In Schneier [81], pages 121–133.

[69] S. Murphy and M.J.B. Robshaw. Comments on the security of the aes and the xsl technique. Public reports of the NESSIE project NES/DOC/RHU/WP5/026/1, 26 September 2002.

[70] Sean Murphy. The effectiveness of the linear hull effect. Technical Report RHUL-MA-2009-19, Royal Holloway University of London, 2009.

[71] Jorge Nakahara Jr., Daniel Santana de Freitas, and Raphael Chung-Wei Phan. New multiset attacks on rijndael with large blocks. In Ed Dawson and Serge Vaudenay, editors, *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 277–295. Springer, 2005.

[72] National Institute of Standards and Technology. Data encryption standard. FIPS Publication 46-2, December 1993.

[73] National Institute of Standards and Technology. Announcing request for candidate algorithm nominations for the advanced encryption standard (aes). Federal Register (Volume 62, Number 177), September 12 1997.

[74] Kaisa Nyberg. Linear approximation of block ciphers. In *EUROCRYPT*, pages 439–444, 1994.

[75] Gilles Piret and Jean-Jacques Quisquater. Integral cryptanalysis on reduced-round safer++. Cryptology ePrint Archive, Report 2003/033, 2003. `http://eprint.iacr.org/`.

[76] Bart Preneel, editor. *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*. Springer, 2000.

[77] Mathieu Renauld and François-Xavier Standaert. Algebraic side-channel attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Inscrypt*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2009.

[78] Vincent Rijmen. *Cryptanalysis and Design of Iterated Block Ciphers*. PhD thesis, Katholieke Universiteit Leuven, 1997.

[79] Bimal K. Roy, editor. *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*. Springer, 2005.

[80] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In Ross J. Anderson, editor, *FSE*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204. Springer, 1993.

[81] Bruce Schneier, editor. *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*. Springer, 2001.

[82] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28:656–715, 1949.

[83] Akihiro Shimizu and Shoji Miyaguchi. Fast data encipherment algorithm feal. In *EUROCRYPT*, pages 267–278, 1987.

[84] Serge Vaudenay. Provable security for block ciphers by decorrelation. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS*, volume 1373 of *Lecture Notes in Computer Science*, pages 249–275. Springer, 1998.

[85] Michael Vielhaber. Breaking one.fivium by aida an algebraic iv differential attack. Cryptology ePrint Archive, Report 2007/413, 2007. `http://eprint.iacr.org/`.

[86] Auguste Kerckhoffs von Nieuwenhof. La cryptographie militaire. *Journal des Sciences Militaires*, IX:5–83, January 1883.

[87] David Wagner. The boomerang attack. In Knudsen [53], pages 156–170.

[88] Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

[89] Yongjin Yeom, Sangwoo Park, and Iljun Kim. On the security of camellia against the square attack. In Daemen and Rijmen [32], pages 89–99.

[90] Muhammad Reza Z'aba, Håvard Raddum, Matthew Henricksen, and Ed Dawson. Bit-pattern based integral attack. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 363–381. Springer, 2008.

Part I

TOOLS FOR CRYPTANALYSIS

4. MULTIPLE LINEAR CRYPTANALYSIS OF REDUCED ROUND SERPENT

Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent

B. Collard*, F.-X. Standaert**, J.-J. Quisquater

UCL Crypto Group, Microelectronics Laboratory, Louvain-la-Neuve, Belgium

Abstract. This paper reports on the improved and multiple linear cryptanalysis of reduced round Serpent by mean of a branch-and-bound characteristic search within the algorithm. We first present a 9-round linear characteristic with probability $\frac{1}{2} + 2^{-50}$ that involves a reduction of the estimated data complexity of the best reported attack by a factor of 16. Then, we investigate the possibility to take advantage of multiple linear approximations for improving the linear cryptanalysis of Serpent. According to the framework of Biryukov *et al.* from Crypto 2004, we provide estimations of the improved data complexity of such attacks and derive practical cryptanalysis scenarios. For computational reasons, the branch-and-bound search is not guaranteed to be optimal. However, these are the best reported complexities of a linear attack against Serpent.

Keywords: linear cryptanalysis, multiple linear cryptanalysis, Advanced Encryption Standard, Serpent, linear approximations, branch-and-bound.

1 Introduction

The linear cryptanalysis [8] is one of the most powerful attacks against modern block ciphers in which an adversary exploits a linear approximation of the type:

$$P[\chi_P] \oplus C[\chi_C] = K[\chi_K] \tag{1}$$

In this expression, P, C and K respectively denote the plaintext, ciphertext and the secret key while $A[\chi]$ stands for $A_{a_1} \oplus A_{a_2} \oplus ... \oplus A_{a_n}$,with $A_{a_1}, ..., A_{a_n}$ representing particular bits of A in positions $a_1, ..., a_n$ (χ is usually denoted as a mask). In practice, linear approximations of block ciphers can be obtained by the concatenation of one-round approximations and such concatenations (also called characteristics) are mainly interesting if they maximize the deviation (or bias) $\epsilon = p - \frac{1}{2}$ (where p is the probability of a given linear approximation).

In its original paper, Matsui described two methods for exploiting the linear approximations of a block cipher, respectively denoted as *algorithm 1* and *algorithm 2*. In the first one, given an r-round linear approximation with sufficient bias, the algorithm simply counts the number of times the left side of Equation 1

* Work supported by the project Nanotic-Cosmos of the Walloon Region.

** Postdoctoral researcher of the Belgian Fund for Scientific Research (FNRS).

is equal to zero for N pairs (plaintext, ciphertext). If $T > N/2$, then it assumes either $K[\chi_K] = 0$ if $\epsilon > 0$ or $K[\chi_K] = 1$ if $\epsilon < 0$ so that the experimental value $(T - N/2)/N$ matches the theoretical bias. If $T > N/2$, an opposite reasoning holds. For the attack to be successful, it is shown in [8] that the number of available (plaintext, ciphertext)-pairs must be proportional to $\frac{1}{\epsilon^2}$.

In the second method, an r-1-round characteristic is used and a partial decryption of the last round is performed by guessing the key bits involved in the approximation. As a consequence, all the guessed key bits can be recovered rather than the parity $K[\chi_K]$ which yields much more efficient attacks in practice.

Among the various proposals to improve the linear cryptanalysis of block ciphers, Kaliski and Robshaw proposed in 1994 an algorithm using several linear approximations [6]. However, their method imposed a strict constraint as it requires to use only approximations implying the same bits of subkeys $K[\chi_K]$. This restricted at the same time the number and the quality of the approximations available. As a consequence, an approach removing this constraint was proposed in 2004 [4] that can be explained as follows. Let us suppose that one has access to m approximations on r block cipher rounds of the form:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \ (1 \leq i \leq m), \tag{2}$$

and wishes to determine the value of the binary vector:

$$\mathbf{Z} = (z_1, z_2, ..., z_m) = (K[\chi_K^1], K[\chi_K^2], ..., K[\chi_K^m]) \tag{3}$$

The improved algorithm associates a counter T_i with each approximation, that is incremented each time the corresponding linear approximation is verified for a particular pair (plaintext-ciphertext). As for algorithm 1, the values of $K[\chi_K^i]$ are determined from the experimental bias $(T^i - N/2)/N$ and the theoretical bias ϵ_i by means of a maximum likelihood rule. The extension of algorithm 2 to multiple approximations is similarly described in [4].

An important consequence of this work is that the theoretical data complexity of the generalized multiple linear cryptanalysis is decreased compared to the original attack. According to the authors of [4], the attack requires a number of texts inversely proportional to the capacity of the system of equations used by the adversary that is defined as: $\bar{c}^2 = 4 \cdot \sum_{i=1}^{n} \epsilon_i^2$. Therefore, by increasing this quantity by using more approximations, one can decrease the number of texts necessary to perform a successful key recovery.

In this paper, we aim to apply the previously described cryptanalytic tools to the AES candidate Serpent [1]. For this purpose, we first apply a branch-and-bound algorithm to derive an improved single linear characteristic for the cipher. It allows us to reduce the expected complexity of a linear cryptanalysis by a factor of 16. Due to the structure of the Serpent algorithm components (in particular its S-boxes and diffusion layer), the Matsui's branch-and-bound method could not be applied as such and we proposed a modified algorithm,

based on the minimization of the number of active S-boxes in the linear transform. Then, in the second part of the paper, we take advantage of our modified algorithm in order to investigate multiple linear approximations. We show that a large number of linear approximations with significant biases can be derived and evaluate the resulting capacities of the obtained systems. As result of these experiments, the *theoretical* complexity against 10-rounds Serpent can be as low as 2^{80}. We mention that these conclusions have to be tempered by the possibility to perform practical attacks dealing with large number of equations and by the possibility that a significant part of these equations give rise to dependent information, as discussed at the end of this paper. Therefore, practical experiments of multiple linear cryptanalysis against actual ciphers appear to be a necessary step for the better understanding of these theoretical improvements.

2 The Serpent algorithm

The Serpent block cipher was designed by Ross Anderson, Elie Biham and Lars Knudsen [1]. It was an Advanced Encryption Standard candidate, finally rated just behind the AES Rijndael. Serpent has a classical SPN structure with 32 rounds and a block width of 128 bits. It accepts keys of 128, 192 or 256 bits and is composed of the following operations:

- an initial permutation IP,
- 32 rounds, each of them built upon a subkey addition, a passage through 32 S-boxes and a linear transformation L (excepted the last round, where the linear transformation is not applied),
- a final permutation FP.

In each round R_i, only one S-box is used 32 times in parallel. The cipher uses 8 distinct S-boxes S_i $(0 \leq i \leq 7)$ successively along the rounds and consequently, each S-box is used in exactly four different rounds. Finally, the linear diffusion transform is entirely defined by *XOR*s ($\oplus$), *rotations* ($\lll$) and left *shifts* ($\ll$). Its main purpose is to maximize the avalanche effect within the cipher. If one indicates by X_0, X_1, X_2, X_3 the $4 \cdot 32$ bits at the input of the linear transformation, it can be defined by the following operations:

$$
\begin{array}{r}
\text{input} = X_0, X_1, X_2, X_3 \\
\hline
X_0 = X_0 \lll 13 \\
X_2 = X_2 \lll 3 \\
X_1 = X_1 \oplus X_0 \oplus X_2 \\
X_3 = X_3 \oplus X_2 \oplus (X_0 \ll 3) \\
X_1 = X_1 \lll 1 \\
X_3 = X_3 \lll 7 \\
X_0 = X_0 \oplus X_1 \oplus X_3 \\
X_2 = X_2 \oplus X_3 \oplus (X_1 \ll 7) \\
X_0 = X_0 \lll 5 \\
X_2 = X_2 \lll 22 \\
\hline
\text{output} = X_0, X_1, X_2, X_3
\end{array}
$$

3 Matsui's branch-and-bound approximation search

The first step in a linear cryptanalysis consists in finding linear approximations with biases as high as possible. In practice, the adversary usually starts by investigating the non-linear components in the cipher (*e.g.* the S-boxes) and tries to extrapolate partial approximations through the whole. A first problem is then to compute the probability of the concatenated linear approximations, that is usually estimated thanks to the following *piling-up lemma*. Let the bias of a linear approximation on the block cipher ith round be defined as:

$$(\chi_{I_i}, \chi_{O_i}) = \epsilon_i = \Pr\{I_i[\chi_{I_i}] \oplus O_i[\chi_{O_i}] = 0\} - 1/2 \tag{4}$$

The total bias ϵ_{tot} on r rounds is then given by:

$$\epsilon_{tot} = [\epsilon_1, \epsilon_2, ..., \epsilon_r] = 2^{r-1} \prod_{i=1}^{r} \epsilon_i, \tag{5}$$

and the best r rounds linear approximation is defined as:

$$B_r = \max_{\substack{\chi_{I_i} = \chi_{O_{i-1}} \\ (2 \leq i \leq r)}} [(\chi_{I_1}, \chi_{O_1}), (\chi_{I_2}, \chi_{O_2}), ..., (\chi_{I_r}, \chi_{O_r})] \tag{6}$$

Next to the pilling up lemma, the problem of searching the best r-round approximation is not trivial. It consists of finding $r + 1$ binary masks (one for each output round plus one mask for the input of the cipher), which define a linear approximation of maximum bias for a particular encryption algorithm. The difficulty of the problem mainly comes from the great cardinality of the set of candidates. In 1994, Matsui proposed a branch-and-bound algorithm making possible to effectively find the best linear approximation of the DES [9]. The algorithm works by induction: knowing the value of maximum bias on the *r-1* first rounds, it manages to find the maximum bias on r rounds as well as the corresponding input and output masks. For this purpose, it constantly requires to know a lower bound $\overline{B_r}$. This bound must imperatively be lower or equal to the exact B_r and the closer it is from B_r, the faster the algorithm converges.

In practice, the set of all the possible characteristics through the cipher can be seen as a large tree. At the roots stand the input masks of the cipher approximations, at each branch stand the output masks of a round and at the leaves stand the output masks of the cipher. Each branch of the tree is divided in as many new branches as there are possible approximations for the next round. In this tree, a linear approximation on i rounds forms a path starting from a root up to the i-th level. The number of leaves of the tree growing exponentially with the number of rounds, it quickly becomes unthinkable to evaluate all the approximations by exhaustive search. The principle of the branch-and-bound therefore consists in cutting whole branches of the tree as soon as it becomes obvious that all the corresponding linear approximations will have a bias lower than the bound $\overline{B_r}$. Being given an approximation on i rounds and its bias ϵ_i, a

linear approximation on r rounds generated by concatenating an approximation on $r-i$ rounds with this approximation cannot have a bias better than $[B_{r-i}, \epsilon_i]$. Consequently, if $[\epsilon_i, B_{r-i}]$ is strictly lower than $\overline{B_r}$, it is useless to prospect this part of the tree more in detail. Matsui's technique is obtained by the systematic application of this reasoning and is described in more details in [9], Appendix A.

The branch-and-bound strategy can be applied as such to many ciphers. However, its efficiency and time complexity varies, notably depending on the initial estimation $\overline{B_r}$. Small estimations increase the size of the search tree. If the estimation is too large, the algorithm may not return any characteristic at all [5]. For DES, good estimations could be found by first performing a restricted search over all characteristics which only cross a single S-box in each round. Unfortunately, the algorithm may perform poorly for other ciphers and in particular, could hardly be applied as such to Serpent, as the next section underlines.

4 Linear approximations search for Serpent

4.1 Observations on the S-boxes and single round approximations

The Serpent S-boxes have only four (non-trivial) different biases: $\pm 2^{-2}$ and $\pm 2^{-3}$. Consequently, by the piling-up lemma, the bias of any approximation is a power of 2. The S-boxes S_0, S_1, S_2 and S_6 have 36 biases $\epsilon = \pm 0.25$ and 96 biases $\epsilon = \pm 0.125$. The S-boxes S_3, S_4, S_5 and S_7 have 32 biases $\epsilon = \pm 0.25$ and 112 biases $\epsilon = \pm 0.125$. In Table 1, we summarized the distributions of the approximations biases for one round of Serpent and compared them with similar results obtained in [10] for the DES and FEAL.

ϵ	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
DES	1	13	195	3803	40035	371507	...
FEAL	16	1808	98576	$3.45 \cdot 10^6$	$7.74 \cdot 10^8$	$1.22 \cdot 10^9$	...
Serpent: $S_{0,1,2,6}$	1	1153	647041	$2.35 \cdot 10^8$	$6.25 \cdot 10^{10}$	$1.29 \cdot 10^{13}$	$2.15 \cdot 10^{15}$
Serpent: $S_{3,4,5,7}$	1	1025	512513	$1.67 \cdot 10^8$	$3.96 \cdot 10^{10}$	$7.33 \cdot 10^{12}$	$1.10 \cdot 10^{15}$

Table 1: Number of 1-round linear approximations with bias $\geq \epsilon$ for some ciphers

This table clearly illustrates that the number of approximations on a round of Serpent is several orders superior to those of DES and FEAL. We consequently expected practical problems in the linear approximations search due to an explosion of the number of candidates to be explored within the branch-and-bound. This was experimentally confirmed: a classical implementation of Matsui's algorithm appeared unable to determine the best approximation on as low as three rounds. One reason for this phenomenon is the good diffusion provided by the linear transform, causing a significant increase in the number of active S-boxes at each round, with an exponential increase in the number of approximation-candidates being tested. Even by arbitrarily limiting the number of candidates to be explored, the algorithm was stuck after 5 to 6 rounds.

4.2 Modified search algorithm

Matsui's branch-and-bound method is primarily concerned with the analysis of the S-boxes in order to obtain linear approximations with high biases. However, the previously described failure suggests that in Serpent, the search for optimal linear characteristics is severely slowed down by the avalanche effect. Therefore, limiting the number of active S-boxes in the linear approximations could help to improve the search efficiency. This observation led us to modify the branch-and-bound algorithm. In our modified method, we start by exhaustively counting all the couples (input, output) of the linear transform for which the number of active S-boxes is arbitrarily low. Such couples are called *transform candidates* and are entirely defined by the number of active input/output S-boxes, the position of these S-boxes and their corresponding mask value. The transform candidates are then stored in a database (*e.g.* a hash table) so that all the candidates having the same active output S-boxes are stored in the same list.

Once the database is created, we launch the actual approximation search: a linear approximation on r rounds can be obtained by the concatenation of r transform candidates, if the positions of the active S-boxes at the output of a transform candidate correspond to those of the active S-boxes at the input of the next transform candidate (*i.e.* we have to fulfill boundary conditions, as in the original method). These constraints are easily checked, as the candidates are picked up in the database according to the position of their active output S-boxes. To calculate the bias of an approximation, we simply apply the piling up lemma to the S-boxes approximations generated between the transform candidates. We can then determine the best approximations by means of a traditional branch-and-bound, where the only difference is that we pile up the transform candidates instead of piling up round approximations. Note that we pile up the candidates starting with the last round, then going down gradually until the first round, in order to benefit from the knowledge of best bias in the branch-and-bound.

In order to improve the flexibility of the approximation search, the input mask in the first round and the output mask in the last round are chosen in order to maximize the biases of these rounds. Therefore, in a r rounds approximation, transform candidates are only picked up for the $r - 1$ inner rounds, which speeds up the research. Additionally, the first round input mask and last round output mask can be replaced by any other mask, provided that the biases are left unchanged. Due to the properties of the Serpent S-boxes, for any approximation found by the algorithm, many more similar approximations can be generated by the modification of the outer masks.

4.3 Performance and hardware constraints

The number of possible transform candidates that one can pile up in each round is limited by the boundary conditions and the size of the lists. Moreover, among these candidates, a majority leads to a zero bias and is thus directly rejected. Indeed, a significant proportion of the Serpent S-boxes linear approximations is null: the proportion varies between $93/225 = 0.413$[1] for the S-boxes S_0, S_1, S_2, S_6 and $81/225 = 0.36$ for the others. If there are q active S-boxes at the output of the transform candidates, then only a fraction of roughly $(1 - 0.413)^q$ or $(1 - 0.36)^q$ of these candidates will lead to a non-zero bias. As this proportion falls when q increases, there is no exponential increase of the number of candidates anymore.

Nevertheless, the major drawback of the proposed method remains in the consequent size of the database used. Even if the proportion of useful transform candidates is weak, the number of candidates stored can easily reach several millions. In an optimized structure, a transform candidate with q active S-boxes requires $(10 + 9 * q)$ bits of memory[2]. If one considers a reasonable average of 16 active S-boxes per candidate, it yields 154 bits, *i.e.* 54400 candidates per megabyte, or approximately $55 \cdot 10^6$ candidates per gigabyte. Additionally, it is of course possible that the algorithm does not find any linear approximation beyond a certain number of rounds, either because the list of candidates checking the boundary conditions is empty, or because all these candidates lead to a zero bias. The higher the number of transform candidates, the lower the risk.

On the positive side, this database can be precomputed before the execution of the branch-and-bound. Once it is created, the execution of the algorithm is very fast since in each stage of the branch-and-bound, we only consult tables and calculate biases. In our experimentations, we generated the database as follows: we exhaustively generated all the possible transform candidates with maximum i $(1 \leq i \leq 5)$ active input S-boxes and stored only those with maximum $(15 - i)$ active output S-boxes. This required the analysis of approximately $3 \cdot 10^{11}$ linear transformations[3] among which only about $130 \cdot 10^6$ were stored/kept. With about 10^6 analyzes per second, this search took roughly 4 days on a 2GHz processor[4]. The exhaustive search of all the transform candidates with $i = 6$ would require the analysis of approximately $21 \cdot 10^{12}$ transformations, *i.e.* roughly 250 days of computation on the same processor. Better strategies could probably be investigated, taking advantage of the linearity of the diffusion layer, and are a scope for further research. Note that the database would not be the same if differential approximations were to be found [2, 9] although it would be strongly correlated.

[1] As $\exists$ 132 approximations with non-zero biases among the 15*15 non-trivial ones.
[2] Namely 2*5 bits to store the number of input/output active S-boxes (1...32) and 9 bits to store the 5-bit position (1...32) and 4-bit mask (0...15) of each active S-box.
[3] That is $2 \cdot \sum_{k=1}^{5} 15^k * \mathrm{C}_{32}^k$.
[4] Note that the task can be parallelized.

5 Practical results

5.1 Best approximation found

Since the best 9-round characteristic found by Biham *et al.* involves at most 15 active S-boxes at the extremity of the linear transform [3], our previously described database ensures us to find an approximation with a bias at least as high. Table 2 summarizes the results of our approximations search, in function of the starting S-box (and therefore round). The best biases are given in the penultimate column and the values of Biham *et al.* are in the last column. As a first result of our investigations, we found an improved approximation starting with S_3. Our improved characteristic is very similar to the one in [3], excepted for the three first rounds. It involves a reduction of the linear cryptanalysis data complexity by a factor of 16. Due to a lack of room, the description of the linear approximations used in our attacks are not given in this paper. It is available at: http://www.dice.ucl.ac.be/crypto/publications/2007/inscrypt_appendix.pdf.

r	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	Max	$\overline{B_r}$
3	–	2^{-8}	2^{-8}	2^{-8}	2^{-8}	2^{-7}	2^{-8}	2^{-7}	2^{-7}	2^{-7}
4	–	2^{-14}	2^{-13}	2^{-16}	2^{-12}	2^{-12}	2^{-12}	2^{-12}	2^{-12}	2^{-12}
5	–	2^{-23}	2^{-21}	2^{-23}	2^{-17}	2^{-18}	2^{-18}	2^{-18}	2^{-17}	2^{-18}
6	–	–	–	2^{-30}	2^{-23}	2^{-25}	2^{-24}	2^{-33}	2^{-23}	2^{-25}
7	–	–	–	2^{-36}	2^{-30}	2^{-32}	–	–	2^{-30}	2^{-32}
8	–	–	–	2^{-43}	2^{-37}	–	–	–	2^{-37}	2^{-39}
9	–	–	–	2^{-50}	–	–	–	–	2^{-50}	2^{-52}

Table 2: Biases of the best 9-round approximations for different starting S-boxes.

Note that we also tried to find iterative approximations, *i.e.* approximations of which the input masks fulfill the boundary conditions of their own output. Such approximations are useful in practice because they can be straightforwardly extended to an arbitrary number of rounds [7]. However, our restricted database did not allow us to find any such approximation on 8 rounds. Searching over larger databases would therefore be necessary to further investigate the existence of good iterative characteristics.

5.2 Multiple linear approximations found

In addition to the previously reported characteristic, we ran our search algorithm in order to generate a list of useful approximations for a multiple linear cryptanalysis of Serpent. As for the previous section, this generation is very fast due to the properties of the branch-and-bound that allows an effective limitation of the candidates to explore. Table 3 reports the bias distribution of the 150 best 9-rounds linear approximations of Serpent found with our database (starting with S_3). Additionally, since each of these approximations can generate several others by simply replacing its input and output masks (as discussed in Section 4.2), we straightforwardly obtained the distribution in table 4.

bias	# of approx.	bias	# of approx.
2^{-50}	3	2^{-53}	42
2^{-51}	12	2^{-54}	66
2^{-52}	27	2^{-55}	

Table 3: Bias distribution of the 150 best approximations found.

bias	# of approx.	capacity	bias	# of approx.	capacity
2^{-50}	786432	$2.482 \cdot 10^{-24}$	2^{-55}	$1.208 \cdot 10^{13}$	$5.184 \cdot 10^{-20}$
2^{-51}	$6.134 \cdot 10^{7}$	$5.087 \cdot 10^{-23}$	2^{-56}	$1.250 \cdot 10^{14}$	$1.481 \cdot 10^{-19}$
2^{-52}	$2.290 \cdot 10^{9}$	$5.024 \cdot 10^{-22}$	2^{-57}	$1.059 \cdot 10^{15}$	$3.520 \cdot 10^{-19}$
2^{-53}	$5.447 \cdot 10^{10}$	$3.188 \cdot 10^{-21}$	2^{-58}	$7.513 \cdot 10^{15}$	$7.138 \cdot 10^{-19}$
2^{-54}	$9.281 \cdot 10^{11}$	$1.463 \cdot 10^{-20}$	2^{-59}	$4.553 \cdot 10^{16}$	$1.262 \cdot 10^{-18}$

Table 4: Bias distribution and cumulative capacities of the extended approximations.

5.3 Resulting capacity and discussion of the results

According to the framework in [4], the use of multiple approximations in linear cryptanalysis allows decreasing the number of plaintexts needed for a successful key recovery proportionally to the capacity of the obtained system (given in Table 4). It involves the following observations:

- The data complexity of the simple linear cryptanalysis of 10-rounds Serpent is approximately 2^{100} (using the best 9-round approximation with bias 2^{-50}).
- If the 786432 approximations with bias 2^{-50} are used, the resulting capacity equals $2.482 \cdot 10^{-24}$, which yields a theoretical data complexity of $2^{78.4}$.
- Cumulatively using all the $2.29 \cdot 10^{9}$ approximations of bias higher than 2^{-52}, we could theoretically reach a capacity of $5.024 \cdot 10^{-22}$, which would correspond to $2^{70.8}$ plaintext-ciphertext pairs.
- The more realistic use of 2048 (*resp.* $1.802 \cdot 10^{6}$) approximations with bias 2^{50} (*resp.* greater than 2^{52}) involving the same target subkey would result in a theoretical data complexity of 2^{87} (*resp.* 2^{81}).

As a matter of fact, these results do not take the size of the target subkey (and therefore the time complexity) into account but only consider the data complexity. In the next section, we propose more realistic attacks presenting a better trade-off between data and time complexities. Let us also mention that a possibly more powerful way to exploit multiple approximations would be to consider Matsui's *algorithm 1* and therefore avoid the time complexity problems related to key guesses, *e.g.* using the three 2^{-50} bias approximations and their derivatives. Since each approximation reveals up to one bit of information on the secret key and $m \gg 128$, the resulting linear system is strongly overdefined.

In general, the previous results have to be tempered by the possible influences of dependencies between the various linear approximations exploited in an attack. This is specially true in our context since our approximations are all generated from an initial set of 150 characteristics and the number of derivatives is much larger than $2 \cdot 128$. As discussed in [4], it is actually hard to determine

the consequence of these dependencies on the capacity of the multiples approximations. Because $2 \cdot m \cdot \epsilon \ll 1$ (for any reasonable choice for the number m of approximations), the dependencies between the text masks (χ_P^i, χ_C^i) should have a negligible influence on the capacity. Therefore, the major question relates to the dependencies between the linear trails. As a matter of fact, the estimated data complexities in our analysis (as well as in Biryukov *et al.*'s) are fairly optimistic and an important next step in the understanding of linear cryptanalysis would be to experiment these predictions with a real life cipher.

6 Realistic attack scenarios against Serpent

In this section, we present realistic attack scenarios on reduced round Serpent using (multiple) linear cryptanalysis. The reduced version is just like Serpent, excepted for its reduced number of rounds. After the last S-box, the linear transformation is omitted as it has no cryptographic significance. The linear approximations used were generated with the algorithm presented before.

Using an approximation on $r-1$ rounds, one can recover bits of the subkey in round r. In Matsui's original method, a partial decryption of the last round is performed for every ciphertext by guessing the key bits involved in the approximation. The parity of the approximation for the plaintext and the partially decrypted ciphertext is then evaluated and a counter corresponding to each key guess is incremented if the relation holds or decremented otherwise. The key candidate with the highest counter in absolute value is then assumed to be the correct key. However, as we only consider a limited number of bits k (in the active S-boxes) during the partial decryption of the ciphertexts, the same pattern for these k bits possibly appear several times during the attack. In order to avoid doing the same partial decryption work several times, Biham *et al.* proposed in [3] an improvement which considerably reduces the time complexity of an attack:

- Initialize an array of 2^k counters (k is the size of the target subkey).
- *For each generated ciphertext:* extract the k-bit value corresponding to the active S-boxes and evaluate the parity of the plaintext subset defined by the approximation. Increment or decrement a counter corresponding to the extracted k-bit value according to the obtained parity.
- *Once all the ciphertexts have been generated:* for each k-bit ciphertext and for each k-bit subkey, partially decrypt the k-bit ciphertext under the k-bit subkey and evaluate the parity of the output subset (as defined by the linear approximation). Keep this value in a table of size $2^k \cdot 2^k$.
- For each k-bit subkey, evaluate its experimental bias by checking, for each k-bit ciphertext, the parity of the approximation and the value of the corresponding counter. Then output the subkey with maximal bias.

This method reduces the attack time complexity from $O(N \cdot 2^k)$ to $O(2^k \cdot 2^k)$.

6.1 Attack on 7 rounds Serpent

The attack uses a 6-round approximation starting with S-box 4 and ending with S-box 1. This approximation is derived from the best 6-round approximation found in section 5 except a small change in the last round that reduces the number of active S-boxes from 13 to 5. The bias of the approximation falls from 2^{-23} to 2^{-25}. Using only one approximation, the data complexity is approximately 2^{52} known plaintexts. The attack requires 2^{20} counters and a time complexity of approximately 2^{40} decryptions of 1-round Serpent. Several similar approximations can be obtained by changing the input mask of the relation. We found up to 8 approximations with bias 2^{-25} and up to 96 approximations with bias 2^{-26}. This would lead to a capacity of respectively 2^{-45}, 2^{-43}, reducing the data complexity to 2^{47} or 2^{45} at the cost of a slight increase of the time/memory complexities.

6.2 Attack on 8 rounds Serpent

Similarly, we can attack 8-round Serpent using a 7-round approximation starting with S-box 4 and ending with S-box 2. The approximation is the one found in section 5. It has a bias of 2^{-30} and 7 active S-boxes. Consequently, an attack using only one approximation requires 2^{56} 1-round decryptions, 2^{28} counters and $\simeq 2^{62}$ known plaintexts. We can again reduce the data complexity by taking advantage of multiple approximations. We found 8 approximations with the same bias and the same active S-boxes giving a capacity of 2^{-55}, thus a data complexity of approximately 2^{57} plaintexts. Adding 96 approximations with bias 2^{-31}, we obtain a capacity of 2^{-53} and therefore a data complexity of 2^{55} plaintexts (see Table 6). Again, this effect can be increased at the cost of more memory and computation. For example, there are 384 approximations with bias 2^{-32} and 512 approximations with bias 2^{-33}, that gives rise to data complexities of respectively $2^{54.1}$ or 2^{54}, but requires 2^{28} counters and 2^{56} memory access for each counter.

6.3 Attack on 9 rounds Serpent

The best approximation found on 8 rounds has a bias of 2^{-37} but it has 23 actives input S-boxes. We can slightly modify its input in order to lower this number to 11 active S-boxes. This way, the bias of the approximation is 2^{-39} instead of 2^{-37}. This approximation starts with S-box 4 and ends with S-box 3. The complexity of an attack based on this approximation is 2^{88} 1-round decryptions, 2^{44} counters and about 2^{80} known plaintexts. Multiples approximations allow us to decrease the number of texts needed. We found 128 approximations with bias 2^{-39}, 3584 approximations with bias 2^{-40}, 43008 approximations with bias 2^{-41} and 286720 approximations with bias 2^{-42}. The corresponding capacities are 2^{-69}, 2^{-66}, $2^{-64.1}$, 2^{-63}, thus requiring 2^{71}, 2^{68}, $2^{66.1}$, 2^{65} generated plaintexts.

6.4 Attack on 10 rounds Serpent

We finally ran our search algorithm to generate a list of 150 9-round approximations with high bias and a reasonable number of active S-boxes. Among the the huge number of candidates (see table 4), we found the three following approximations starting and finishing with S-box 3:

- Approximation 1 with bias 2^{-55} and 11 active S-boxes,
- Approximation 2 with bias 2^{-58} and 10 active S-boxes,
- Approximation 3 with bias 2^{-59} and 8 active S-boxes.

Using the first approximation, we obtain an attack requiring 2^{112} texts, 2^{44} counters and 2^{88} decryptions. Using the second approximation, we obtain an attack requiring 2^{118} texts, 2^{40} counters and 2^{80} decryptions. Using the third approximation, we obtain an attack requiring 2^{120} texts, 2^{32} counters and 2^{64} decryptions. Multiple linear cryptanalysis based on the first approximation leads to capacities equal to 2^{-97}, $2^{-93.42}$ or $2^{-90.93}$ according to bias of the approximations. Using the second approximation, the capacities decrease to 2^{-103}, $2^{-99.42}$ or $2^{-96.93}$. With the third approximation the capacities then become 2^{-105}, $2^{-101.42}$ or $2^{-98.93}$. All the presented attack results are summarized in Table 6 and Table 5 remembers the previously known attacks against Serpent.

6.5 Attack on 11 rounds Serpent

We can attack 11 round Serpent with a 9-rounds linear approximation. Such an attack requires a partial encryption before the first round of the approximation and a partial decryption after the last round. In this context, it becomes essential to minimize the total number of active S-boxes, both in input and in output.

Our algorithm provided a 9-rounds approximation with a bias of 2^{-58} and only 27 active S-boxes (15 in input and 12 in output). Using the trick proposed in [3], section6, we obtain a time complexity of $2^{120} \cdot 15/352 + 2^{48} \cdot (2^{118} \cdot 12/352 + 2^{60}) = 2^{166}$ 11-round Serpent encryptions and a memory complexity of 2^{121}. Incidentally, if we perform a partial encryption of the first round in the external loop (instead of a partial decryption of the last round), the time complexity becomes $2^{96} \cdot 12/352 + 2^{60} \cdot (2^{118} \cdot 15/352 + 2^{48}) = 2^{173.5}$ 11-round Serpent encryptions and a memory complexity of 2^{97}. The data complexity is left unchanged, that is 2^{118} known plaintext.

In this case, it is practically not possible to use multiples approximations, as they should have at least the same active S-boxes, both in their input and output.

6.6 Summary

Rounds	Type of attack	complexity		
		data	time	memory
6	differential [13]	2^{83}CP	2^{90}	2^{44}
	differential [13]	2^{71}CP	2^{103}	2^{79}
	differential [13]	2^{41}CP	2^{163}	2^{49}
7	differential [11]	2^{84}CP	$2^{78.9}$	2^{56}
8	Amp.Boomerang [13]	2^{128}CP	2^{163}	2^{137}
	Amp.Boomerang [13]	2^{110}CP	2^{175}	2^{119}
	differential [11]	2^{84}CP	$2^{206.7}$	2^{89}
9	Amp.Boomerang [13]	2^{110}CP	2^{252}	2^{212}
10	Rectangle [14]	$2^{126.3}$CP	2^{165}	$2^{131.8}$
	Boomerang [14]	$2^{126.3}$ACPC	2^{165}	2^{89}
	Lin.Cryptanalysis [3]	2^{116}KP	2^{92}	2^{45}
	Diff.Lin.Cryptanalysis [15]	$2^{105.2}$CP	$2^{123.2}$	2^{40}
11	Lin.Cryptanalysis [3]	2^{118}CP	2^{187}	2^{193}
	Diff.Lin.Cryptanalysis [15]	$2^{125.3}$CP	$2^{172.4}$	2^{30}
	Diff.Lin.Cryptanalysis [15]	$2^{125.3}$CP	$2^{139.2}$	2^{60}
Complexity is measured in encryption units. Memory is mesured in Bytes. CP - Chosen Plaintexts, KP - Known Plaintexts, ACPC - Adaptive Chosen Plaintexts and Ciphertext.				

Table 5: Summary of previous attacks on Reduced-rounds Serpent (see [15]).

Rounds	Type of attack	complexity		
		data	time	memory
7	Lin.cryptanalysis	2^{52}KP	2^{40}	2^{20}
	Mult.Lin.Cryptanalysis (8 appr.)	2^{47}KP	2^{43}	2^{23}
8	Lin.cryptanalysis	2^{62}KP	2^{56}	2^{28}
	Mult.Lin.Cryptanalysis (8 appr.)	2^{57}KP	2^{59}	2^{31}
	Mult.Lin.Cryptanalysis (104 appr.)	2^{55}KP	$2^{62.7}$	$2^{34.7}$
9	Lin.cryptanalysis	2^{80}KP	2^{88}	2^{44}
	Mult.Lin.Cryptanalysis (128 appr.)	2^{71}KP	2^{95}	2^{51}
	Mult.Lin.Cryptanalysis (3712 appr.)	2^{68}KP	$2^{99.9}$	$2^{55.9}$
10	Lin.cryptanalysis ($\epsilon = 2^{-55}$)	2^{112}KP	2^{88}	2^{44}
	Mult.Lin.Cryptanalysis (2048 appr.)	2^{99}KP	2^{99}	2^{55}
	Lin.cryptanalysis ($\epsilon = 2^{-59}$)	2^{120}KP	2^{64}	2^{32}
	Mult.Lin.Cryptanalysis (2048 appr.)	2^{107}KP	2^{75}	2^{43}
11	Lin.cryptanalysis ($\epsilon = 2^{-58}$)	2^{118}KP	2^{166}	2^{121}
	Lin.cryptanalysis ($\epsilon = 2^{-58}$)	2^{118}KP	$2^{173.5}$	2^{97}

Table 6: Summary of attacks on Serpent presented in this paper.

7 Conclusion and further works

This paper first presents a modification of Matsui's branch-and-bound algorithm for the linear approximation search in a block cipher. It enabled us to find the best reported 9-round approximation for the AES candidate Serpent. The algorithm allows speeding up the search of linear approximations at the cost of larger memory requirements. It is generic and especially well suited for ciphers where the linear transformation involves an important avalanche effect (as the AES candidates and most recent ciphers). Moreover, it could be straightforwardly adapted for the research of differential characteristics.

In a second part of the paper, we take advantage of this modified branch-and-bound algorithm in order to investigate the possible use of multiple linear approximations against Serpent. According to the framework of Biryukov *et al.* [4], we provided estimations of the improved data complexity of such attacks against 10-round Serpent that can be down to approximately 2^{80}. Since these results are mainly theoretical (due to an unrealistic time complexity), we also presented several attacks against 7- to 10-round Serpent using reasonable attack parameters that outperform previously known results.

As an important scope for further research, these results should be experimented against real-life ciphers of tractable size in order to determine the actual influence of dependencies between the different approximations used in an attack. That is, to figure out the extend to which the information provided by multiple linear approximations can lead to efficient attack strategies.

Acknowledgements

The authors thank Orr Dunkelman for his helpful answers and advices. We also thank anonymous reviewers from FSE and Inscrypt for their useful comments.

References

1. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, in the proceedings of the First Advanced Encryption Standard (AES) Conference, Ventura, CA, 1998.
2. E. Biham, *On Matsui's Linear Cryptanalysis*, in the proceedings of Eurocrypt 1994, Lecture Notes in Computer Science, vol. 950, pp. 341-355, Perugia, Italy, May 1994.
3. E. Biham, O. Dunkelman, N. Keller, *Linear Cryptanalysis of Reduced Round Serpent*, in the Proceedings of FSE 2001, Lecture Notes in Computer Science, vol. 2355, pp. 16-27, Yokohama, Japan, April 2001.
4. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of CRYPTO 2004, Lecture Notes in Computer Science, vol. 3152, pp.1-22, Santa Barbara, California, USA, August 2004.
5. A. Biryukov, *Linear Cryptanalysis*, in the Encyclopedia of Cryptography and Security, Kluwer Academic Publishers, 2005.
6. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Sciences, vol. 839, pp. 26-39, Santa Barbara, California, USA, August 1994.
7. L.R. Knudsen, *Iterative characteristics of DES and s^2-DES*, in the proceedings of CRYPTO 1992, Lecture Notes in Computer Science, vol. 746, pp. 497-511, Santa Barbara, California, USA, AUgust 1992.
8. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of Eurocrypt 1993, Lecture Notes in Computer Science, vol. 765, pp. 386–397, Lofthus, Norway, May 1993.
9. M. Matsui, *On Correlation Between the Order of S-boxes and the Strength of DES*, in the proceedings of Eurocrypt 1994, Lecture Notes in Computer Science, vol. 950, pp. 366-375, Perugia, Italy, May 1994.
10. K. Ohta, S. Moriai, K. Aoki, *Improving the Search Algorithm for the Best Linear Expression*, in the proceedings of CRYPTO 1995, Lecture Notes in Computer Science, vol. 963, pp. 157-170, Santa Barbara, California, USA, August 1995.
11. E. Biham, O.Dunkelman, N.Keller, *The Rectangle Attack - Rectangling the Serpent*, Advances in Cryptology - Eurocrypt'01 (Lecture Notes in Computer Science no.2045), pp. 340-357, Springer-Verlag, 2001.
12. T. Kohno, J.Kelsey, B.Schneier, *Preliminary Cryptanalysis of Reduced-Round Serpent*, AES Candidate Conference, pp. 195-211, 2000
13. J.Kelsey, T. Kohno, B.Schneier, *Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent*, proceedings of Fast Software Encryption 7, Lecture Notes in Computer Science, vol. 1978, pp. 75-93, Springer-Verlag, 1999.
14. E. Biham, O. Dunkelman, N. Keller, *New Results on Boomerang and Rectangle Attacks*, in the Proceedings of Fast Software Encryption 9, Lecture Notes in Computer Science, vol. 2501, pp. 254-266, Springer-Verlag, 2002.
15. E. Biham, O. Dunkelman, N. Keller, *Differential-linear Cryptanalysis of Serpent*, in the Proceedings of Fast Software Encryption 2003, Lecture Notes in Computer Science, vol. 2887, pp. 9-21, Springer, 2004.

A Matsui's branch-and-bound for SPN

The following pseudo-code implements Matsui's branch-and-bound algorithm. It can be interpreted as a Depth-First Search algorithm which decides, for each node of the tree (as described in section 3), if it is convenient to continue the descent on the child nodes. The program updates $\overline{B_r}$ each time a better approximation is discovered (in procedure Round-1). As the algorithm virtually goes through the whole tree, $\overline{B_r}$ reaches the optimal value Br at the end of the execution. Note also that if the worst case complexity of the algorithm is still exponential, it can be much less in practice (depending on the choice of the initial values $\{B_1, B_2, B_3, ..., B_{r-1}\}$ and $\overline{B_r}$).

Algorithme 1 branch-and-bound

input: $\{B_1, B_2, B_3, ..., B_{r-1}\}$ and $\overline{B_r}$

procedure Round-r:
for each candidate for χ_{I_r} **do**
  let $\epsilon_r = max_{\chi_{O_r}}(\chi_{I_r}, \chi_{O_r})$
  if $[B_{r-1}, \epsilon_r] \geq \overline{B_r}$ **then**
    call **procedure** Round-$(r-1)$
  end if
end for
return $\overline{B_r}$
exit the program

procedure Round-i $(2 \leq i \leq r-1)$:
for each candidate for χ_{I_i} **do**
  let $\chi_{O_i} = \chi_{I_{i+1}}$ and $\epsilon_i = (\chi_{I_i}, \chi_{O_i})$
  if $[B_{i-1}, \epsilon_i, ..., \epsilon r] \geq \overline{B_r}$ **then**
    call **procedure** Round-$(i-1)$
  end if
end for
return to the upper **procedure**

procedure Round-1:
let $\chi_{O_1} = \chi_{I_2}$ and $\epsilon_1 = max_{\chi_{I_1}}(\chi_{I_1}, \chi_{O_1})$
if $[\epsilon_1, \epsilon_2, ..., \epsilon_r] \geq \overline{B_r}$ **then**
  $\overline{B_r} = [\epsilon_1, \epsilon_2, ..., \epsilon_r]$
end if
return to the upper **procedure**

5. IMPROVING THE TIME COMPLEXITY OF MATSUI'S LINEAR CRYPTANALYSIS

Improving the Time Complexity of Matsui's Linear Cryptanalysis

B. Collard*, F.-X. Standaert**, J.-J. Quisquater

UCL Crypto Group, Université Catholique de Louvain

Abstract. This paper reports on an improvement of Matsui's linear cryptanalysis that reduces the complexity of an attack with *algorithm 2*, by taking advantage of the Fast Fourier Transform. Using this improvement, the time complexity decreases from $O(2^k * 2^k)$ to $O(k * 2^k)$, where k is the number of bits in the keyguess. This improvement is very generic and can be applied against a broad variety of ciphers including SPN and Feistel schemes. In certain (practically meaningful) contexts, it also involves a reduction of the attacks data complexity (which is usually the limiting factor in the linear cryptanalysis of block ciphers). For illustration, the method is applied against the AES candidate Serpent and the speed-up is given for exemplary attacks.

Keywords: block ciphers, linear cryptanalysis, Fast Fourier Transform.

1 Introduction

The linear cryptanalysis [1] is one of the most powerful attacks against modern block ciphers in which an adversary exploits a linear approximation of the type:

$$P[\chi_P] \oplus C[\chi_C] = K[\chi_K] \tag{1}$$

In this expression, P, C and K respectively denote the plaintext, ciphertext and the secret key while $A[\chi]$ stands for $A_{a_1} \oplus A_{a_2} \oplus ... \oplus A_{a_n}$,with $A_{a_1}, ..., A_{a_n}$ representing particular bits of A in positions $a_1, ..., a_n$ (χ is usually denoted as a mask). In practice, linear approximations of block ciphers can be obtained by the concatenation of one-round approximations and such concatenations (also called characteristics) are mainly interesting if they maximize the deviation (or bias) $\epsilon = p - \frac{1}{2}$ (where p is the probability of a given linear approximation).

In its original paper, Matsui described two methods for exploiting the linear approximations of a block cipher, respectively denoted as *algorithm 1* and *algorithm 2*. In the first one, given an r-round linear approximation with sufficient bias, the algorithm simply counts the number of times the left side of Equation 1 is equal to zero for N pairs (plaintext, ciphertext). If $T > N/2$, then it assumes either $K[\chi_K] = 0$ if $\epsilon > 0$ or $K[\chi_K] = 1$ if $\epsilon < 0$ so that the experimental value $(T - N/2)/N$ matches the theoretical bias. If $T > N/2$, an opposite reasoning holds. For the attack to be successful, it is shown in [1] that the number of available (plaintext, ciphertext)-pairs must be proportional to $\frac{1}{\epsilon^2}$.

* Work supported by the project Nanotic-Cosmos of the Walloon Region.

** Postdoctoral researcher of the Belgian Fund for Scientific Research (FNRS).

In the second method *algorithm 2*, an r-1-round characteristic is used and a partial decryption of the last round is performed by guessing the key bits involved in the approximation. As a consequence, all the guessed key bits can be recovered rather than the parity $K[\chi_K]$ which yields much more efficient attacks in practice. Moreover, as it uses a r-1-round characteristic instead of a r-round one for *algorithm 1*, it has a smaller data complexity. However, this improved efficiency has its counterpart in a higher computational complexity, due to the use of possibly large key guesses (*i.e.* involving a large number of key bits).

In this paper, we consequently introduce a general method for improving the time complexity of a linear cryptanalysis attack using *algorithm 2*. Although the limiting factor for linear cryptanalysis attacks is usually the data complexity, such an improvement is relevant and can be motivated both by practical and theoretical reasons, as the following scenarios underline.

- In the evaluation of linear cryptanalysis attacks against modern block ciphers, the attacker usually does a tradeoff between the bias of the approximation and the size of the keyguess. For example, when targeting 10 rounds of Serpent in [6], the authors found a 9-round approximation with bias 2^{-52} but with 92 bits of keyguess (23 active Sboxes). As this leads to an attack with time complexity $O(2^{184})$, they had to choose an approximation with a smaller bias (namely 2^{-58}) but only 44 bits of keyguess. Using our improvement, we can take advantage of the 2^{-52} bias with a time complexity of about $2^{98.5}$, thus reducing the data complexity from 2^{118} to 2^{106}.
- Since most recent ciphers (*e.g.* the AES candidates) have strong diffusion properties, the number of active S-boxes in linear cryptanalysis attacks against their reduced-round versions is usually too high for the time complexity of these attacks to be tractable. The improvement proposed in this paper can consequently be used to perform actual cryptanalytic experiments against these reduced-round ciphers. Therefore, we expect that it will lead to a better understanding of certain open issues, *e.g.* about the exploitation of multiple approximations in linear cryptanalysis.

Independently of these theoretical expectations, we believe that the proposed improvement of the time complexity, from $O(2^k * 2^k)$ to $O(k * 2^k)$ (where k is the number of bits in the keyguess) is meaningful in itself. We note finally that the idea of taking advantage of the Fast Fourier Transform to speed-up the computations in cryptanalysis is not new. For example [3] and [4] describe FFT-based techniques to improve correlation attacks against stream ciphers. However, we are not aware of any publication mentioning explicitly the applicability of the FFT to the linear cryptanalysis of block ciphers.

The rest of the paper is structured as follows. Section 2 describes a generic framework for the analysis of Matsui's linear cryptanalysis using *algorithm 2*. Section 3 details our improved key guess strategy and Section 4 applies our technique to improve some previous cryptanalytic results against the AES candidate Serpent. Finally, our conclusions are in Section 5.

2 General framework for *algorithm 2*

Suppose a linear approximation on r rounds with bias ϵ, requiring $N \approx O(1/\epsilon^2)$ known plaintext-ciphertext pairs for a successful attack. Moreover, this approximation has q active S-boxes in the last round and k bits to guess. In the original *algorithm 2* proposed by Matsui, a partial decryption of the last round is performed for every ciphertext by guessing the key bits involved in the approximation. The parity of the approximation for the plaintext and the partially decrypted ciphertext is then evaluated and a counter corresponding to the guess is incremented if the relation holds, decremented otherwise. The key candidate with the highest counter in absolute value is finally assumed to be the correct key. As a partial decryption is proceeded for every ciphertext and every keyguess, the time complexity of this algorithm is in $O(N \cdot 2^k)$ partial decryptions.

However, as we only consider a limited number of bits (those in the active S-boxes) during the partial decryption of the ciphertexts, the same work is done many times. Indeed, the number of texts required to mount an attack is typically largely superior to the size of the keyguess (*i.e.* $N \gg 2^k$). On the basis of this observation, Matsui proposed in [2] an improvement which considerably reduces the time complexity of an attack. Although it was first applied to the DES, this improvement is valid in the general case. The modified algorithm can be divided in 2 phases (according to the framework proposed in [7] sec. 2.1):

Distillation phase (for each generated ciphertext)**:**

- Initialize an array of 2^k counters.
- For each generated ciphertext, extract the k-bit value corresponding to the active S-boxes and evaluate the parity of the plaintext subset defined by the approximation. Increment or decrement the counter corresponding to the extracted k-bit value according to the parity.

Analysis phase (once all the ciphertext have been generated)**:**

- For each k-bit ciphertext and k-bit subkey, partially decrypt the k-bit ciphertext under the k-bit subkey and evaluate the parity of the output subset (as defined by the linear approximation). Keep this value in a table of size $2^k \cdot 2^k$.
- For each k-bit subkey, evaluate its experimental bias by checking, for each k-bit ciphertext, the parity of the approximation and the value of the corresponding counter. Then output the subkey which has maximal bias.

During the distillation phase, we construct a table that indexes for each k-bit ciphertext, the difference between the frequency of its apparition leading to a null input parity and the frequency leading to a non-null input parity. This information is sufficient to evaluate the bias of the approximation for each key during the analysis phase. This process can be done "on the fly", while the plaintexts are being encrypted. As only simple operations like bit extractions and incrementations are performed during this phase, its complexity is generally assumed to be negligible compared to the one of the encryption process.

During the analysis phase, the actual bias for each subkey candidate is evaluated. In order to avoid multiple evaluations of the same operation, a table is constructed which indexes, for each k-bit ciphertext and each subkey candidate, the parity of the output subset obtained after the partial decryption of the ciphertext XORed with the subkey. For a given subkey candidate, its bias can then be evaluated by summing, for each k-bit ciphertext, the corresponding counter, taking the parity of the approximation for the given ciphertext and subkey into account (this parity is given by the sign of the counter and the correct index in the precomputed table). In this way, the table is accessed 2^k times for each possible subkey, leading to a total time complexity of $O(2^k \cdot 2^k)$, compared to the $O(N \cdot 2^k)$ operations for a naive implementation of *algorithm 2*. Importantly, this complexity depends only on the number of subkey candidates and not on the number of texts used. Note finally that the table can be computed row by row in order to save memory space.

3 Improving the framework

In this section, we present a simple but powerful modification of the above algorithm that allows us to significantly decrease the time complexity of an attack. As the modification concerns only the analysis phase, the distillation phase remains unchanged and so does the data complexity.

3.1 Rewriting the algorithm

The table defined during the analysis phase can be seen as a large matrix $\boldsymbol{C}$ of size $2^k \cdot 2^k$ defined by the following function:

$$\boldsymbol{C}(i,j) = parity(\mathrm{S}^{-1}(i \oplus j)) \qquad (0 \leq i,j \leq 2^k - 1) \tag{2}$$

where $\mathrm{S}^{-1}(l)$ represents the inverse of the last layer of S-boxes for the k-bit digit l and $parity()$ is a function mapping any k-bit subset to ± 1 according to its parity ($+1$ if the parity of the subset is zero, -1 otherwise). With such a definition, the bias ϵ_i corresponding to a particular keyguess i is given by the equation:

$$\epsilon_i = \sum_{j=0}^{2^k-1} parity(\mathrm{S}^{-1}(i \oplus j)) \cdot \boldsymbol{x}(j) = \sum_{j=0}^{2^k-1} \boldsymbol{C}(i,j) \cdot \boldsymbol{x}(j) = \boldsymbol{C}(i,:) \cdot \boldsymbol{x} \tag{3}$$

where $\boldsymbol{x}$ is the vector of counters such as defined in the distillation phase. This equation evaluates the bias of the linear approximation for a particular k-bit subkey candidate i. Consequently, the vector $\boldsymbol{\epsilon}$ of the experimental bias for every subkey candidates can be computed by the matrix-vector product:

$$\boldsymbol{\epsilon} = \boldsymbol{C} \cdot \boldsymbol{x} \tag{4}$$

At this point, the complexity for the evaluation of the experimental biases is still in $O(2^k \cdot 2^k)$ as it implies a matrix-vector product with size 2^k.

3.2 Analysis of the new algorithm

We underline the fact that the matrix $\boldsymbol{C}$ has a very particular structure. Taking this structure into account will allow us to significantly reduce the number of operations required to evaluate the vector $\boldsymbol{\epsilon}$ of the bias.

First, as $\boldsymbol{C} = f(i \oplus j)$ for a known function f, every rows or column of $\boldsymbol{C}$ defines the complete matrix (in particular, $\boldsymbol{C}$ is symetric). For example,

$$\boldsymbol{C}(i,j) = f(i \oplus j) = f((i \oplus j) \oplus 0) = \boldsymbol{C}(i \oplus j, 0) \tag{5}$$

Let us introduce the following definitions (*cfr.* [10]):

Definition 1 (circulant). *A matrice is circulant iff each row (column) vector is rotated one element to the right relative to the preceding row (column) vector.*

Definition 2 (block circulant). *A matrice is m-block circulant iff it is circulant blockwise and the number of blocks in each row (or column) is m.*

Definition 3 (level circulant).

- *(1) A matrice is level-*1 *circulant with type(n) iff it is circulant of size n.*
- *(2) A matrice is level-*2 *circulant with type(m,n) iff it is m-block-circulant and each block is a circulant of size n itself.*
- *(3) A matrice is level-*3 *circulant with type(m,n,o) iff it is a m-block circulant whose blocks are level* 2 *circulant with type (n,o).*
- *(q) A matrice is level-q circulant with type(m,n,o,...) iff it is a m-block circulant whose blocks are level* $q-1$ *circulant with type (n,o,...).*

Proposition 1. *Let* $\boldsymbol{C}(i,j) = f(i \oplus j)$*, then* $\boldsymbol{C}$ *is level-k circulant with type* $\underbrace{(2,2,...,2)}_{k \quad times}$.

Demonstration 1. *Let us define the matrix* $\boldsymbol{M}$ *of size* $(2^k * 2^k)$ *as :* $\boldsymbol{M}(i,j) = i \oplus j (0 \leq i,j \leq 2^k - 1)$*. Let us divide* $\boldsymbol{M}$ *in 4 blocks with size* $(2^{k-1} * 2^{k-1})$ *each:*

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M_{11}} & \boldsymbol{M_{12}} \\ \boldsymbol{M_{21}} & \boldsymbol{M_{22}} \end{pmatrix}$$

Then for $(0 \leq a,b \leq 2^{k-1} - 1)$*:*

- $\boldsymbol{M_{11}}(a,b) = \boldsymbol{M}(a,b) = a \oplus b$,
- $\boldsymbol{M_{12}}(a,b) = \boldsymbol{M}(a, b + 2^{k-1}) = a \oplus b \oplus 2^{k-1}$,
- $\boldsymbol{M_{21}}(a,b) = \boldsymbol{M}(a + 2^{k-1}, b) = a \oplus 2^{k-1} \oplus b$,
- $\boldsymbol{M_{22}}(a,b) = \boldsymbol{M}(a + 2^{k-1}, b + 2^{k-1}) = a \oplus 2^{k-1} \oplus b \oplus 2^{k-1}$,

This is true because $a + 2^{k-1}$ *is equivalent to* $a \oplus 2^{k-1}$ *since* $0 \leq a \leq 2^{k-1} - 1$*. Consequently,* $\boldsymbol{M_{11}} = \boldsymbol{M_{22}}$ *and* $\boldsymbol{M_{12}} = \boldsymbol{M_{21}}$*, thus* $\boldsymbol{M}$ *is 2-block circulant. Moreover,* $\boldsymbol{M_{12}} = \boldsymbol{M_{11}} \oplus 2^{k-1}$*, so it has the same circulant structure as* $\boldsymbol{M_{11}}$*. We can repeat the same reasoning for* $\boldsymbol{M} = \boldsymbol{M_{11}}$ *with* $k = k - 1$*, and so the proposition is proved by induction. Finally, it is obvious that* $f(\boldsymbol{M})$ *keeps the circulant properties if* $f()$ *is applied elementwise.*

3.3 Fast algorithm

We now describe how the properties given above can be used to speed up the linear cryptanalysis. We exploit the following result (*cfr.* [10] for a proof):

Theorem 1. *A circulant $\boldsymbol{C}$ of level k and type (m,n,o,...,r) is diagonalizable by the unitary matrix $\boldsymbol{F} = \boldsymbol{F_m} \otimes \boldsymbol{F_n} \otimes \boldsymbol{F_o} \otimes ... \otimes \boldsymbol{F_r}$:*

$$\boldsymbol{C} = \boldsymbol{F^*} diag(\boldsymbol{\lambda}) \boldsymbol{F}, \tag{6}$$

where $\boldsymbol{\lambda}$ is the vector of eigenvalues of $\boldsymbol{C}$, The symbol $\otimes$ is the Kronecker product and F_n is the Fourier matrix of size $n * n$ defined by:

$$\boldsymbol{F_n}(i,j) = \frac{1}{\sqrt{n}} w^{i \cdot j} \quad (0 \leq i, j \leq n-1), \tag{7}$$

with:

$$w = e^{\frac{2\pi\sqrt{-1}}{n}} \tag{8}$$

The matrix $\boldsymbol{F}$ is the k-dimensional Discrete Fourier Transform matrix. Therefore, the multidimensional Fast Fourier Transform allows us to quickly compute the matrix-vector product with $\boldsymbol{F}$ or $\boldsymbol{F^*}$. Using the FFT, the complexity of this product decrease from $O(n^2)$ to $O(n\ log_2(n))$ [9].

Proposition 2. *The eigenvalues vector $\boldsymbol{\lambda}$ of a circulant matrix $\boldsymbol{C}$ of level k and type (m,n,o,...,r) can be computed with the following matrix-vector product:*

$$\boldsymbol{\lambda} = \boldsymbol{FC}(:,1)\sqrt{mno...r}, \tag{9}$$

where $\boldsymbol{C}(:,1)$ means (using Matlab notation) we take the first column of $\boldsymbol{C}$.

Demonstration 2. *From theorem 1, it follows that:*

$$\boldsymbol{C} = \boldsymbol{F^*} diag(\lambda) \boldsymbol{F} \tag{10}$$

Multiplying both sides by $\boldsymbol{F}$, this gives (as $\boldsymbol{FF^*} = \boldsymbol{I}$):

$$\boldsymbol{FC} = diag(\lambda) \boldsymbol{F} \tag{11}$$

If we consider the first column only, this reduces to:

$$(\boldsymbol{FC})(:,1) = (diag(\lambda)\boldsymbol{F})(:,1) = \boldsymbol{\lambda} \circ \boldsymbol{F}(:,1) \tag{12}$$

From equation 7 and the definition of $\boldsymbol{F}$, it follows that:

$$\boldsymbol{F}(:,1) = \frac{1}{\sqrt{mno...r}} (1,1,1...1)^T \tag{13}$$

Consequently,

$$\boldsymbol{\lambda} = (\boldsymbol{FC})(:,1)\sqrt{mno...r} = \boldsymbol{FC}(:,1)\sqrt{mno...r} \tag{14}$$

Hence, the matrix-vector product $\boldsymbol{\epsilon} = \boldsymbol{C}\boldsymbol{x}$ is equivalent to $\boldsymbol{\epsilon} = \boldsymbol{F}^* diag(\boldsymbol{\lambda})\boldsymbol{F}\boldsymbol{x}$. The eigenvalues of $\boldsymbol{C}$ can be computed using the formula: $\boldsymbol{\lambda} = \boldsymbol{F}\boldsymbol{C}(:,1)\sqrt{2^k}$. Therefore, the matrix-vector product can be computed using the three following matrix-vector products: $\boldsymbol{y} = \boldsymbol{F}\boldsymbol{x}$, $\boldsymbol{z} = \boldsymbol{F}\boldsymbol{C}(:,1)\sqrt{2^k}$ and $\boldsymbol{\epsilon} = \boldsymbol{F}^*(\boldsymbol{z} \circ \boldsymbol{y})$ (where $\circ$ is the Hadamard product). As each of these three products involves the matrix $\boldsymbol{F}$, the complete computation can be made by applying only three k-dimension FFTs of size 2^k, leading to a complexity of $3 \cdot 2^k \cdot log_2(2^k) = 3 \cdot k \cdot 2^k$. The Matlab implementation code for this improved strategy is given below (see Algorithm 1). As a typical numerical example, for a $2^{20} * 2^{20}$ matrix $\boldsymbol{C}$, the matrix-vector product is computed in less than 5 seconds on a Pentium D 3.20GHz.

Algorithme 1 Matlab code

```
function b=product(C,x)

% compute the product Cx by the mean of the fft
% C is a level k circulant matrix of type (2,2,2,...,2).
% C is completely specified by its first column
% x is an unspecified vector

k= log2(size(C,1));

% compute F*x:
x= reshape(x,2*ones(k));
prod1= fftn(x);
prod1= reshape(prod1,2^k,1);

% compute the eigenvalues of C
c= reshape(C(:,1), 2*ones(k));
eig= fftn(c);
eig= reshape(eig,2^k,1);

% compute eig*F*x
prod2= eig.*prod1;

% compute b=F'*eig*F*x
b= reshape(prod2,2*ones(k));
b= ifftn(b);
b= reshape(b,2^k,1);

return b;
```

3.4 Implication for multiple linear approximations

In context of multiple linear approximations (*cfr.* [11], [7]), more speed-up may be achievable. Every input mask defines its own vector of counter $\boldsymbol{x}$, while every output mask defines a different matrix $\boldsymbol{C}$. Consequently, the use of multiple approximations with the same active S-boxes but different input masks (as it is usually the case) requires to compute the eigenvalues only once, as the matrix $\boldsymbol{C}$ remains the same. Thus, the time complexity of linear cryptanalysis with n approximations is reduced to the computation of $2n+1$ FFTs instead of $3n$ FFTs. This involves an additional reduction of up to 33% for the time complexity.

3.5 Extension to key additions *modulo* 2^k

In certain ciphers, the mixing with the key material is done using a modular addition instead of a XOR with the subkey (practical examples include [12], [13]). A similar approach as in the previous sections can be applied to reduce the time complexity of such systems.

Definition 4 (left-circulant). *A matrice is left-circulant iff each row (column) vector is left-rotated by one element relative to the preceding row (column) vector.*

Proposition 3. *Let* $\boldsymbol{C_{left}}(i,j) = f(i+j \bmod 2^k)$ $(0 \leq i, j \leq 2^k - 1)$*, then* $\boldsymbol{C_{left}}$ *is left-circulant.*

Demonstration 3. *For every* λ, $\boldsymbol{C_{left}}(a+\lambda, b-\lambda) = f(a+\lambda+b-\lambda \bmod 2^k) = f(a+b \bmod 2^k)$*. Thus, all the element in the same increasing diagonal of the matrix are equal. Moreover, the value in a diagonal is repeated every* 2^k *diagonal due to the mod* 2^k*, and so the matrix is left-circulant.*

We can easily convert a left-circulant matrix $\boldsymbol{C_{left}}$ to a circulant matrix $\boldsymbol{C}$ with the same first row thanks to a particular matrix of permutation $\boldsymbol{\Gamma}$:

$$\boldsymbol{C_{left}} = \boldsymbol{\Gamma C}, \tag{15}$$

where:

$$\boldsymbol{\Gamma} = \begin{pmatrix} 1\,0\,0\,\ldots\,0\,0 \\ 0\,0\,0\,\ldots\,0\,1 \\ 0\,0\,0\,\ldots\,1\,0 \\ \vdots\,\vdots\,\vdots\quad\quad\vdots\,\vdots \\ 0\,1\,0\,\ldots\,0\,0 \end{pmatrix}$$

This requires the application of 2^k row permutations and it is useful for the following result that we require in our investigations (see [10] pp.72-73):

Theorem 2. *A circulant* $\boldsymbol{C}$ *is diagonalizable by the Fourier matrix* $\boldsymbol{F}$ *of size* 2^k*:*

$$\boldsymbol{C} = \boldsymbol{F}^* diag(\boldsymbol{\lambda}) \boldsymbol{F}, \tag{16}$$

where $\boldsymbol{\lambda}$ *is the vector of eigenvalues of* $\boldsymbol{C}$*.*

Again, we can easily compute the matrix-vector product between $\boldsymbol{C}$ and $\boldsymbol{x}$ using 3 (one-dimensional) FFTs. The algorithm is roughly the same as described above, except that the multi-dimensional FFTs are replaced by one-dimensional FFTs and that we must perform a last permutation $\boldsymbol{\Gamma}$ in the end (in order to switch from a right- to a left-circulant matrix). Finally, for a matrix $\boldsymbol{C_{left-toeplitz}}$:

$$\boldsymbol{C_{left-toeplitz}}(i,j) = f(i + j \ mod \ 2^m); \ (0 \leq i,j \leq 2^k - 1), \ m \neq k, \qquad (17)$$

with the left-Toeplitz structure, it can be shown that the matrix-vector product can be computed by embedding the $2^k * 2^k$ matrix in a $2^{k+1} * 2^{k+1}$ matrix with left-circulant structure, leading to a complexity of $O(2 * (k+1) * 2^{k+1})$.

4 Practical improvements

The improved algorithms described above can be straightforwardly applied to improve previous cryptanalytic results. For illustrations purposes, we applied them to the AES candidate Serpent [5]. Using the FFT to compute the linear approximation biases for the subkey candidates allows speeding-up the attack of Biham *et al.* as summarized in Table 1. In Table 2, we additionally report on the improved linear and multiple linear cryptanalysis of Serpent, described in [8].

Rounds	Type of attack	complexity		
		data	time	memory
10	Lin.Cryptanalysis[6]	2^{118}KP	$2^{88} \rightarrow 2^{51}$	2^{44}
	Lin.Cryptanalysis[6]	2^{116}KP	$2^{96} \rightarrow 2^{55.2}$	2^{48}
	Lin.Cryptanalysis[6]	2^{106}KP	$2^{184} \rightarrow 2^{100.1}$	2^{92}
11	Lin.Cryptanalysis[6]	2^{118}KP	$2^{214} \rightarrow 2^{148.7}$	$2^{88} \rightarrow 2^{140}$
KP - Known Plaintexts Complexity is measured in number of arithmetic operation. Memory is mesured in Bytes.				

Table 1: Previous and improved attacks on Reduced-rounds Serpent.

We note that, as previously mentioned, certain improvements are particularly relevant with respect to the experimental testing of the attacks. For example, targeting 7 rounds of Serpent with a multiple linear cryptanalysis attack appears as a reasonable target thanks to time complexity reduction. Using special dedicated hardware like *Copacobana* [14], it could even be possible to attack up to 8 rounds Serpent. The reduced time complexity also allows considering the exploitation of better biased linear approximations with larger key guesses and therefore to reduce the data complexity of the best reported attacks. For example, Table 1 includes the scenario described in the introduction of this paper: moving from a time complexity of 2^{184} to a time complexity of 2^{100} allows to reduce the attack data complexity from 2^{118} to 2^{106}. This example clearly

Rounds	Type of attack	complexity		
		data	time	memory
7	Lin.cryptanalysis	2^{52}KP	$2^{40} \rightarrow 2^{25.9}$	2^{20}
	Mult.Lin.Cryptanalysis(8 appr.)	2^{47}KP	$2^{43} \rightarrow 2^{28.4}$	2^{23}
8	Lin.cryptanalysis	2^{62}KP	$2^{56} \rightarrow 2^{34.4}$	2^{28}
	Mult.Lin.Cryptanalysis(8 appr.)	2^{57}KP	$2^{59} \rightarrow 2^{36.9}$	2^{31}
	Mult.Lin.Cryptanalysis(104 appr.)	2^{55}KP	$2^{62.7} \rightarrow 2^{40.5}$	$2^{34.7}$
9	Lin.cryptanalysis	2^{80}KP	$2^{88} \rightarrow 2^{51}$	2^{44}
	Mult.Lin.Cryptanalysis(128 appr.)	2^{71}KP	$2^{95} \rightarrow 2^{57.5}$	2^{51}
	Mult.Lin.Cryptanalysis(3712 appr.)	2^{68}KP	$2^{99.9} \rightarrow 2^{62.3}$	$2^{55.9}$
10	Lin.cryptanalysis ($\epsilon = 2^{-55}$)	2^{112}KP	$2^{88} \rightarrow 2^{51}$	2^{44}
	Mult.Lin.Cryptanalysis(2048 appr.)	2^{99}KP	$2^{99} \rightarrow 2^{61.5}$	2^{55}
	Lin.cryptanalysis ($\epsilon = 2^{-59}$)	2^{120}KP	$2^{64} \rightarrow 2^{38.6}$	2^{32}
	Mult.Lin.Cryptanalysis(2048 appr.)	2^{107}KP	$2^{75} \rightarrow 2^{49}$	2^{43}
11	Lin.cryptanalysis ($\epsilon = 2^{-58}$)	2^{118}KP	$2^{178} \rightarrow 2^{116.3}$	2^{108}

Table 2: Additional improved attacks on Reduced-rounds Serpent (see [8]).

emphasize the practical impact of our result on the overall complexity of linear cryptanalysis attacks.

It is also possible to attack 11 rounds Serpent with this technique. We use a 9-rounds linear approximation starting and ending with S-box 9. This approximation was generated similarly to the ones presented in [8]. It has a 2^{-58} bias and a total of 27 active S-boxes (15 in the first round and 12 in the last round). The attack follows the same principle as the one presented so far, except that we must also perform a partial encryption in the begining of the cipher. We first define an array $\boldsymbol{x}$ of 2^{108} counters in the folling way: for each plaintext-ciphertext pairs, we extract the 108-bits value corresponding to the active S-boxes and we increment the corresponding counter. Then we define a matrix $\boldsymbol{C}$ of size $2^{108} * 2^{108}$ as:

$$\boldsymbol{C(i,j)} = parity(S(i_{1:60} \oplus j_{1:60}) || S^{-1}(i_{61:108} \oplus j_{61:108})) \tag{18}$$

That is to say, $\boldsymbol{C(i,j)}$ is the parity of the linear approximation after partial en/decryption of the 108-bit text i with 108-bit subkey j. As seen previously, the experimental bias for any keyguess is given by the matrix-vector product $\boldsymbol{C} \cdot \boldsymbol{x}$. Thanks to the level circulant structure of $\boldsymbol{x}$, the time complexity is equal to $3 \cdot 108 \cdot 2^{108} = 2^{116}$. Without this trick, the time complexity would have been $2^{88} + 2^{60} \cdot (2^{118} + 2^{88}) = 2^{178}$ (the details are in [6]). For comparison purpose, the best known attack on Serpent targets 11 rounds using a combination of linear and differential cryptanalysis techniques [15]. It has a data complexity of $2^{125.3}$ choosen plaintexts, $2^{139.2}$ encryptions and 2^{60} bytes of memory.

5 Conclusion and further works

In this paper, we presented an improvement of Matsui's linear cryptanalysis that reduces the time complexity of an attack using *algorithm 2* from $O(2^k * 2^k)$ to $O(k * 2^k)$ partial decryptions, where k is the number of bits in the keyguess. Moreover, in the case of multiple linear cryptanalysis, additional speed-ups can be reached. This improvement is very generic and can be applied against a broad variety of ciphers including SPN and Feistel schemes. For illustration purposes, we applied the method to the block cipher Serpent and exhibited the reduced complexities of some (state-of-the-art) exemplary attacks. As a scope for further research, the exploitation of the improved time complexity of attacks using multiple linear approximations should allow performing actual experiments and therefore evaluate the validity of certain assumptions detailed in [7].

Acknowledgements

The authors thank Pr. Paul Van Dooren from UCL\Inma for his helpful answers and advices. We also thank anonymous reviewers from SAC and ICISC for their useful comments.

References

1. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of Eurocrypt 1993, LNCS, vol. 765, pp. 386-397, Lofthus, Norway, May 1993.
2. M. Matsui, *The First Experimental Cryptanalysis of the Data Encryption Standard*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Science, vol. 839, pp. 1-11, Santa Barbara, California, USA, August 1994.
3. P. Chose, A. Joux, M. Mitton, *Fast Correlation Attacks: An Algorithmic Point of View*, in the proceedings of Eurocrypt 2002, Lecture Notes in Computer Science, vol. 2332, pp.209-221, Amsterdam, The Netherlands, May 2002.
4. Y. Lu, W. Meier, S. Vaudenay, *The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption*, in the proceedings of CRYPTO 2005, LNCS, vol. 3621, pp. 97-117, Santa Barbara, California, USA, August 2005.
5. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, in the proceedings of the First Advanced Encryption Standard (AES) Conference, Ventura, CA, 1998.
6. E. Biham, O. Dunkelman, N. Keller, *Linear Cryptanalysis of Reduced Round Serpent*, in the Proceedings of FSE 2001, Lecture Notes in Computer Science, vol. 2355, pp. 16-27, Yokohama, Japan, April 2001.
7. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of CRYPTO 2004, Lecture Notes in Computer Science, vol. 3152, pp.1-22, Santa Barbara, California, USA, August 2004.
8. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent*, to appear in the proceedings of InsCrypt 2007.
9. J.W. Cooley and J.W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comput. 19, pp. 297301, 1965.
10. P.J. Davis, *Circulant Matrices*, Wiley-Interscience, pp. 176-191, NY, 1979.
11. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Sciences, vol. 839, pp. 26-39, Santa Barbara, California, USA, August 1994.
12. X. Lai, J.L. Massey, *A Proposal for a New Block Encryption Standard*, in the proceedings of Eurocrypt 1990, Lecture Notes in Computer Science, vol. 473, pp. 389-404, Aarhus, Denmak, May 1990.
13. D.J. Wheeler, R.M. Needham, *TEA, a Tiny Encryption Algorithm*, in the proceedings of Fast Software Encryption 1994, Lecture Notes in Computer Science, vol. 1008, pp. 363-366, Leuven, Belgium, December 1994.
14. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, M. Schimmler, *Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker*, in the proceedings of Cryptographic Hardware and Embedded Systems - Ches 2006, Lectrure Notes in Computer Science, vol.4249, Springer, 2006.
15. E. Biham, O. Dunkelman, N. Keller, *Differential-linear Cryptanalysis of Serpent*, in the Proceedings of Fast Software Encryption 2003, Lecture Notes in Computer Science, vol. 2887, pp. 9-21, Springer, 2004.

Part II

DEDICATED CRYPTANALYSIS OF THE BLOCK CIPHER PRESENT

6. STATISTICAL SATURATION ATTACK AGAINST THE BLOCK CIPHER PRESENT

A Statistical Saturation Attack against the Block Cipher PRESENT

B. Collard*, F.-X. Standaert**

UCL Crypto Group, Microelectronics Laboratory, Université catholique de Louvain, Place du Levant 3, Louvain-la-Neuve, Belgium

baudoin.collard;fstandae@uclouvain.be

Abstract. In this paper, we present a statistical saturation attack that combines previously introduced cryptanalysis techniques against block ciphers. As the name suggests, the attack is statistical and can be seen as a particular example of partitioning cryptanalysis. It extracts information about the key by observing non-uniform distributions in the ciphertexts. It can also be seen as a dual to saturation (*aka* square, integral) attacks in the sense that it exploits the diffusion properties in block ciphers and a combination of active and passive multisets of bits in the plaintexts. The attack is chosen-plaintext in its basic version but can be easily extended to a known-plaintext scenario. As an illustration, it is applied to the block cipher PRESENT proposed by Bogdanov *et al.* at CHES 2007. We provide theoretical arguments to predict the attack efficiency and show that it improves previous (linear, differential) cryptanalysis results. We also provide experimental evidence that we can break up to 15 rounds of PRESENT with $2^{35.6}$ plaintext-ciphertext pairs. Eventually, we discuss the attack specificities and possible countermeasures. Although dedicated to PRESENT, it is an open question to determine if this technique improves the best known cryptanalysis for other ciphers.

Introduction

This paper introduces a statistical attack that is closely related to previous works in *partitioning cryptanalysis* [2, 8, 9]. Such attacks can be seen as a generalization of the linear cryptanalysis in which one exploits partitions of the plaintexts (*resp.* ciphertexts) leading to significantly non uniform distributions of the ciphertexts (*resp.* plaintexts). While arguably more powerful than linear cryptanalysis, they usually face the question of how to find good partitions for a given cipher. Hence, in practice they generally rely on some specificities that a cryptanalyst may find within the ciphers, *e.g.* in [7, 15]. Following these works, our results focus on a (relatively) generic and simple way to find partitions that can, in certain contexts, lead to efficient attacks. For this purpose, we exploit ideas from the *integral cryptanalysis* [13], originally introduced as a specialized attack against the SQUARE block cipher [6] and also known as saturation attacks [11]. Such attacks are chosen-plaintext and generally study the propagation of well chosen

* Work supported by the Walloon Region under the project Nanotic-Cosmos.

** Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

sets of plaintexts through the cipher. In practice, they typically fix a number of plaintext bytes to a constant value and track the evolution of some other bytes having a known distribution. To some extent, the proposed statistical saturation attack can be seen as a dual of the previous saturation attacks. It also takes advantage of several plaintexts with some bits fixed while the others vary randomly. But instead of observing the evolution of the variable bits in the cipher state, we observe the diffusion of the fixed bits during the encryption process. That is, we track the evolution of a non-uniform input plaintext distribution through the cipher. The name statistical saturation attack refers both to the the way the inputs are generated and to the fact that it exploits the diffusion properties (and possibly weaknesses) of the target cipher.

As an illustration, we apply the proposed technique to the block cipher PRESENT that was presented at CHES 2007 by Bogdanov *et al.* It is a compact block cipher primarily designed for hardware constrained environments such as RFID tags and sensor networks. The name PRESENT reflects its similarity with the block cipher SERPENT [1], known for its security and hardware performances. In the specifications of the cipher [4], the authors analyze the security of PRESENT against various cryptanalytic attacks. In order to argue about the immunity against linear and differential cryptanalysis, they provide lower bounds for the number of active S-boxes in any linear/differential trail of the cipher. Resistance against structural, algebraic and related-key attacks is also analyzed. The security of PRESENT against differential cryptanalysis was further studied in [17] in which the authors present an attack against 16 rounds that requires the entire codebook and a time complexity of 2^{65} memory accesses.

PRESENT is a good target for the proposed statistical saturation attack because it exhibits a particular weakness in its diffusion layer. As a consequence, our following results improve the complexities of the best reported attacks against this cipher, both in theoretical estimations and in experimental validations. In practice, we broke up to 15 rounds of PRESENT with $2^{35.6}$ plaintext-ciphertext pairs. Additionally to these results, we discuss the specificities of the attack compared to other known cryptanalytic techniques. We show that it depends both on the diffusion and substitution layers in a block cipher. We also show that current criteria for S-box design do not properly capture the non-uniformities that are exploited in our partitions. Due to the generality of its principles, the proposed technique could be applied to other ciphers as well. However, since its effectiveness depends on the diffusion properties of the targets, it is an open question to determine if it can improve other cryptanalytic results.

The rest of this paper is divided in three parts. Section 1 presents the basic principles of our attack with theoretical arguments that support it. A comparison between theoretical predictions and experimental observations is also provided. The second section extends the basic profiling attack of Section 1 to a distinguishing attack that is more efficient, both in terms of computations and data complexity. Section 3 discusses countermeasures and the impact of the S-boxes on the attack performances. We conclude the paper and suggest further research in Section 4. The PRESENT block cipher is additionally described in Appendix.

1 A basic profiling attack

1.1 Principle of the attack

Our attack is based on a weakness in the diffusion layer of PRESENT. A closer look at the permutation shows that, *e.g.* for the S-boxes 5,6,9 and 10 (counting from S-box 0 at the right), only 8 out of 16 input bits are directed to other S-boxes. Figure 1 illustrates this observation. Note that there exists many other examples of poor diffusion in the permutation (*i.e.* with 8 bits out of 16 remaining in the same 4 S-boxes after permutation). Consequently, if we fix the 16 bits at the input of the S-boxes 5-6-9-10, then 8 bits will be known at the very same input for the next round. We can iteratively repeat this process round by round and observe a non-uniform behavior at the output of the S-boxes 5-6-9-10.

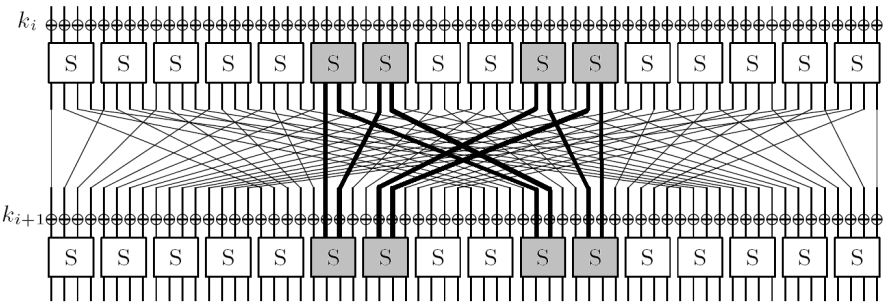

Fig. 1: Permutation layer of PRESENT: bold lines underline the poor diffusion property.

In order to exploit this weakness, we first evaluate theoretically the distribution of the 8 bits in the bold trail of Figure 1 at the output of the S-box layer, for a fixed value of the same 8 bits of plaintext. This requires to guess the 8 subkey bits involved in the trail. One also needs to assume that the bits not situated in the trail are uniformly distributed. This is a reasonable assumption as soon as the 56 remaining bits of plaintext (excluding the 8 bits in the trail) are randomly generated. Then, given the distribution of the 8-bit trail at the input of a round, it is possible to compute the 8-bit distribution at the output of the round with Algorithm 1 (given in Appendix B). By iteratively applying this algorithm, we can compute the distribution for an arbitrary number of rounds. For each key guess, the work needed to compute the theoretical distribution of the target trail after r rounds is equivalent to $r \cdot 2^{16}$ partial encryptions.

Once we have computed the theoretical distributions of the trail for each possible key guess, we can attack the cipher by simply comparing them with a practical distribution obtained by encrypting a large number of plaintexts with a secret key. The key guess minimizing the distance between theoretical and practical distributions is chosen as the correct key. As in [3], we can construct a list of key candidates sorted according to the distance between theory and practice. The better the position of the right key in the list, the better the attack.

1.2 Experimental results

We evaluated the practicability of our attack against a reduced-round version of PRESENT. In order to reduce the guess work, the key-scheduling of PRESENT was simplified in these experiments and the same subkey was used at each round. With this modification, only 8 bits of the master key have to be estimated and the correct distribution has to be found among 256 possible ones.

Comparison between theoretical and experimental distributions. Figure 2 depicts the distribution of the 8 bits in the trail after 2,4,6 and 8 rounds for a fixed 8-bit key byte. The theoretical predictions (in black) are compared with experimental results (in grey) generated with 2^{30} plaintexts-ciphertexts pairs. Note that our attack is choosen-plaintext as we have to fix 8 plaintext bits. But it can be turned into a known-plaintext attack by dividing random plaintexts in 256 classes according to the value of the 8 fixed input bits in the trail and observing the output distributions for each of the 256 cases independently.

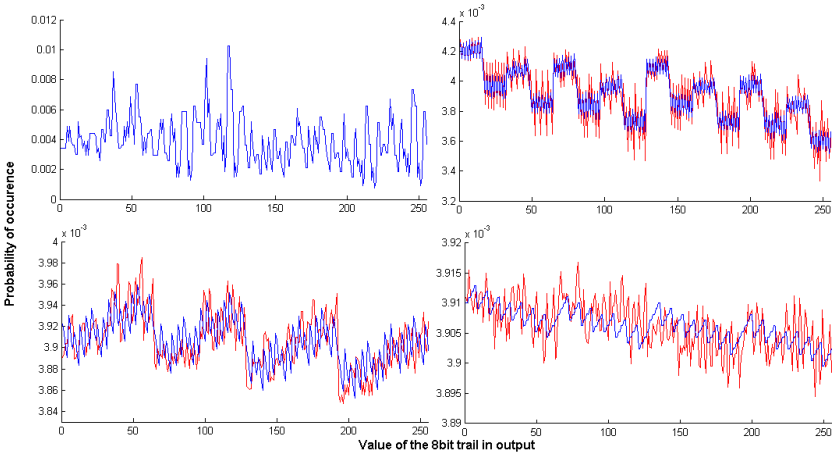

Fig. 2: Comparison between the experimental (in gray) and theoretical (in black) distributions of the target trail output for a given key byte and 2, 4, 6 and 8 rounds.

Both experimental and theoretical distributions present a significant deviation from uniform, even after 8 rounds. The deviation tends to decrease with the number of rounds however. We can observe that predictions match experiments very closely for up to 6 rounds, and then begin to distinguish. This illustrates that the sampling is not sufficient anymore to approximate the distributions.

Comparison between theoretical and experimental distances. Figure 3 shows the evolution of the distance between the distribution corresponding to a secret key byte 32 and the distributions corresponding to the 256 possible

key guesses for this secret key byte. The number of rounds in the figure again varies from 2 to 8. The black (*resp.* grey) curves represent the distance between the theoretical (*resp.* experimental) distribution of the correct key byte and the theoretical distributions for each possible key guess. For up to 8 rounds, we observe that position 32 minimizes the distance between theory and practice. Note that we used both an Euclidean and a Kullback-Leibler distance in our experiments: both metrics gave similar results.

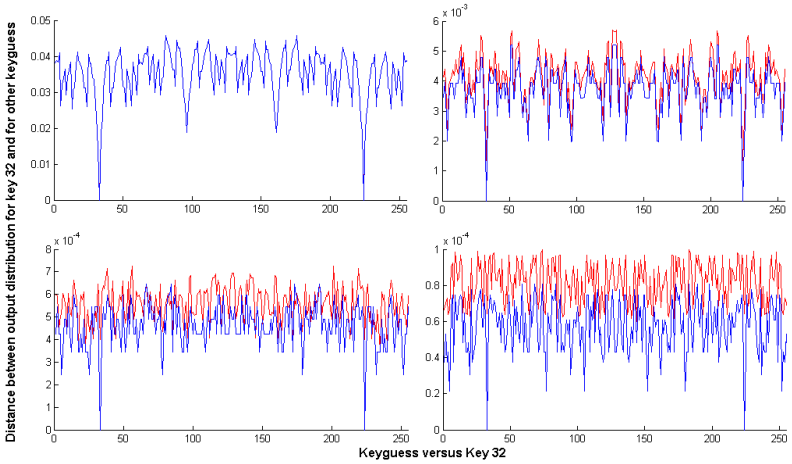

Fig. 3: Distance between the experimental (in gray) and theoretical (in black) distribution of a secret key byte and the theoretical distributions of the 256 possible key guesses. 2^{30} plaintexts were used for the experimental results.

In order to confirm the effectiveness of the proposed cryptanalysis in a key recovery context, we also computed the gain of the attack, as defined in [3]:

Gain the attack. *If an attack is used to recover an n-bit key and is expected to return the correct key after having checked on the average M candidates, then the gain of the attack, expressed in bits, is defined as:*

$$\gamma = -log_2 \frac{2 \cdot M - 1}{2^n} \tag{1}$$

Intuitively, the gain is a measure of the remaining workload (or number of key candidates to test) after a cryptanalysis has been performed. In the context of our attack, we can produce a list of key candidates sorted according to the distance between their theoretical distribution and the experimental distribution computed with the correct secret key. The gain is simply determined by the position of the secret key in this list. Figure 4 shows the gain of the attack for 1 to 14 rounds of PRESENT (still with a modified key-scheduling) in function of

the data complexity. This experiment used up to 2^{30} plaintext-ciphertext pairs. The gain is bounded by 8, as we guess 8 bits of key material. It increases with the number of texts and decreases with the number of rounds.

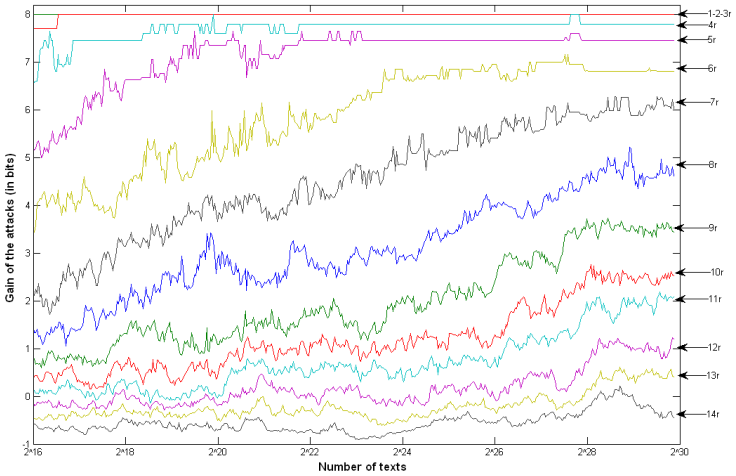

Fig. 4: Gain of the profiling attack against 1- to 14- round PRESENT.

Effect of the key-scheduling algorithm Unlike slide and related key attack, the proposed technique does not use a particular weakness in the key-scheduling. However, the number of subkey bits to guess at each round directly affects the time complexity of the attack, as one must compute the theoretical distribution for each of these key guesses. It may also increase the data complexity as distinguishing more keys generally requires more text pairs. The number of bits to guess according to the number of rounds is given in Table 1 for the complete key-scheduling algorithm. After 12 rounds, we have to guess 63 bits of the key, for a complexity equivalent to $2^{63} \times 2^{16} = 2^{79}$ encryptions. Consequently, this bounds the interest of the profiling attack to 12 rounds of the (simplified) cipher.

#rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
#bits	8	15	24	31	35	39	43	47	51	55	59	63	65	67	69	71	73	75	77	79	80

Table 1: Number of bits to guess according to the number of rounds in the attack.

2 A distinguishing attack

In order to get rid of the previous computational limits, we now present a variant of the attack. It has the advantage of not requiring the precomputation of theoretical distributions anymore. The distinguisher is based on the fact that the theoretical distribution at the output of the target trail (as computed with Algorithm 1) is significantly different from uniform, whatever subkey is used.

2.1 Principle of the attack

The attack is similar to the one presented above, yet it is simpler. We begin by generating a large number of plaintexts with 8 fixed bits. We encrypt those plaintexts using r-rounds PRESENT and record the distribution of the ciphertexts for the 16 bits at the output of the 4 active S-box in the last round. Given this experimental distribution, it is possible to compute the output distribution of the target 8-bit trail one round before by a classical partial decryption process. For one key guess, the evaluation of such an $r-1$-round distribution requires 2^{16} computations. Hence the total time complexity for all the key guesses equals $2^{16} * 2^{16} = 2^{32}$. Additionally using an FFT-based trick similar to the technique presented in [5], this complexity can be decreased to $16 \cdot 2^{16} \cdot 2^8 = 2^{28}$. For the correct key guess, the experimental 8-bit distribution in the penultimate round is expected to be more non-uniform than for any other guess. This is because decrypting with a wrong guess is expected to have the same effect as encrypting one more round. We can thus hope to distinguish the correct key from the wrong ones by computing the distance between a partially decrypted distribution and the uniform distribution. If the attack works properly, the distribution with the highest distance should correspond to the correct key.

2.2 Extensions of the attack

(ext. 1) Increase the fixed part in the plaintext. One can easily gain one round in the attack by simply fixing the 16 bits of plaintext corresponding to the 4 active input S-boxes of the trail. This way, the 8-bit trail in the second round is also fixed and the diffusion is postponed by one round. By fixing 32 bits out of 64 (corresponding to S-boxes 4-5-6-7-8-9-10-11), one can similarly extend the attack by 2 rounds. However, we are then limited in the generation of at most 2^{32} texts. This limitation may be mitigated with the following extension.

(ext. 2) Use multiple fixed plaintext values. The same analysis can be performed multiple times, using different values for the 8-bit (or 16- or 32-bit) fixed part of the plaintexts and then combining the results (*e.g.* taking the sum of the uniform *vs.* measured distances corresponding to the different fixed plaintexts). This allows exploiting more texts and moving to a known-plaintext context. The resulting attack is similar to multiple linear cryptanalysis: each fixed part of the plaintext can be seen as analogous to an additional approximation in [3, 12].

(ext. 3) Partial decryption of two rounds instead of one. In this case, 8 S-boxes are active in the last round instead of 4. Therefore, we have to keep a 32-bit distribution table in memory. Additionally, 38 bits of the key must be guessed for the partial decryption (32 bits for the last round + 16 bits for the penultimate round − 10 bits that are redundant). Using this trick, the adversary has to distinguish an $r-2$-round distribution for the correct key from an $r+2$-round distribution for the wrong candidates. The time complexity would be $(32 \cdot 2^{32} \cdot 2^{16}) \cdot (16 \cdot 2^{16} \cdot 2^8) = 2^{81}$ using again the results in [5]. Fortunately, the

2-round partial decryption process can be improved in the following way. Just observe that it can actually be divided in two independent partial decryptions, as illustrated in Figure 5. Using this trick allows reducing the time complexity down to: $2 \cdot (16 \cdot 2^{16} \cdot 2^8) \cdot (8 \cdot 2^8 \cdot 2^4) = 2^{44}$.

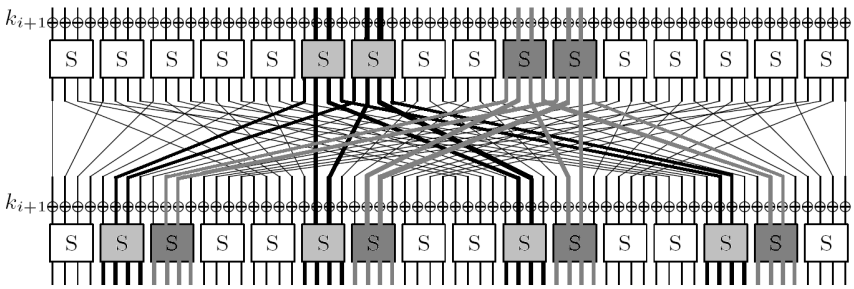

Fig. 5: Practical trails for 2-round partial decryption in PRESENT with reduced time complexity. The two independent trails are shown in different shades of gray.

2.3 Experimental results

We have run experiments against reduced-round versions of PRESENT with up to 15 rounds and evaluated the gain of the attack in different contexts. First, Figure 6 represents the mean result of 4 attacks using 2^{34} plaintexts where 16 input bits were fixed (*i.e.* using only **ext. 1**).

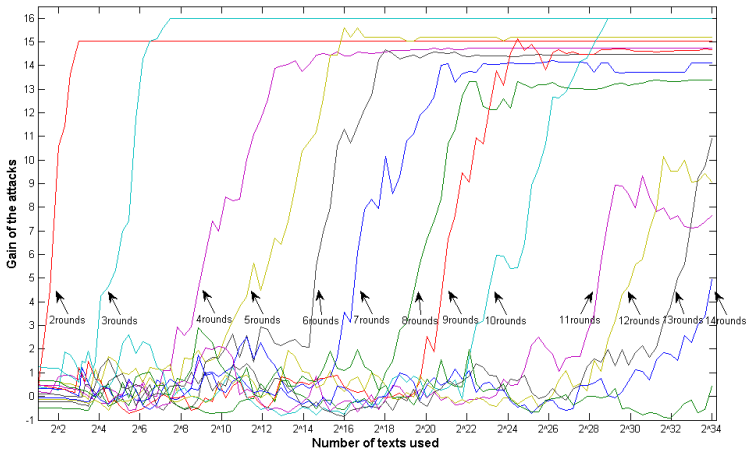

Fig. 6: Average gain of 4 attacks against 2 to 15-round PRESENT (**ext. 1**, 16 fixed bits).

To confirm the intuition that non-uniform distributions are observed for the correct key candidate, we represented the distance between the experimental distributions of the trail after partial decryption using a correct key and a uniform

distribution. Figure 7 illustrates that this distance decreases with the number of rounds and stabilizes after a sufficient number of plaintexts have been reached.

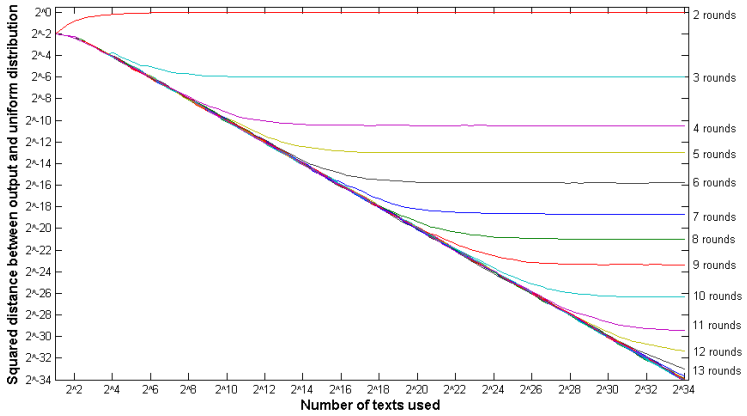

Fig. 7: Distance between uniform and output distributions after 1 to 15 rounds.

Figure 8 illustrates the results of a variant of the attack where 32 plaintext bits were fixed and consequently only 2^{32} texts were generated. As expected, the results are slightly better than in the previous experiment.

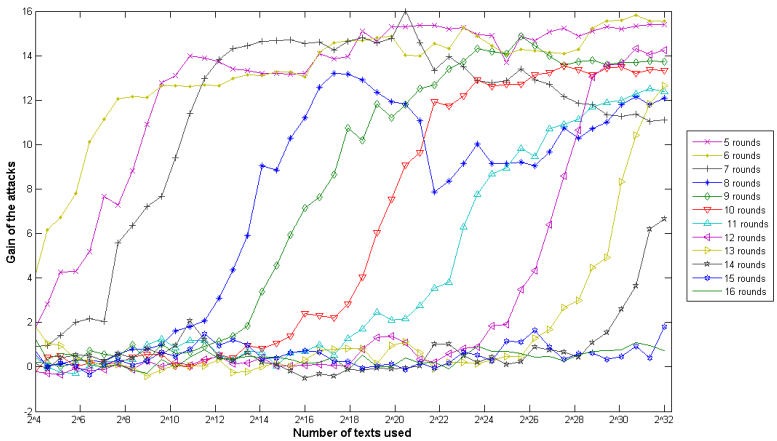

Fig. 8: Avg. gain of 12 attacks against 5 to 16-round PRESENT (**ext. 1**, 32 fixed bits).

Figure 9 finally shows an application of **ext. 2**. The graph represents the evolution of the attack gain against 1 to 32-rounds after 2^{32} plaintexts. The top and bottom curves represent the maximum and minimum gains among 12 experiments, while the two other curves represent respectively the average gain and the gain of the attack combining the 12 experiments. We clearly observe that

combining the distances corresponding to the 12 experiments and computing a list of key candidates afterwards gives rise to much better result than computing a list for each experiment and then taking the average position for each key guess. Using the first method, we reach a significant gain up to 15 rounds.

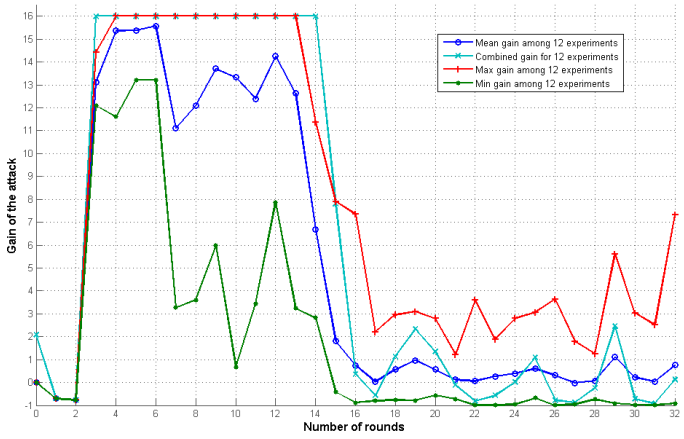

Fig. 9: Gain of attacks against 1- to 32-rounds PRESENT with $12 \cdot 2^{32} = 2^{35.6}$ texts.

For discussion purpose, Figure 14 in appendix C represents the gain of a linear cryptanalysis against 6- to 16-rounds PRESENT. The attack is based upon an iterative approximation involving one S-box with bias 2^{-2} in the first round and one S-box with bias 2^{-3} in each other round. It can recover up to 12 bits of the last subkey. Note that this example is not given for comparing the efficiencies of different attacks but to illustrate the big difference between security bounds as provided for the "best possible" linear attacks in [4] and attacks based on approximations that can be found in practice. The one that we proposed in appendix is obviously not optimal, but it exploits the same iterativeness as our statistical saturation attack. As a matter of fact, the attacks we discuss in this paper are experimented and therefore cannot be straightforwardly compared with bounds. They more directly relate to actual attacks such as presented in [17].

Given the intensive computations required by the previous attacks and our limited computational resources, we could only launch a reduced set of experiments for each attack. In conjunction with the statistical nature of these attacks, this can account for some of the experimental variability appearing in the previous graphs. In particular, this can explain some counter-intuitive observations, like a slight decrease in gain when more plaintexts are exploited.

3 Theoretical complexity

Intuitively, the efficiency of our distinguisher depends on the extent to which an experimental distribution after r-rounds PRESENT can be distinguished from a uniform distribution. Therefore, it can be nicely related to the theoretical analysis of Baignères *et al.* in [2] which shows that the data complexity required to distinguish two distributions is proportional to the inverse of the squared Euclidean distance between these distributions. Using Algorithm 1, we can easily compute a theoretical approximation of this Euclidean distance for PRESENT. It directly gives rise to Figure 10 in which the complexity of distinguishing the theoretical distributions at the output of PRESENT from uniform distributions is given for 1 to 16 rounds. We also illustrate the complexity of a linear cryptanalysis using the same approximation as the attack in Appendix C. Again, the difference between the effectiveness of an actual linear attack as in this figure and the security bounds in [4] should be emphasized. But even these security bounds (*e.g.* 2^{84} plaintext-ciphertext pairs to break 28-rounds PRESENT) suggest that our attack is an improvement compared to the theoretical expectations.

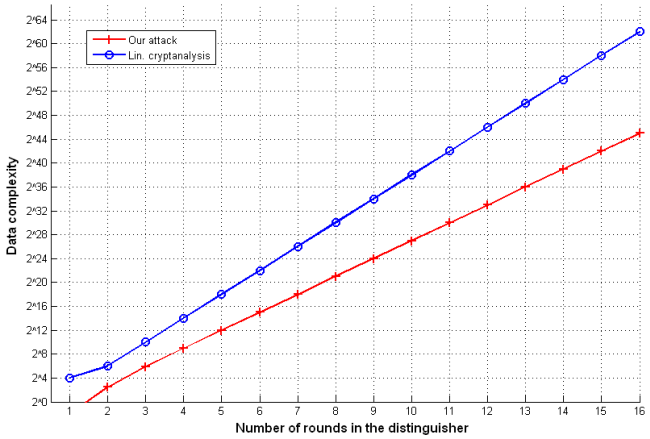

Fig. 10: Theoretical data complexity of the attack against PRESENT.

More interesting are the results in Table 2 that summarize the complexity of the attacks against PRESENT known so far (*i.e.* mainly [17] and the results in this paper). Note that due to the iterative nature of our trail, the time and memory complexities do not vary with the number of rounds in the trail. They only depend on the number of rounds that are partially decrypted. Most importantly, the provided data complexities are based upon the theoretical values given by the graph in Figure 10 and rely on the following assumptions:

– All attacks use **ext. 1** with 32 plaintext bits fixed.
– In 1-round (*resp.* 2-round as suggested in **ext. 3**) decryption attacks, we use a $r-3$-round (*resp.* $r-4$-round) distinguisher and have the time and memory complexities discussed in Sections 2.1 and 2.2.
– When the number of plaintexts needed to perform the attack exceeds 2^{32}, we use **ext. 2**. By combining multiple fixed plaintext values, we consider an attack exploiting distributions of larger dimensions, similarly to the multiple linear cryptanalysis in [3]. But it is an open problem to determine exactly the effect of this extension to the attack complexities. At least, our experiments suggest that these estimations are valid up to 16 rounds.

Note that our attack only recovers 16 key bits while [17] recovers the whole key. But as mentioned in Section 1.1, similar trails could be used to recover 32 more key bits with similar complexities. Hence, our results (including the part confirmed experimentally) anyway improve the best reported attack considerably.

$\#rounds$	type of attack	data compl.	time compl.	memory compl.	gain	reference
16	Diff. Crypt.	2^{64}	$2^{65}MA$	$6*2^{32}bits$	≤ 32	[17]
8	our attack*	$c*2^{12}$	2^{28} op.	2^{16} counters	≤ 16	this paper
	our attack**	$c*2^{9}$	2^{44} op.	2^{32} counters	≤ 38	this paper
12	our attack*	$c*2^{24}$	2^{28} op.	2^{16} counters	≤ 16	this paper
	our attack**	$c*2^{21}$	2^{44} op.	2^{32} counters	≤ 38	this paper
16	our attack*	$c*2^{36}$	2^{28} op.	2^{16} counters	≤ 16	this paper
	our attack**	$c*2^{33}$	2^{44} op.	2^{32} counters	≤ 38	this paper
20	*our attack**	$c*2^{48}$	2^{28} op.	2^{16} counters	≤ 16	this paper
	*our attack***	$c*2^{45}$	2^{44} op.	2^{32} counters	≤ 38	this paper
24	*our attack**	$c*2^{60}$	2^{28} op.	2^{16} counters	≤ 16	this paper
	*our attack***	$c*2^{57}$	2^{44} op.	2^{32} counters	≤ 38	this paper

* 1-round decryption, ** 2-round decryption

Table 2: Summary of attacks (italic are not experimented and use **ext. 2**).

4 Countermeasures and influence of the S-box

The origin of the proposed statistical saturation attack against PRESENT mainly lies in a weakness of the diffusion layer. A straightforward countermeasure would be to modify the permutation in order to avoid poor diffusion in any subset of S-boxes. But the proposed attack relates to the overall diffusion properties of the cipher. Hence the S-boxes also have an impact with this respect that we shortly study in this section. To do so, we generated 5000 different S-boxes respecting the four conditions imposed in the generation of the PRESENT S-box (see [4], Section 4.3). According to the authors, these constraints ensure that PRESENT is resistant to differential and linear attacks. Figure 15 in Appendix D represents the evolution of the squared distance between the uniform and output distribution of the cipher according to the number of rounds. Each curve represent a different choice for the S-box used in the cipher. It is noticeable that the PRESENT S-box is among the worst possible choices to resist our attack.

To confirm this impact of a weak *vs.* strong S-box in our cryptanalysis, we finally ran new experiments against a tweaked PRESENT where the original S-boxes were replaced by the dashed S-boxes of Figure 15 (*i.e.* those corresponding to the best and worst diffusion properties among our 5000 generated S-boxes). Figure 11 gives the gain of these attacks for different number of rounds (each attack used 2^{30} chosen plaintexts). As expected, the attack against the weak version of the cipher gives the best results. The figure emphasizes that the proposed attack is not directly related to linear or differential cryptanalysis (*i.e.* it is possible to find a cipher that is immune against linear and differential cryptanalysis, but not against the proposed statistical saturation attack).

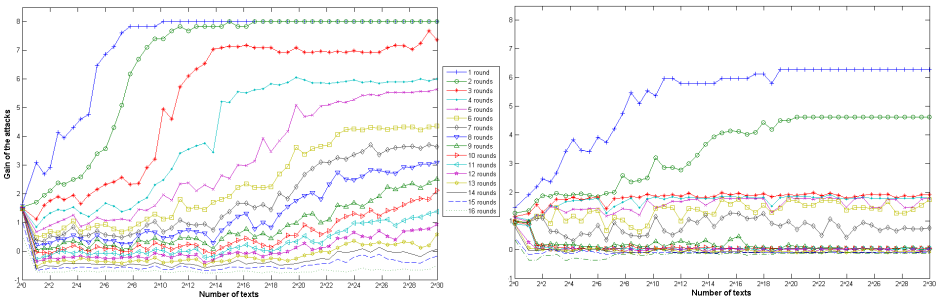

Fig. 11: Comparison between weak (left) and strong (right) S-boxes.

5 Conclusion and further works

In this paper, we presented a new attack against the block cipher PRESENT that improves previously known cryptanalyses against this cipher. Experimentally, it allows us to break 15 rounds with $2^{35.6}$ plaintext-ciphertext pairs. We also present theoretical estimations of attacks than can break more cipher rounds. Additionally, we show that the proposed cryptanalysis is not directly related to linear and differential attacks. In practice, the security of the full cipher does not seem to be compromised by our results although the proposed attack was not discussed in the algorithm specifications. However, it confirms and emphasizes that PRESENT has been designed with little security margins.

Determining if the proposed statistical saturation attacks can improve cryptanalytic results against other ciphers is an interesting open question. Since they only exploit very general principles (namely, uncomplete diffusion after some cipher rounds), they are likely to be applicable to other reduced algorithms. But on the other hand, the proposed attack was particularly efficient against PRESENT due to a weakness in its permutation. Hence, it is not clear if the proposed technique can be as effective against other ciphers, with better diffusion properties. More theoretically, a better theoretical analysis of the attack, in particular the analogy between multiple linear cryptanalysis and the use of multiple fixed plaintext bytes in our context is also worth further investigation. Eventually, it would be interesting to investigate if the multidimensional cryptanalysis presented in [10] could be used to improve our results.

Another research direction would be to use the trail of Figure 12 in which 27 bits out of 36 are redirected to only 9 S-boxes at each round. It means that the lack of diffusion could be worse than in the trail of Figure 1 (we found 4 trails of this kind). However, this weakness is compensated by a larger trail size (36 bits instead of 16) which increases the diffusion inside the trail. Applying the attack presented in this paper to this new trail is also more difficult to experiment because of the 36-bit distributions for which a 1-round decryption would have to be performed. Hence, the efficiency of this attack is an open question.

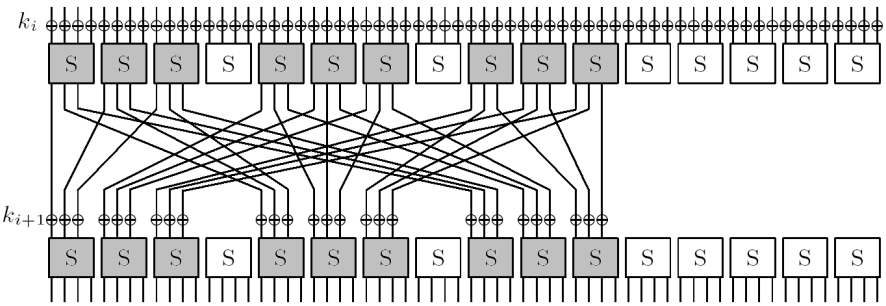

Fig. 12: Another poor diffusion trail in the permutation layer of PRESENT.

References

1. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, in the proceedings of the First Advanced Encryption Standard (AES) Conference, Ventura, CA, August 1998.
2. T. Baignères, P. Junod, S. Vaudenay, *How Far Can We Go Beyond Linear Cryptanalysis?*, in the proceedings of ASIACRYPT 2004, Lecture Notes in Computer Science, vol 3329, pp 432-450, Jeju Island, Korea, December 2004.
3. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of CRYPTO 2004, Lecture Notes in Computer Science, vol 3152, pp 1-22, Santa Barbara, California, USA, August 2004.
4. A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, C. Vikkelsoe, *PRESENT: An Ultra-Lightweight Block Cipher*, in the proceedings of CHES 2007, Lecture Notes in Computer Science, vol 4727, pp 450-466, Vienna, Austria, September 2007.
5. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improving the Time Complexity of Matsui's Linear Cryptanalysis*, in the proceedings of The International Conference on Information Security and Cryptology - ICISC 2007, Lecture Notes in Computer Science, vol 4817, pp 77-88, Seoul, Korea, November 2007.
6. J. Daemen, L.R. Knudsen, V. Rijmen, *The Block Cipher Square*, in the proceedings of Fast Software Encryption 1997, Lecture Notes in Computer Science, vol 1267, pp 149-165, Haifa, Israel, January 1997.
7. H. Gilbert, H. Handschuh, A. Joux, S. Vaudenay, *A Statistical Attack on RC6*, in the proceedings of Fast Software Encryption, Lecture Notes in Computer Science, vol 1978, pp 64-74, Yokohama, Japan, April 2001.
8. C. Harpes, G. Kramer, J. Massey, *A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-Up Lemma*, in the proceedings of EUROCRYPT 1995, LNCS, vol 921 , pp 24-38, Saint-Malo, France, May 1995.
9. C. Harpes, J. Massey, *Partitioning Cryptanalysis*, in the proceedings of Fast Software Encryption 1997, LNCS, vol 1267, pp 13-27, Haifa, Israel, January 1997.
10. M. Hermelin, J.Y. Cho, K. Nyberg, *Multidimensional Linear Cryptanalysis of Reduced Round Serpent*, in the proceedings of ACISP 2008, Lecture Notes in Computer Science, vol 5107, pp 203-215, Wollongong, Australia, July 2008.
11. K. Hwang, W Lee, S. Lee, S. Lee, J. Lim, *Saturation Attacks on Reduced Round Skipjack*, in the proceedings of Fast Software Encryption 2002, Lecture Notes in Computer Science, vol 2365, pp 100-111, Leuven, Belgium, February 2002.
12. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Sciences, vol 839, pp 26-39, Santa Barbara, California, USA, August 1994.
13. L.R. Knudsen, D. Wagner, *Integral cryptanalysis*, in the proceedings of Fast Software Encryption 2002, Lecture Notes in Computer Science, vol 2365, pp 112-127, Leuven, Belgium, February 2002.
14. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of EUROCRYPT 1993, LNCS, vol 765, pp 386-397, Lofthus, Norway, May 1993.
15. M. Minier, H. Gilbert, *Stochastic Cryptanalysis of Crypton*, in the proceedings of Fast Software Encryption 2000, Lecture Notes in Computer Science, vol 1978, pp 121-133, New York, USA, April 2000.
16. S. Vaudenay, *An experiment on DES - Statistical Cryptanalysis*, in the third ACM Conference on Computer Security, New Dehli, India, pp 139-147, March 1996.
17. M. Wang, *Differential Cryptanalysis of Reduced-Round PRESENT*, in the proceedings of AFRICACRYPT 2008, Lecture Notes in Computer Science, vol 5023, pp 40-49, Casablanca, Morocco, June 2008.

A The block cipher PRESENT

PRESENT is a Substitution-Permutation Network with a block size of 64 bits. The recommended key size is 80 bits, which should be sufficient for the expected applications of the cipher. However a 128-bit key-schedule is also proposed. The encryption is composed of 31 rounds. Each of the 31 rounds consists of a XOR operation to introduce a round key K_i for $1 \leq i \leq 32$, where K_{32} is used for post-whitening, a linear bitwise permutation and a non-linear substitution layer. The non-linear layer uses a single 4-bit S-box S which is applied 16 times in parallel in each round. The cipher is described in pseudo-code in Figure 1.

generateRoundKeys()
for $i = 1$ to 31 **do**
  addRoundKey(STATE,K_i)
  sBoxLayer(STATE)
  pLayer(STATE)
end for
addRoundKey(STATE,K_{32})

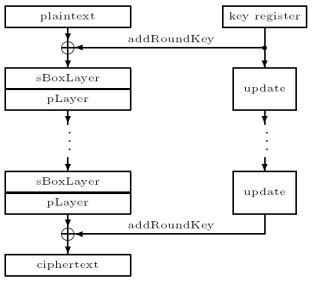

Fig. 13: Top-level algorithmic description of PRESENT according to [4].

The linear permutation is defined by Table 3 where bit i of input is moved to bit position $P(i)$. The 4-bit S-box is defined according to the table 4. The 4-bit nibble i at the input of an S-box is substituted by the 4-bit $\mathsf{S}[i]$ in output.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	48	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Table 3: Permutation layer for PRESENT.

i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[i]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 4: S-box table for PRESENT (hexadecimal notation).

We don't mention the key-schedule here as we don't make explicit use of it in our attack. We refer to the original paper [4] for the details of the specifications.

B Theoretical evaluation of the target trail distribution

Algorithme 1

```
1  input: a 8-bit subkey guess sk and the 8-bit input distribution distrib_in[256]
2  output: the 8-bit output distribution distrib_out[256]
3
4  initialize distrib_out[256] to the all-zero state
5  for each 8-bit values text do
6     for each 8-bit values rand do
7        fix the 8-bit trail to text and xor with sk
8        fix the 8-bit non trail to rand
9        apply the sboxes
10       apply the permutation
11       evaluate the value of the 8 bit trail out
12       update distrib_out[out]= distrib_out[out]+ distrib_in[text]/256;
13    end for
14 end for
```

C Linear cryptanalysis using a single approximation

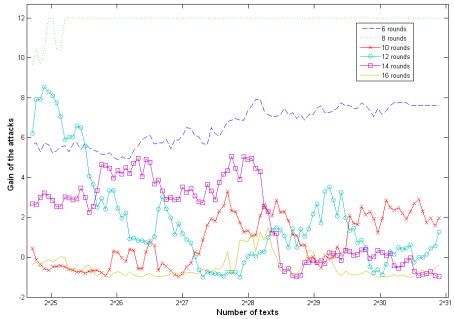

Fig. 14: Gain of a linear cryptanalysis against 6- to 16-rounds PRESENT.

D Influence of the S-box on the attack effectiveness

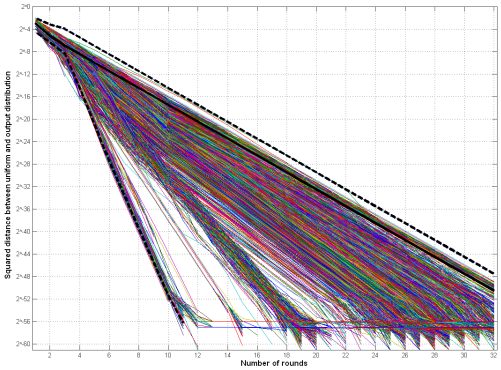

Fig. 15: Evolution of the squared distance between uniform and output distribution for 5000 different S-boxes (the PRESENT S-box is in plain black).

7. MULTI-TRAIL STATISTICAL SATURATION ATTACKS

Multi-Trail Statistical Saturation Attacks

B. Collard*, F.-X. Standaert**

UCL Crypto Group, Microelectronics Laboratory, Université catholique de Louvain.
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium.

baudoin.collard; fstandae@uclouvain.be

Abstract. Statistical Saturation Attacks have been introduced and applied to the block cipher PRESENT at CT-RSA 2009. In this paper, we consider their natural extensions. First, we investigate the existence of better trails than the one used in the former attack. For this purpose, we provide a theoretical evaluation of the trail distributions using probability transition matrices. Since the exhaustive evaluation of all possible distributions turned out to be computationally hard, we additionally provide a heuristic branch-and-bound algorithm that allows us to generate a large number of good trails. These tools confirm that the trail of CT-RSA 2009 was among the best possible ones, but also suggest that numerous other trails have similar properties. As a consequence, we investigate the use of multiple trails and show that it allows significant improvements of the previous cryptanalysis attempts against PRESENT. Estimated complexities indicate that PRESENT-80 is safe against key recovery, by a small security margin. We also discuss the impact of multiple trails for the security of the full PRESENT-128. We finally put forward a "statistical hull" effect that makes the precise theoretical analysis of our results difficult, when the number of block cipher rounds increases.

Introduction

PRESENT is a block cipher presented at CHES 2007, that was designed for small embedded applications [3]. It has a Substitution Permutation Network architecture, with a 64-bit block size and 31 rounds. The same 4-bit S-box is applied 16 times in parallel in each round. The designers have proposed two key sizes: 80 and 128 bits. Due to its simple and elegant structure, it has been the focus of different cryptanalysis attempts. In [23], the author presented a first attack against PRESENT, using differential cryptanalysis. It applies to 16 block cipher rounds and requires the whole codebook and a time complexity of 2^{65}. In 2009, Ozen et al. proposed a related key rectangle attack against up to 17 rounds of PRESENT, with a time complexity of 2^{104} and 2^{63} chosen plaintexts [20]. Two papers exploit the linear cryptanalysis and target 26 rounds, respectively by taking advantage of the linear hull effect [17] and multiple approximations [10]. These attacks require the whole codebook. Finally, [19] combines linear cryptanalysis and weak keys and targets up to 28 rounds in this context.

* Work supported by the project Nanotic-Cosmos of the Walloon Region.
** Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

In this paper, we pay a particular attention to a Statistical Saturation Attack that was specifically devised for the cryptanalysis of PRESENT (although it could apply to other ciphers). It exploits the weak diffusion of certain bits (called the trail) during the encryption process, when some of the plaintext bits are fixed. This property did lead to an estimated attack against up to 24 rounds, using approximately 2^{60} chosen plaintexts. As detailed in [6], the main limitation when trying to extend this technique towards more rounds is the data complexity that exceeds the complete codebook. We consequently investigated the tracks that could be used to get rid of this limitation. Our contributions are threefold.

First, we provide tools allowing one to approximate the diffusion in a trail, using Markov chains. We exhibit that, besides the iterative trail proposed in [6], there exists many other trails with a similarly weak diffusion. We also propose a heuristic branch-and-bound algorithm in order to generate them efficiently. We finally confirm our theoretical analysis with experiments that can break up to 16 rounds PRESENT. Then, in a second part of the paper, we investigate the exploitation of these multiple trails. We show that they can be used to trade data complexity for time complexity. As a result, we discuss the possibility to mount a key-recovery attack against the full 31-round PRESENT-128, using the complete codebook, and a time complexity below 2^{128} memory accesses. We put forward that such an attack could be possible under certain (optimistic) conditions of independence for the trails - the exact evaluation of these conditions being an important scope for further research. We note that such an attack would anyway be of theoretical interest only. In particular, the authors in [3] clearly suggest PRESENT-80 for their target applications (rather than PRESENT-128). However, these results question the number of rounds for PRESENT-128 and highlights that they could have been increased over those for PRESENT-80. Finally, in a third part of the paper, we put forward a "statistical hull effect", *i.e.* a counterpart of the linear hull effect in linear cryptanalysis. We discuss its impact for the theoretical analysis of our estimated attack complexities.

1 The Statistical Saturation Attack

1.1 Principle of the attack

The Statistical Saturation Attack, originally described in [6], takes advantage of a weakness in the diffusion layer of PRESENT. For the S-boxes 5, 6, 9 and 10 (called the *active* S-boxes), only 8 out of 16 input bits are directed to other S-boxes. Figure 1 illustrates this observation (note that there exists many other examples of weak diffusion in the permutation). Consequently, if we fix the 16 bits at the input of the active S-boxes, then 8 bits will be known at the very same input for the next round. We can iteratively repeat this process round by round and observe a non-uniform behavior at the output of the active S-boxes.

Thanks to this non-uniform behavior, 16 bits of the last subkey can be recovered as follows. We first generate a large number of plaintexts with 8 fixed bits. The plaintexts are encrypted using r-rounds PRESENT and the distribution of

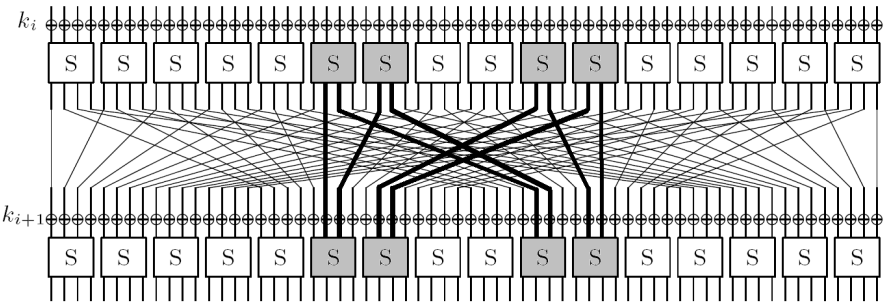

Fig. 1: Permutation layer of PRESENT: bold lines show the weak diffusion property.

the ciphertexts are recorded for the 16 bits at the output of the 4 active S-boxes in the last round. Given this experimental distribution, it is possible to compute the output distribution of the target 8-bit trail one round before, using a partial decryption process. For one key guess, the evaluation of such an $r-1$-round distribution requires 2^{16} computations. Hence the total time complexity for all the key guesses equals $2^{16} * 2^{16} = 2^{32}$. Additionally using an FFT-based trick similar to the technique presented in [4], this complexity can be decreased to $16 \cdot 2^{16} \cdot 2^8$. For the correct key guess, the experimental 8-bit distribution in the penultimate round is expected to be more non-uniform than for any other guess. This is because decrypting with a wrong guess is expected to have the same effect as encrypting one more round. We can thus hope to distinguish the correct key from the wrong ones by computing the distance between a partially decrypted distribution and the uniform distribution. If the attack works properly, the distribution with the highest distance should correspond to the correct key.

1.2 Extensions of the attack

In [6], the authors propose 3 extensions to improve the cryptanalysis:

(ext. 1) Increase the fixed part in the plaintexts. One can easily gain one round in the attack by simply fixing the 16 bits of plaintext corresponding to the 4 active input S-boxes of the trail. This way, the 8-bit trail in the second round is also fixed and the diffusion is postponed by one round. By fixing 32 bits out of 64 (corresponding to S-boxes 4-5-6-7-8-9-10-11), one can similarly extend the attack by 2 rounds. However, we are then limited in the generation of at most 2^{32} texts. This limitation may be mitigated with the following extension.

(ext. 2) Use multiple fixed plaintext values. The same analysis can be performed multiple times, using different values for the 8-bit (or 16- or 32-bit) fixed part of the plaintexts and then combining the results (*e.g.* taking the sum of the uniform *vs.* measured distances corresponding to the different fixed plaintexts). This allows exploiting more texts and moving to a known-plaintext context. The resulting attack is similar to multiple linear cryptanalysis: each fixed part of the plaintext can be seen as analogous to an additional approximation in [2, 11].

(ext. 3) Partial decryption of two rounds instead of one. In this case, 8 S-boxes are active in the last round and 4 S-boxes are active in the penultimate round. As detailed in [7] and illustrated in Figure 4, one can perform two independent partial decryptions in parallel, in order to decrease the time complexity of the attack down to $2 \cdot (16 \cdot 2^{16} \cdot 2^8) \cdot (8 \cdot 2^8 \cdot 2^4) = 2^{44}$ elementary operations.

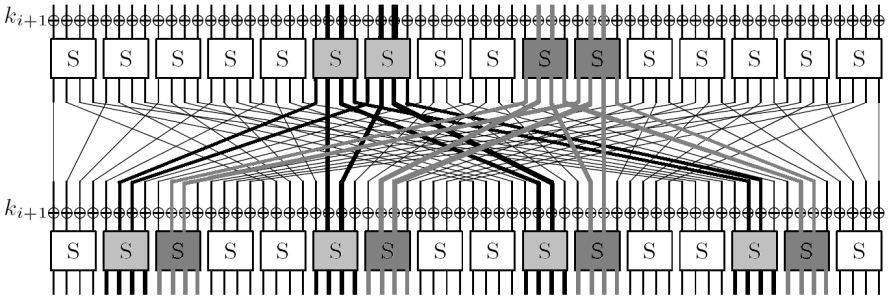

Fig. 2: Practical trails for 2-round partial decryption in PRESENT with reduced time complexity. The two independent trails are shown in different shades of gray.

2 Evaluating the trail distributions with Markov chains

As a matter of fact, the previous attack essentially exploits the property that it is possible to evaluate the distribution of a subset of output bits given the distribution of a subset of input bits for one round of PRESENT. Minier and Gilbert use a similar technique in their attack against Crypton [16]. In [6], the authors exploited an iterative trail with 8 active bits in 4 active S-boxes in each round. However, as already mentioned, this is not the only possible trail and an interesting problem is to determine if there are other trails leading to better attacks. In this section, we show how to evaluate the distribution of a trail going through several rounds of PRESENT. For this purpose, we characterize such a trail by a matrix containing the transition probabilities between the inputs and outputs. Additionally, we rely on the assumption that the bits that are not part of the trail are uniformly distributed. This assumption as well as the Markov chains that we exploit in the following were also used by Vaudenay in his paper on χ^2 cryptanalysis [21]. As will be discussed in Section 5, this is becoming incorrect as the number of rounds in the trail increases. But as the next section will show, this assumption is required in order to limit the computational cost of our estimations to tractable values, when comparing different trails.

2.1 Transition matrix for an S-box

Let us consider an active S-box with size $n*n$, and suppose that the trail includes i active bits among n in input and j active bits in output. Consequently, there are 2^i possible inputs and 2^j possible outputs, and the size of the transition matrix is $2^i * 2^j$. This matrix is constructed in the following way:

- Initialize a matrix of size $2^i * 2^j$ and fill it with zeros.
- For every possible S-box input value, extract a masked i-bit input and the j-bit output, and increment the matrix in the corresponding position.
- Multiply the matrix by $2^i/2^j$ for normalization.

By construction, any transition matrix has the properties that the sum over any row is equal to one. For example, the iterative trail represented in Figure 1 uses the same transition matrix for each active S-box:

$$A = \begin{pmatrix} 0 & 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0 & 0.5 \\ 0.5 & 0 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 & 0 \end{pmatrix}$$

In this case, the matrix is square, but it is not mandatory as it depends on the number of active input/output bits. The interpretation of the transition matrix is easy: each row represents a possible value for the input and the column represents the probability of transition to a particular output value given the input.

2.2 Transition matrix for the permutation layer

For a permutation layer like the one used in PRESENT, the number of active bits in output is equal to the number k of active bits in input. Consequently, the matrix is square with size $2^k * 2^k$. Moreover, at each input value in the trail corresponds one and only one output value and thus the transition matrix contains only zeros and ones (i.e. it is a particular instance of permutation matrix).

2.3 Transition matrix for the subkey addition

The effect of a XOR between the input bits in the trail and unknown key bits is similar to a permutation. To each value in the trail before the key addition corresponds only one output value after the key addition. Hence, the transition matrix for the subkey addition is also a permutation matrix. However, unlike the transition matrix for the permutation layer, this transition matrix does not increase the diffusion in the trail. Intuitively, this is because the key addition does not mix the active bits coming from different active S-boxes as with the permutation layer. Mathematically, this corresponds to the property that the transition matrix can be decomposed into a Kronecker product of small submatrices. Consequently, given the assumption of uniform distribution for the bits that are not part of the trail, different subkeys have different output distributions in the trail, but they present an identical non-uniform behavior. Hence, it is sound to compute the distribution of a trail independently of the keys.

2.4 Composition of transition matrices

If several S-boxes are active in parallel in a trail, the overall transition matrix is given by the Kronecker product of the transition matrix related to each S-box. The transition matrix of a round can then be computed as the matrix product

of the transition matrices for the S-box and permutation layers. Thereafter, given the transition matrix for a complete trail, the output distribution can be directly evaluated as the vector-matrix product of the input distribution and the transition matrix. The main drawback of this method is that it requires to compute a matrix product with matrices of size $2^n * 2^n$ where n is the number of active bits involved at any point in the trail during the encryption process.

2.5 Practical example

We illustrate this technique using the iterative trail of Figure 1. This trail is composed of 4 active S-boxes at each round, with 2 bits out of 4 active in each S-box. As there are 8 active bits at each round, the full transition matrix has a size of $2^8 * 2^8$. It is computed in the following way: The matrix transition for the 4 parallel S-boxes is computed as: $A^4 = A \otimes A \otimes A \otimes A$, where $\otimes$ is the symbol for the Kronecker product. The matrix for one full round is then given by $R = A^4 \cdot P$ (where P is the transition matrix for the permution on 8 bits). The transition matrix after n rounds is then $R^n = \underbrace{R \cdot R.... \cdot R}_{n\ times}$. Given a vector d_{in} of size $(1 * 2^8)$ describing the distribution of the 8-bits active bits in input, the distribution of the output active bits is finally given by $d_{out} = d_{in} \cdot R^n$.

3 Heuristic branch-and-bound for trail search

As detailed in the previous section, the evaluation of the distribution for a single trail can be computationally intensive if this trail involves a lot of active bits. This was not an issue for the attack in [6], but may become the limiting factor for other trails (or other ciphers). In order to mitigate this limitation, we now present a heuristic algorithm based on the branch-and-bound proposed by Matsui for linear approximation search in [15]. The goal of this heuristic is to perform a pre-selection of "interesting trails" that minimizes the number of active S-boxes (so that the treatment of the previous section can still be applied) while trying to limit the diffusion based on a simpler criteria than a probabilistic distance.

3.1 Description of the algorithm

The basic principle of the heuristic is to maximize the ratio between the number of active bits and the number of active S-boxes in a trail. While this criteria does not ensure finding the best trail distributions, it is extremely fast to evaluate and to integrate into the execution of a branch-and-bound algorithm. As will be shown later in the section, it also provides reasonably good results. The justification is the following: even though all the trails with low diffusion are not necessarily good trails (because of the influence of the transition matrices for the S-boxes), all the good trails must have low diffusion in their permutations layers. Consequently, a good strategy is to first generate a large number of trail candidates with a branch-and-bound heuristic, then to compute the full transition matrices for each of these candidates and to select the best ones only.

In order to speed-up the execution of the algorithm, we used a similar implementation technique as presented in [4]. That is, we start by exhaustively counting the couples (input, output) of the permutation for which the number of active S-boxes is low. Such couples are called *permutation candidates* and are entirely defined by the number of active input and output S-boxes, the position of these S-boxes and their corresponding mask value. The permutation candidates are then stored in a database (a hash table) instead of being generated on-the-fly during each branching phase. All the candidates having the same active output S-boxes are stored in the same list. Once the database is created, we launch the actual trail search: a trail on r rounds can be obtained by the concatenation of r permutation candidates, if the positions of the active S-boxes at the exit of a permutation candidate correspond to those of the active S-boxes at the input of the next candidate. These constraints are easily checked, as the candidates are picked up in the database according to the position of their active output S-boxes. The objective function that we need to maximize is the average ratio between the number of active bits and the number of active S-boxes in the trail. Note finally that we pile up the candidates starting with the last round, then going down gradually until the first round, in order to benefit from the knowledge of the best ratio in each phase of the branch-and-bound.

3.2 Results

As an illustration, we generated 1000 trails with maximum 5 active S-boxes in each round and computed the theoretical data complexity as in [6], for the distinguishers based on each of these trails. That is, following the analysis of Baignère et al. [1], we estimate the data complexity as proportional to the inverse of the Euclidean distance between the distributions evaluated in Section 2 and a uniform distribution. The results in the figure show that after 15 rounds, the data complexity varies between 2^{50} and 2^{66}, according to the trail. The original

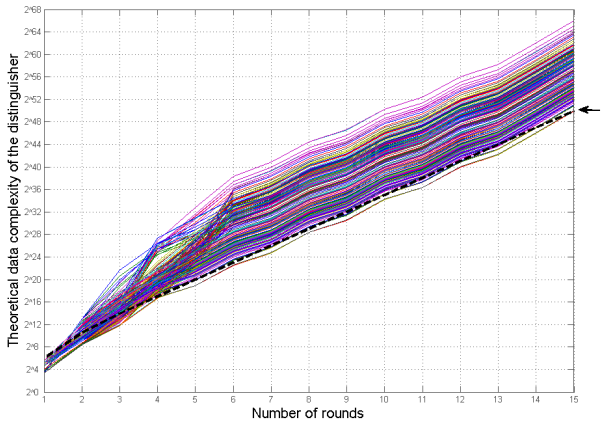

Fig. 3: Theoretical data complexity for distinguishers based on 1000 different trails.

trail of Figure 1 is marked with an arrow and is among the best ones (see Figure 3). Note that by increasing the number of trails generated by the branch-and-bound (beyond 1000), we can easily produce very large amounts of trails with good theoretical data complexities. As will be detailed in Section 5, the amount of such trails increases exponentially with the number of rounds.

3.3 Experimental validation of the estimated data complexity

The estimations in the previous section indicate that the data complexity increases by approximately 2^3 for every additional round. As these estimations rely on the assumptions needed to evaluate the distributions in Section 2, we confirmed these predictions experimentally, in order to verify that our assumptions hold to a sufficient extent. For this purpose, we complemented the experiments in [6] and attacked up to 15 rounds PRESENT with 2^{32} texts, using a 2-round partial decryption process. Figure 4 illustrates that we gain one round compared to the original attack of CT-RSA, and confirms the theoretical expectations. Note that the two-round decryption also allows a significantly increased gain (because there are more key bits guessed in the experiment).

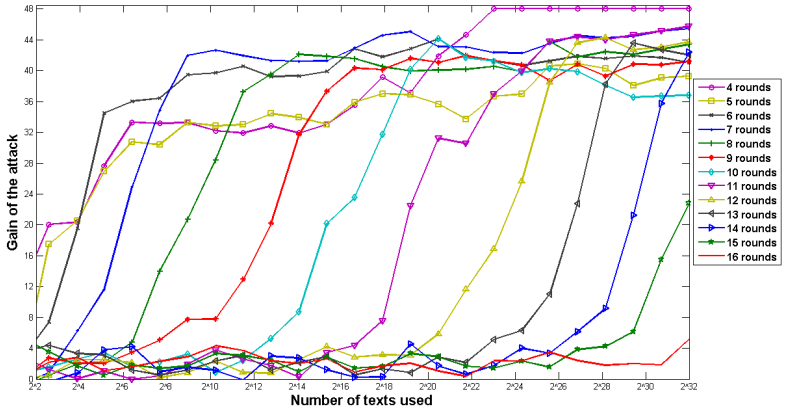

Fig. 4: Average gain of 6 attacks against 4 to 16 rounds PRESENT, using up to 2^{32} texts (these attacks exploit **ext. 1**, with 32 fixed bits and **ext. 3**).

4 Multiple trails

The previous section shows that the simple trail of Figure 1 is among the best ones to perform a Statistical Saturation Attack against PRESENT. On the other hand, we also observe that a large number of trails perform similarly good in theory. Hence, a natural idea is to investigate the use of multiple trails, as can be done in linear cryptanalysis with multiple approximations [2]. In the following, we consequently consider two questions. First, we study the possibility to exploit several trails with different input masks and the same output mask, in order to

increase the gain of the attack. Our experiments suggest that this technique yields good results and allows improving the best-reported cryptanalysis against PRESENT. Then, in Section 5, we show that there exists many different trails with the same input and output masks, the combination of which affects the distribution of the output in a hardly predictable way. We discuss the impact of such a "statistical hull" effect on the assumptions of Section 2.

4.1 (ext. 4) Multiple trails cryptanalysis

In the Statistical Saturation Attack of [6], the main limitation of the attack was the number of texts required to find the correct subkey. Above 24 rounds (and assuming that **ext. 2** yields the expected improvements), the data complexity of the attack reaches the codebook size of PRESENT. In this section, we consequently investigate the possibility to use several distinct trails in order to partially remove this limitation. Thanks to our branch-and-bound, we were able to generate many trails with different input masks and the same output mask. It allowed us to run several independent attacks in parallel, each one using a different trail and thus different partitioning of the plaintexts. As the output mask is the same, we can combine the results of the attacks together because the partial decryption involves the same subkey bits. In practice, each single-trail attack produces a vector containing the distances between the uniform and output distributions after partial decryption with the keyguess. A straightforward combination that was used in the context of linear cryptanalysis using multiple approximations (and for **ext. 2**) simply consists of taking the mean of these vectors. Such heuristic may not be optimal (*e.g.* compared to a maximum likelihood approach), but as detailed in [5], it is convenient when we lack the exact information about the expected distribution of the trail output after partial decryption. In particular, it is useful when linear hull (or related) effects imply errors when determining the approximated probability density functions of multidimensional approximations in linear cryptanalysis (see [9]). As will be exhibited in Section 5, this is exactly the type of situation that we face in this paper.

If we use n such trails, the time complexity is multiplied by the same factor because we have to repeat the partial decryption for each trail. The overhead required to combine the results is negligible. The effect on the data complexity is more intricate because each input mask defines a different partition of the plaintexts, according to the bits that are fixed and those that can change. For example, if the whole codebook is used, each plaintext will be used exactly once for each trail. The plaintexts can either be stored, requiring 2^{67} bytes of memory, or they can be generated on-the-fly, which would require $n * 2^{64}$ encryptions.

In order to evaluate the feasibility of this technique, we applied the statistical saturation attack on 9-round PRESENT for 128 different input masks with 2^{28} texts each. We selected the trails according to two different rules:

1. Best trails: we generated masks using our branch-and-bound and we selected the 8-round trails leading to the lowest theoretical data complexity.
2. Random trails: we generated trails from random (compatible) masks.

The results of our experiments are in Figure 5, which compares the mean gain (as defined in [2]) of attacks against 9-round PRESENT, exploiting different amounts of trails (up to 128), in function of the number of plaintexts used in the attacks. They illustrate that the combination of the information coming from different trails can be done constructively (*i.e.* lead to increased gains). For example, reaching a gain of 10 bits with a single trail requires approximately 2^{24} texts in the left part of the figure. But a combination of 2^7 trails leads to a nearly equivalent gain after 2^{18} texts in this case. Interestingly, the positive impact of combining several trails appears to be reduced for the random trails case (in the left part of the figure). In addition, there are two phenomenons that are worth being mentioned. First, the practical gains of trails having similar theoretical data complexity turned out to be quite different. For example, in the context of the best selected trails, we observed that 12.5% of them led to relatively low bias (less than 4 bits) even after 2^{28} texts. Second, the difference between the best and random trails was not as strong in practice as expected from the theoretical data complexities computed in the previous section. In both cases, this observation relates to the "statistical hull" effect discussed in Section 5.

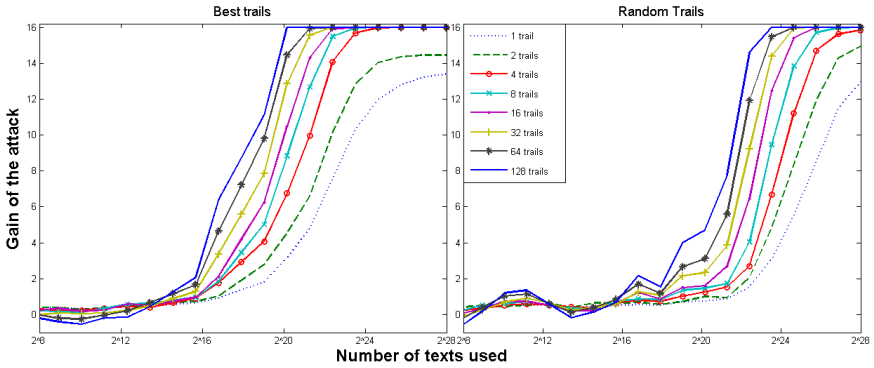

Fig. 5: Gains of attacks using multiple trails (left: best trails, right: random trails).

Note that, since the experiments in Figure 5 are far from using the full codebook of PRESENT, the attacks using different trails also use different plaintexts. By contrast, when estimating the effectiveness of attacks against more than 26-round PRESENT, the data complexity gets close to the full codebook. It means that exploiting multiple trails will require to rearrange (and hence, reuse) the codebook several times. Such a context raises the question to evaluate whether these multiple partitions of the codebook also improve the gain of the attack. Quite naturally, generating the full codebook is unfeasible for a 64-bit cipher. As a first step, we consequently considered a reduced-size version of PRESENT, with 16-bit blocks [12]. This allowed us to compute the average gain of one versus a combination of 128 trails, for different amounts of plaintexts. The results of these experiments are in Figure 6, for attacks against different number of rounds.

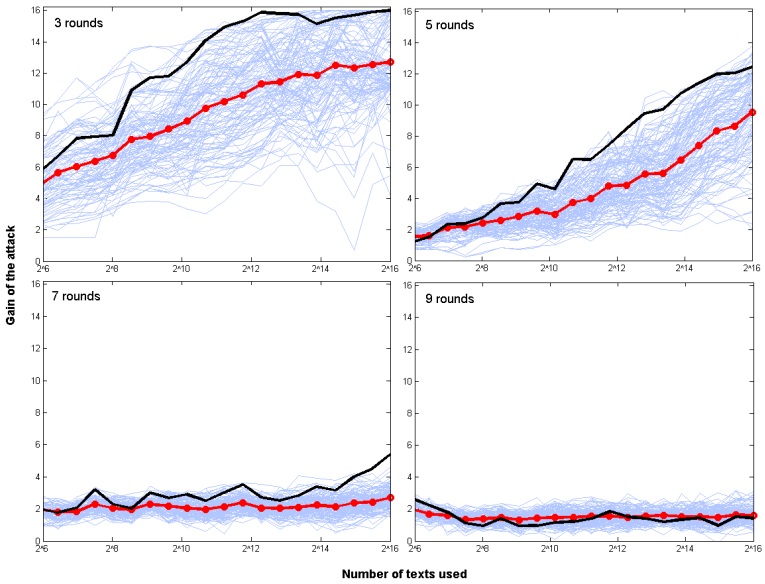

Fig. 6: Comparison between the gain of a single trail and the combined gain of 128 trails, using the full codebook against a simplified PRESENT with 16-bit block size.

A first observation is that, even in this extreme context, the combination of the trails improves the overall gain of the attack significantly. That is, the bold plain curves (representing the combined gains) exceed the bold dotted curves (representing the average gains of single trails - the other curves representing all the 128 single-trail experiments). On the other hand, the improvements are not as large as in Figure 5, arguably because in such a small scale example, the input masks of the different trails are correlated. Also, it is noticeable that for the 9-round case, the gain of the multi-trail attack is similar to the one of a single-trail attack. This illustrates a context where the key-dependent signal provided by a single trail is so small that combining 128 multiple trails is not sufficient to reach a significant gain (since multiple trails can only be used to amplify an existing signal, here too small for the considered data complexities).

Summarizing, in the best case, different trails bring independent information, meaning that using two trails is equivalent to doubling the amount of texts with a single trail (this is the expectation in multiple linear cryptanalysis [2, 9])[1]. In practice, these (best) conditions of independence (*e.g.* for the masks) are not perfectly respected in our context. But as the previous experiments illustrate for reduced-round PRESENT, different trails yield useful information, even when recombining the same set of plaintext with correlated masks. We leave the exact evaluation of these dependencies as an important scope for further research.

[1] Just as it is expected when using multiple fixed values in **ext. 2**.

4.2 Consequence for the security of PRESENT-128

The previous section showed experimentally that combining multiple trails can lead to an improvement of the attack's gain with constant data complexity. In this section, we consider the impact of this observation for the security of PRESENT and quantify the overheads that it causes in terms of time and memory complexity. In particular, we analyze the possibility to perform a key-recovery attack exploiting the complete codebook of PRESENT-128.

According to [6], an attack against 24 rounds requires between 2^{57} and 2^{60} texts and the data complexity increases by a factor of 2^3 for every additional round. This was experimentally confirmed in Figure 4 for up to 16 rounds. If we extrapolate these estimations for 7 more rounds, it amounts to a total of approximately $2^{60+3\cdot 7} = 2^{81}$ texts for 31 rounds, which is more than the whole codebook. However, using multiple trails, we can decrease this complexity by extracting more information from a reduced number of texts. For example, using $2^{81}/2^{64} = 2^{17}$ trails with similar distributions as the one in [6] with the whole codebook - and assuming that they give rise to independent information ! - should be enough to recover 48 bits of the key with a significant gain.

The time complexity of such an hypothetical attack would be 2^{64} memory accesses for each trail, meaning 2^{81} memory accesses for all the 2^{17} trails. It would additionally require 2^{67} bytes to store the codebook. This complexity is slightly higher than an exhaustive search for 80-bit keys, but is a significant improvement for a 128-bit key. Also, there is a possible time-memory tradeoff since one can avoid storing the codebook by re-generating it for each trail.

Again, it is important to emphasize that these complexities are optimistic compared to what would be observed if experiments could be launched with the full codebook. This is because they assume that multiple trails bring independent information. As experimented in the previous section, this is only correct up to a certain (for now, hard to quantify) extent. Hence, it is necessary to multiply our estimated time complexities by a constant factor (as it was done with the data complexities in [6]). These corrective terms should mainly incorporate two effects: first, the possible correlation between different mask and trails as mentioned in this section; second, the possible deterioration and key dependencies of the statistical biases of single trails when the number of rounds increases, due to the statistical hull effect that we detail in the next section. Since we do not have a sound theory to analyze this statistical hull effect, and its experimental evaluation beyond 16 rounds is computationally intensive, we can only conjecture that the combination of multiple trails can be used to trade data complexity for time complexity up to a certain level. Yet, it remains that multiple trails improve the previous results from CT-RSA 2009. And the approximated time complexity of the hypothetical attack (*i.e.* 2^{81}) is small enough compared to 2^{128}, so that there is a reasonable chance that it will remain faster than exhaustive key search, even after the introduction of these corrections. At least, these estimations raise interesting questions about the number of rounds in PRESENT-128.

5 Statistical hull effect

As detailed in Section 4.1, the theoretical data complexities computed following the transition matrices in Section 2 do not always correspond to our practical experiments. In this section, we underline one possible reason explaining this divergence, in relation with the assumption of uniform distributions for the bits that are not part of the trail. That is, while this assumption is nicely respected for the input plaintexts, it becomes incorrect as the number of rounds increases. In fact, this behavior can be related to the statistical hull effect that has been put forward in the context of linear cryptanalysis. A linear hull describes a set of linear approximations that share the same input and output masks, but have different trails and different biases. Consequently, each of these approximations contributes to the bias of the overall approximation [18]. This phenomenon explains why linear cryptanalysis can perform better than expected by the theoretical bias evaluated for a particular approximation. Differential cryptanalysis has a similar concept of differential that is made of several differential characteristics with the same input and output differences, *e.g.* described in [14].

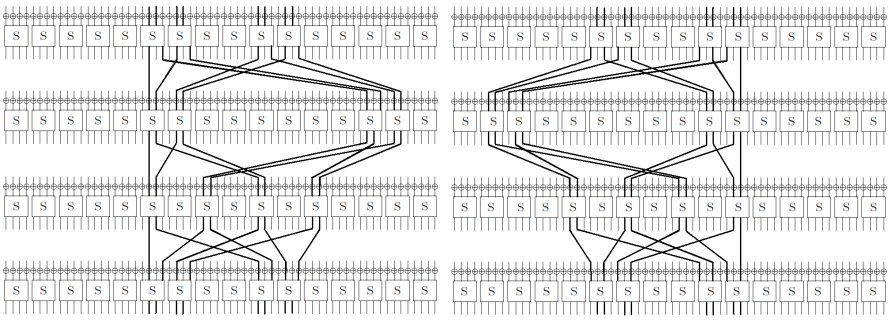

Fig. 7: 4-round trails with the same input and output masks.

In Statistical Saturation Attacks, an analogous phenomenon can also be observed. Namely, several trails with the same active input and output bits can be found, each of them having its own specific transition matrix. For example, two trails with the same input and output masks as the one of Figure 1 are given in Figure 7. By running our branch-and-bound algorithm, we could find numerous other trails corresponding to this input and output masks, as detailed in Table 1.

#rounds	*#trails*
2	1
3	5
4	54
5	1044

Table 1: Number of trails with the same input and output masks as in Figure 7.

The table directly suggests that the number of such trails increases exponentially with the number of rounds. For a large enough number of trails, it consequently becomes difficult to estimate their global effect on the distribution of the output bits. That is, as the trails may be correlated, the combined output distribution is not a simple combination of the theoretical output distributions of each trail. This observation can in fact be related to the work of Keliher *et al.* [13], in which the estimation of an upper bound for the linear hull effect was shown to be computationally hard in the number of rounds.

In the context of linear cryptanalysis, such experiments explain why, as the number of rounds increases, random masks can be almost as effective in recovering a key than a carefully selected trail (as witnessed, *e.g.* by Vaudenay's χ^2 cryptanalysis [21]). Strong hull effects may also imply key dependencies in the sense that the behavior of a trail for different keys may not be identical anymore (hence illustrating that the key equivalence hypothesis first discussed in [8] would not hold for PRESENT). In our (mainly experimental) setting, we conjecture that similar effects explain the deviations between the practical gain of trails having similar deviations from uniform under the assumptions of Section 2. However, we note that for a number of trails (*e.g.* the iterative one in Figure 1), these assumptions holds nicely. Analyzing the possible differences between these experimental observations and the ones made in the context of a linear cryptanalysis in another interesting scope for further research.

6 Conclusion and further works

A summary of the published cryptanalysis results against PRESENT is given in Table 2. It shows that Statistical Saturation Attacks outperform other types of cryptanalyses (in particular linear and differential) against this cipher. This is due to the design of its permutation layer. The main outcome of this paper is to show that the use of multiple trails allows improving the previous result of CT-RSA 2009. Also, if the assumptions in this paper are verified for larger number of rounds, the use of multiple-trails could lead to attacks with smaller time complexity than exhaustive key search against the full PRESENT-128.

As discussed in the previous sections, these estimations have to be considered with care, which is made explicit with the constant multiplicative factor c that we give for the time complexities in the table. This situation is similar to the one in linear cryptanalysis, where the precise estimation of the complexities is made difficult by the large cardinality of the trails to investigate.

In fact, the situation in the present paper is even more difficult, since we have to deal with complete distributions rather than scalar bias values. Positively, the experimental attacks that we performed against reduced number of rounds confirm our theoretical estimations to a reasonable extent. They at least show a significant improvement of the attacks when using multiple trails.

#rounds	Attack	Data compl.	Time compl.	Memory compl.	Ref.
16	SSA	$c * 2^{36}$CP	$2^{28}MA$	2^{16} counters	[6]
16	DC	2^{64}CP	$2^{65}MA$	$6 * 2^{32}$ bits	[23]
17	RKR	2^{63}CP	$2^{104}MA$	2^{53} counters	[20]
24	*SSA*	$c * 2^{60}CP$	$2^{28}MA$	2^{16} *counters*	[6]
26	LH	2^{64}KP	$2^{98.7}MA$	2^{40} counters	[17]
26	MLC	2^{64}KP	$2^{72}MA$	2^{34} bytes	[10]
27	*MT-SSA*	$2^{64}CP$	$c \cdot 2^{69}MA$	2^{67} *bytes*	This paper
29	*MT-SSA*	$2^{64}CP$	$c \cdot 2^{75}MA$	2^{67} *bytes*	This paper
31	*MT-SSA*	$2^{64}CP$	$c \cdot 2^{81}MA$	2^{67} *bytes*	This paper

CP-Chosen Plaintext, KP-Known Plaintext, MA-Memory Access
DC-Differential Cryptanalysis, SSA-Statistical Saturation Attack, RKR-Related Key Rectangle, MLC-Multidimensional Linear Cryptanalysis, LH-Linear Hull

Table 2: Summary of attacks(italic are not experimented and use **ext. 2**, **ext. 4**.).

While these results do not threaten the practical applications of PRESENT (especially since it is mainly its 80-bit version that was advertised in [3]), they raise interesting open questions. For example, they make a case for designing efficient ciphers in which all the statistical effects that can be exploited in cryptanalysis are taken into account. The decorrelation theory appears as an interesting alternative in this respect [22]. But most importantly, the present experimental work implies the need of a better understanding of the statistical saturation attack and its extensions (in particular, **ext. 2**, *i.e.* using multiple fixed values in the trails, and **ext. 4**, *i.e.* using multiple trails). This implies providing sound explanations for the statistical hull and correlation effects between masks and trails, informally described in this paper. The similarities of our results with recent works in multidimensional cryptanalysis [9] also need to be investigated.

References

1. T. Baignères, P. Junod, S. Vaudenay, *How Far Can We Go Beyond Linear Cryptanalysis?*, in the proceedings of ASIACRYPT 2004, Lecture Notes in Computer Science, vol 3329, pp 432-450, Jeju Island, Korea, December 2004.
2. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of CRYPTO 2004, Lecture Notes in Computer Science, vol 3152, pp 1-22, Santa Barbara, California, USA, August 2004.
3. A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, C. Vikkelsoe, *PRESENT: An Ultra-Lightweight Block Cipher*, in the proceedings of CHES 2007, Lecture Notes in Computer Science, vol 4727, pp 450-466, Vienna, Austria, September 2007.
4. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improving the Time Complexity of Matsui's Linear Cryptanalysis*, in the proceedings of The International Conference on Information Security and Cryptology - ICISC 2007, Lecture Notes in Computer Science, vol 4817, pp 77-88, Seoul, Korea, November 2007.
5. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent*, Fast Software Encryption 2008, Lecture Notes in Computer Science, vol 5086, pages 382-397, Springer, February 2008.

6. B. Collard, and F.-X. Standaert, *A Statistical Saturation Attack on the Block Cipher PRESENT*, in the proceedings of CT-RSA 2009, Lecture Notes in Computer Science, vol 5473, pages 195-210, San Francisco, California, USA, April 2009.
7. B. Collard, and F.-X. Standaert, *A Statistical Saturation Attack on the Block Cipher PRESENT, Errata and Improvement*, available for download from: http://www.dice.ucl.ac.be/ fstandae/PUBLIS/62b.pdf
8. C. Harpes, G. Kramer, J. Massey, *A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-Up Lemma*, in the proceedings of EUROCRYPT 1995, LNCS, vol 921 , pp 24-38, Saint-Malo, France, May 1995.
9. M. Hermelin, J.Y. Cho, K. Nyberg, *Multidimensional Extension of Matsui's Algorithm 2*, in the proceedings of FSE 2009, Lecture Notes in Computer Science, vol 5665, pp 209-227, Leuven, Belgium, February 2009.
10. Joo Yeon Cho, *Linear Cryptanalysis of Reduced-Round PRESENT*, Cryptology ePrint Archive: Report 2009/397, available on: http://eprint.iacr.org/2009/397.
11. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Sciences, vol 839, pp 26-39, Santa Barbara, California, USA, August 1994.
12. G. Leander, *Small Scale Variants of the Block Cipher PRESENT*, IACR ePrint Archive, http://eprint.iacr.org/2010/143.
13. L. Keliher, H. Meijer, S.E. Tavares, *New Method for Upper Bounding the Maximum Average Linear Hull Probability for SPNs*, proceedings of EUROCRYPT 2001, Lecture Notes in Computer Science, vol 2045, pp 420-436, Innsbruck, Austria, May 2001.
14. X. Lai, J.L. Massey, and S. Murphy, *Markov Ciphers and Differential Cryptanalysis*, Advances in Cryptology - EUROCRYPT 1991, Lecture Notes in Computer Science, vol 547, pp. 17-38, Brighton, United Kingdom, April 1991.
15. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of EUROCRYPT 1993, LNCS, vol 765, pp 386-397, Lofthus, Norway, May 1993.
16. M. Minier, H. Gilbert, *Stochastic Cryptanalysis of Crypton*, in the proceedings of Fast Software Encryption 2000, Lecture Notes in Computer Science, vol 1978, pp 121-133, New York, USA, April 2000.
17. J. Nakahara Jr, P. Seperhdad, B. Zhang, M. Wang, *Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT*, to appear in the proceedings of CANS 2009, Kanazawa, Japan, December 2009.
18. K. Nyberg, *Linear Approximation of Block Ciphers*, proceedings of EUROCRYPT 1994, LNCS, vol 950, pp 439-444, Perugia, Italy, May 1994.
19. K. Ohkuma, *Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis*, in the proceedings of SAC 2009, Lecture Notes in Computer Science, vol 5867, pp 249-265, Calgary, Alberta, Canada, August 2009.
20. O. Özen, K. Varici, C. Tezcan, *Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT*, proceedings of ACISP 2009, Lecture Notes in Computer Science, vol 5594, pp 90-107, Brisbane, Australia, July 2009.
21. S. Vaudenay, *An Experiment on DES Statistical Cryptanalysis*, In the proceedings of the third ACM Conference on Computer and Communications Security (CCS 1996), ACM, pp. 139-147, New Delhi, India, March 1996.
22. S. Vaudenay, *Decorrelation: A Theory for Block Cipher Security*, Journal of Cryptology, vol 16, num 4, pp 249-286, Springer, 2003.
23. M. Wang, *Differential Cryptanalysis of Reduced-Round PRESENT*, in the proceedings of AFRICACRYPT 2008, Lecture Notes in Computer Science, vol 5023, pp 40-49, Casablanca, Morocco, June 2008.

Part III

REVIEWING THE BASIC ASSUMPTIONS

8. EXPERIMENTS ON THE MULTIPLE LINEAR CRYPTANALYSIS OF SERPENT

Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent

B. Collard*, F.-X. Standaert**, J.-J. Quisquater
UCL Crypto Group, Microelectronics Laboratory, Louvain-la-Neuve, Belgium

Abstract. In 2004, Biryukov *et al.* presented a new theoretical framework for the linear cryptanalysis of block ciphers using multiple approximations. Although they provided first experimental results to confirm the relevance of their approach, a scope for further research was to apply this framework to other ciphers. In this paper, we present various attacks against reduced-round versions of the AES candidate Serpent. Our results illustrate that the hypotheses of Crypto 2004 hold (at least) as long as the number of approximations exploited in the linear attack are computationally tractable. But they also underline the limits and specificities of Matsui's *algorithms* 1 and 2 for the exploitation of such approximations. In particular, they show that the optimal application of *algorithm 2* requires good theoretical estimations of the approximation biases, which may be a problem when the linear hull effect is non-negligible. These results finally confirm the significant reductions of the attacks data complexity that can be obtained from multiple linear approximations.

1 Introduction

The linear cryptanalysis [10] is one of the most powerful attacks against modern block ciphers in which an adversary exploits a linear approximation of the type:

$$P[\chi_P] \oplus C[\chi_C] = K[\chi_K] \tag{1}$$

In this expression, P, C and K respectively denote the plaintext, ciphertext and the expanded key while $A[\chi]$ stands for $A_{a_1} \oplus A_{a_2} \oplus \cdots \oplus A_{a_n}$, with $A_{a_1}, \cdots, A_{a_n}$ representing particular bits of A in positions $a_1, \cdots, a_n$ (χ is usually denoted as a mask). In practice, linear approximations of block ciphers can be obtained by the concatenation of one-round approximations and such concatenations (also called characteristics) are mainly interesting if they maximize the deviation (or bias) $\epsilon = p - \frac{1}{2}$ (where p is the probability of a given linear approximation).

In its original paper, Matsui described two methods for exploiting the linear approximations of a block cipher, respectively denoted as *algorithms* 1 and 2. In the first one, given an r-round linear approximation with sufficient bias, the algorithm simply counts the number of times T the left side of Equation 1 is equal to zero for N pairs (plaintext, ciphertext). If $T > N/2$, then it assumes either $K[\chi_K] = 0$ if $\epsilon > 0$ or $K[\chi_K] = 1$ if $\epsilon < 0$ so that the experimental value $(T - N/2)/N$ matches the theoretical bias. If $T < N/2$, an opposite reasoning

* Work supported by the project Nanotic-Cosmos of the Walloon Region.
** Postdoctoral researcher of the Belgian Fund for Scientific Research (FNRS).

holds. For the attack to be successful, it is shown in [10] that the number of available (plaintext, ciphertext)-pairs must be proportional to $\frac{1}{\epsilon^2}$.

In the second method, a r-1-round characteristic is used and a partial decryption of the last round is performed by guessing the key bits involved in the approximation. As a consequence, all the guessed key bits can be recovered rather than the parity $K[\chi_K]$ which yields much more efficient attacks in practice. In addition, since it exploits a r-1-round approximation rather than a r-round one, it also takes advantage of a better bias which reduces the data complexity.

Among the various proposals to improve the linear cryptanalysis of block ciphers, Kaliski and Robshaw proposed in 1994 an algorithm using several linear approximations [7]. However, their method imposed a strict constraint as it requires to use only approximations implying the same bits of subkeys $K[\chi_K]$. This restricted at the same time the number and the quality of the approximations available. As a consequence, an approach removing this constraint was proposed in 2004 [2] that can be explained as follows. Let us suppose that one has access to m approximations on r block cipher rounds of the form:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \ (1 \leq i \leq m), \tag{2}$$

and wishes to determine the value of the binary vector of parity:

$$\mathbf{Z} = (z_1, z_2, ..., z_m) = (K[\chi_K^1], K[\chi_K^2], ..., K[\chi_K^m]) \tag{3}$$

The improved algorithm associates a counter T_i with each approximation, that is incremented each time the corresponding linear approximation is verified for a particular pair (plaintext-ciphertext). As for *algorithm* 1, the values of $K[\chi_K^i]$ are determined from the experimental bias $(T_i - N/2)/N$ and the theoretical bias ϵ_i by means of a maximum likelihood rule. The extension of *algorithm* 2 to multiple approximations is similarly described in [2].

An important consequence of this work is that the theoretical data complexity of the generalized multiple linear cryptanalysis is decreased compared to the original attack. According to the authors of [2], the attack requires a number of texts inversely proportional to the capacity of the system of equations used by the adversary that is defined as: $c = 4 \cdot \sum_{i=1}^{m} \epsilon_i^2$. Therefore, by increasing this quantity (*i.e.* using more approximations), one can decrease the number of plaintext/ciphertext pairs necessary to perform a successful key recovery.

In this paper, we aim to apply the previously described cryptanalytic tools to the Advanced Encryption Standard (AES) candidate Serpent [1] in order to confirm the analysis of Crypto 2004 and put forward a number of intuitive facts about its implementation. Our results confirm the significant reductions of the attacks data complexity that can be obtained from multiple linear approximations but also their computational limitations. From a more theoretical point of view, multiple linear cryptanalysis is also the best understood technique to take advantage of the linear hull effect [11]. Therefore it allows to fill the gap between the practical [8] and provable security approaches for block ciphers.

Finally, our results underline the specificities of Matsui's *algorithms* 1 and 2 for the exploitation of multiple approximations. In particular, they show that the optimal application of *algorithm 2* requires good theoretical estimations of the approximation biases, which may be a problem when the linear hull effect is non-negligible. As a consequence, sub-optimal strategies sometimes have to be applied. By contrast, our application of *algorithm* 1 nicely follows the predictions of [2], even if the approximation biases are underestimated.

The rest of the paper is structured as follows. Section 2 refers to our linear approximation search algorithm. Section 3 describes preliminary observations on the linear cryptanalysis of Serpent and highlights the existence of a linear hull effect. Sections 4 and 5 respectively provide the experimental results of our attacks against reduced-round Serpent, using *algorithms* 1 and 2. Finally, conclusions are in Section 6 and a description of the Serpent cipher is in Appendix A.

2 Linear approximations search

The first step in a linear cryptanalysis attack consists in finding linear approximations of the cipher with biases as high as possible. But the problem of searching such approximations is not trivial, because of the great cardinality of the set of candidates. In 1994, Matsui proposed a branch-and-bound algorithm making possible to effectively find the best approximation of the DES [12]. However, for practical reasons that are out of the scope of this paper, this method hardly applies to block ciphers with good diffusion such as the AES candidates. As a consequence, we rely on approximations found with a modified heuristic that is described in [3]. Although it does not ensure to obtain the best approximations of Serpent, it provided the best-reported ones in the open literature. Note that, given a r-round approximation found with the branch-and-bound algorithm, the first round masks and last round masks can be replaced by any other mask provided that the biases are left unchanged. Due to the properties of the Serpent S-boxes, several similar approximations can easily be generated with this technique. This allowed us to obtain large sets of approximations involving the same key bits to guess at the cost of dependencies in the linear trails[1].

3 Preliminary observations

Prior to the investigation of multiple linear approximations, we performed experiments with single approximations. We started with *algorithm 1* of which the principle is as follows. For each plaintext-ciphertext pair, we evaluate the left part of equation 1 and increment or decrement a counter given the result. This way, the counter can be used to evaluate the experimental bias of the approximation. If the experimental and theoretical biases have the same sign, then we can presume that the parity of the subkey bits in the right part of Equation 1 is zero. Otherwise, we guess that this parity is one, so that empirical and theoretical results match. As it is suggested in [10], this heuristic relies on a maximum likelihood approach in which we choose the parity so that theoretical and practical results fit well. Thus, the unknown parity can be guessed with an arbitrary high

[1] A linear trail is a set of $r + 1$ masks describing a r-round approximation [11]

probability by computing the experimental mean of the bias and choosing the parity that minimizes the distance between theory and practice. As an illustration, we used a 4-round linear approximation of Serpent with a theoretical bias of 2^{-12} and observed its experimental value for a number of plaintext-ciphertext pairs proportional to 2^{24}. Figure 1 illustrates that the bias value becomes stable after approximately $8/\epsilon^2$ encrypted pairs. It also shows that the theoretical bias (provided by the branch-and-bound algorithm in [3]) was underestimated, which suggests that the linear hull effect is not negligible in our experiments [11]: there are several approximations with the same input/output mask that contribute to the bias in a non negligible way. This effect can cause the complexity of a linear cryptanalysis attack to be overestimated [6].

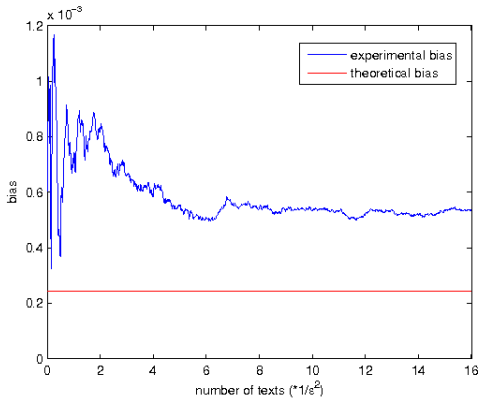

Fig. 1: Evolution of the experimental bias *w.r.t.* the number of known-plaintexts used.

Next to this first experiment, we observed the behavior of 64 linear approximations with various biases, as illustrated in Figure 2. It shows that, provided a sufficient number of encrypted plaintexts, the approximations separate in two classes: the ones with positive bias and the ones with negative bias. This experiment suggests the interest of exploiting multiple linear approximations: since any of these experimental biases provide the adversary with some information on the block cipher key, it is worth trying to exploit them in an efficient way.

Following Kaliski and Robshaw [7], Biryukov *et al.* proposed a general approach to extend Matsui's linear cryptanalysis to multiple linear approximations [2]. As in the simple approximation case, an experimental bias is derived for each approximation in a distillation phase during which counters are extracted from the data. Then, in the analysis phase, an euclidian distance between the theoretical prediction and the experiment is evaluated for each possible parities of the key bits involved in the approximations. The parity minimizing this distance is guessed to be the correct one. The expectation is that the number of encrypted plaintexts required to achieve a given success rate can be reduced when the number of approximation is increased, according to the value of the capacity $c = 4 \cdot \sum_{i=1}^{m} \epsilon_i^2$. In the next sections, we experimentally evaluate the extent to which these expectations can be fulfilled, both for Matsui's *algorithm* 1 and 2.

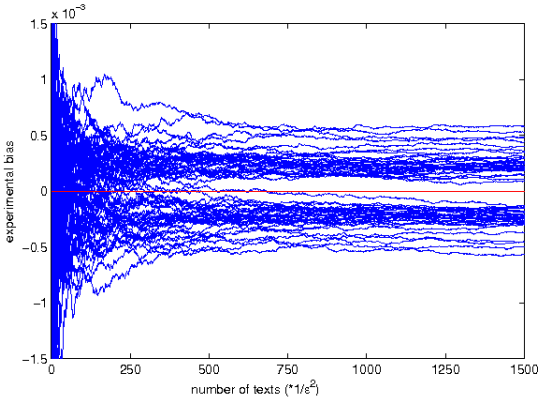

Fig. 2: Evolution of 64 experimental biases *w.r.t.* the number of known-plaintexts used.

4 Experimental attacks with *algorithm 1*

4.1 Selection of the approximations

A significant drawback of Matsui's *algorithm* 1 compared to the second one is that the adversary does not recover master key bits, but a linear equation involving key bits in all the cipher rounds. In the context of non-linear key-scheduling algorithms, this makes the practical exploitation of the attack results difficult since it does not straightforwardly reduce the complexity of an exhaustive key search. When using multiple linear approximations, this drawback can be partially relaxed in the following way[2]. First, the best approximation provided by the branch-and-bound algorithm is selected. Then, only the input/output masks are modified in order to generate large sets of equations. Finally, the adversary progressively increases the size of its system of equations: each time he adds an equation to the system, he also checks the rank r of the corresponding matrix, indicating the number of linearly independent relations in his system. By choosing the system of equations such that the independencies between the equations only relate to meaningful key bits, the adversary ends up with an exploitable information on the cipher key. For example, if the adversary only modifies the input masks to generate a system of the form:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \ (1 \leq i \leq m), \tag{4}$$

he can recover first round key bits. Only one bit (corresponding to the non-variable part of the trail in the system) has to be guessed additionally. As an illustration, we performed attacks against 4-rounds of Serpent, using 64 approximations such that the resulting system of equations has rank $r = 10$.

[2] Of course, it remains that *algorithm* 1 uses a r-round approximation compared to a $r-1$-round approximation in *algorithm* 2 which increases its data complexity.

4.2 Attacks results

Figure 3 depicts the evolution of the distance between the theoretical and experimental biases, for various values of the parity guess and number of encrypted plaintexts. The correct key candidate is expected to minimize this distance which is verified in practice: $32/c$ encrypted plaintexts are sufficient to uniquely determine the correct parity guess. As expected, increasing the number of encrypted plaintexts improves the confidence (or reduces the noise) in the attack result. For example, Figure 4 illustrates the result of a similar attack when $4096/c$ encrypted plaintexts are provided to the adversary. Interestingly, the attack result has a very regular structure underlining the impact of the Hamming distance between different key candidates: close keys have close biases. However, such figure does not tell us (or quantify) how much the exploitation of multiple approximations allowed reducing the attack data complexity.

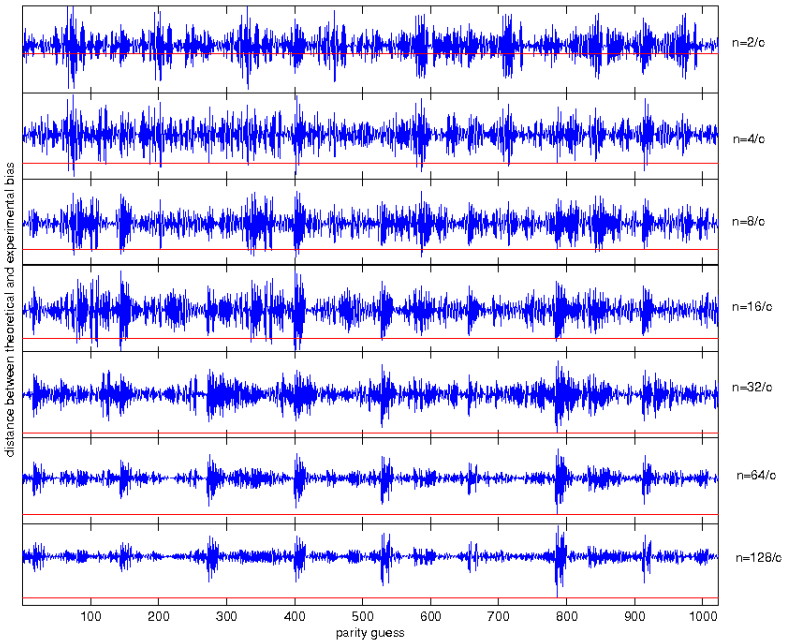

Fig. 3: Evolution of the distance between the theoretical and experimental biases *w.r.t.* the parity guess when the number of encrypted plaintexts increases (64 4-rounds approximations with a capacity of $5.25 \cdot 10^{-6}$). The horizontal line in each graph indicates this distance for the correct parity guess. The scale is the same in each figure.

For this purposes, we ran another set of experiments in which we computed the gain of the attack. As defined in [2], *if an attack is used to recover an n-bit key and is expected to return the correct key after having checked on the average M candidates, then the gain of the attack, expressed in bits, is defined as:*

$$\gamma = -log_2 \frac{2 \cdot M - 1}{2^n} \tag{5}$$

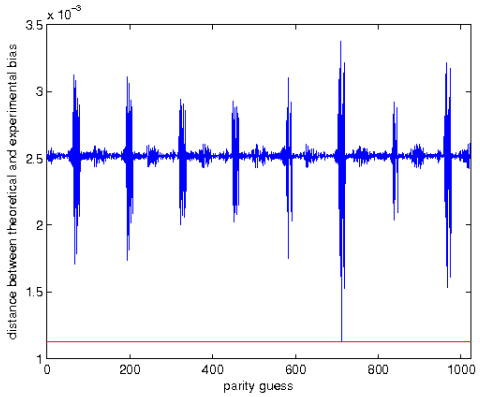

Fig. 4: Evolution of the distance between the theoretical and experimental biases (same as Figure 3) *w.r.t.* the parity guess when using $4096/c$ encrypted plaintexts.

Intuitively, it is a measure of the remaining workload (or number of key candidates to test) after a cryptanalysis has been performed. In the context of multiple linear cryptanalysis attacks, the gain is simply determined by the position of the correct key (or parity) candidate in the weighted list of candidates obtained from the analysis phase. For example, if an attack is used to recover 8 key bits and the correct key candidate is the most likely (*resp.* second most likely), it has a gain of 8 bits (*resp.* 6.42 bits). Importantly, when *algorithm* 1 is used, the maximum gain of an attack depends on the rank of its systems of equations.

In Figure 5, the gain of three attacks are given with respect to the data complexity. The first attack (in red) recovers one bit of parity using only one approximation (*i.e.* it is a simple linear cryptanalysis). The second attack (in green) uses 10 approximations and recovers up to 10 parity bits, while the third attack (in blue) recovers 10 parity bits using 64 linearly dependent approximations. As expected, the gain obtained using 64 approximations increases about 8 times faster than with 10 approximations. This example shows the flexibility of-

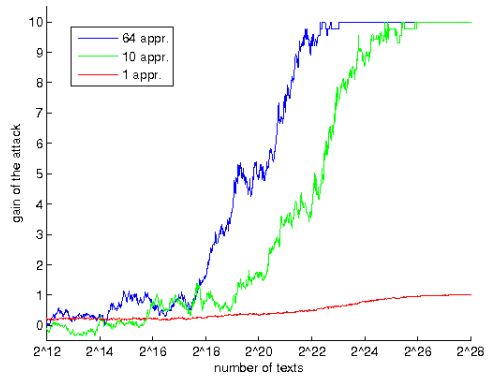

Fig. 5: Evolution of the gain with respect to the number of encrypted plaintexts.

fered by multiple approximations when *algorithm* 1 is applied: they can be used both to get a better gain for a fixed number of plaintext or to get a lower data complexity for a fixed gain. This observation is even better quantified in Figure 6 where the evolution of the gain is given according to the number of encrypted plaintexts normalized by the capacity (*i.e.* the number of plaintexs divided by the joint capacity of the approximations used in the attack). It clearly illustrates the tradeoff between attack complexity and gain. It also confirms that $N \propto 1/c$ is the number of plaintexts required for the attack to reach its maximum gain.

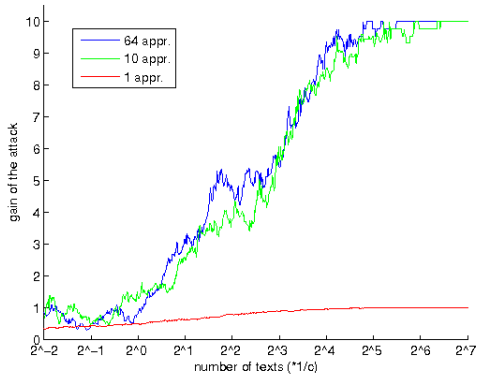

Fig. 6: Evolution of the gain *w.r.t.* the number of plaintexts normalized by the capacity.

4.3 Gain versus success rate and further insights

Let us introduce the following definition:

Definition 1 (success rate). *The success rate of an attack using n approximations (for a given number of plaintexts/ciphertexts) is the number of parity bits guessed correctly among the n parities derived from the distance between the experimental and theoretical bias values.*

Figure 7 represents the evolution of the gain and of the success rate when the number of encrypted plaintexts increases. Interestingly, we can see that the gain increases much faster than the success rate. For example, after about 2^{23} encrypted plaintexts, the gain of the attack reaches its maximum, while the success rate only equals 0.8 at this point. This is a consequence of the linear dependencies between the approximations. Suppose we are given m linear approximations:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \ (1 \leq i \leq m), \tag{6}$$

Such that the following relation holds (case of dependent text masks):

$$\chi_P^1 \oplus \chi_P^2 \oplus ... \oplus \chi_P^m = \chi_C^1 \oplus \chi_C^2 \oplus ... \oplus \chi_C^m \tag{7}$$

Approximation m is linearly dependent in the sense that no additionnal information is given in the deterministic case (i.e. if the approximations hold with probability 1). However, in the probabilistic case, some information can still be extracted (as shown in [13]), as the bias of the $m-th$ approximation is not necessarily related to the bias of the $m-1$ first. When performing multiple linear approximations cryptanalysis, the left part of each approximation is evaluated for a large number of plaintext-ciphertexts pairs and then the parity of the right part (involving subkey bits) is chosen so as to minimize the distance between experimental and theoretical bias. However, some of the parity guess might be wrong in which case the system of equations (where p^i is the parity guess for approximation i):

$$\begin{cases} K[\chi_K^1] = p^1 \\ K[\chi_K^2] = p^2 \\ \quad ... \\ K[\chi_K^m] = p^m \end{cases}$$

can be inconsistent given the linear dependencies, and one or more parity guess must be changed. This can be verified only if there are linear dependencies between the approximations. As the consistency check can be performed before the exhaustive search for the remaining unknown bits, this increases the gain of the attack. Intuitively, this shows that using more approximations than the rank of the system in an attack provides an effect similar to an error correcting code: some parity candidates can be rejected a-priori. By contrast, the success rate of an attack is expected to remain the same when the number of approximations increases (provided that the theoretical biases of each approximation are equal).

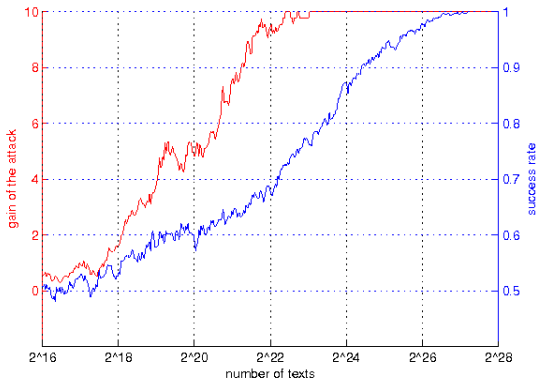

Fig. 7: Evolution of the gain and success rate *w.r.t.* the number of encrypted plaintexts.

As an illustration, we can study the relation between the success rate and the gain of an attack. Suppose that at least n' out of the n approximation parity are guessed correctly. Obviously, the succes rate is higher than n'/n. In order to recover the key (and evaluate the gain), we must generate a list of candidates from the value of the parity bits and then try each candidate until the correct one is found. This can be done using the following strategy:

- Choose the first candidate so as to minimize the euclidian distance between theoretical and experimental bias.
- Assume one guess is incorrect; choose one parity bit and take its complement; try the $\binom{n}{1}$ possible candidates;
- Assume two guesses are incorrect; choose two parity bits and take their complements; try the $\binom{n}{2}$ possible candidates;
- ...
- Assume $n - n'$ guesses are incorrect; choose $n - n'$ parity bits and take their complements; try the $\binom{n}{n-n'}$ possible candidates;

After $n - n'$ steps, we have necessarily found the correct candidate as there is maximum $n - n'$ wrong guesses, thus the gain of the attack equals:

$$\gamma = -log_2\left(\frac{\sum_{i=0}^{n-n'}\binom{n}{i}}{2^n}\right) \tag{8}$$

In this equation, we implicitly assume the independence between the approximations (*i.e.* we assume the maximum gain can be n). However, experiments using up to 416 approximations (including 15 linearly independent ones) show that this prediction fits reasonably well even when the approximation are not independent, as long as the gain does not saturate to its maximum value (see Figure 8). We observe that for a given success rate, the gain of the attack increases with the number of approximations. This example highlights (from a different point of view) the advantage of multiple linear approximations compared to single linear cryptanalysis in which the success rate is equivalent to the gain.

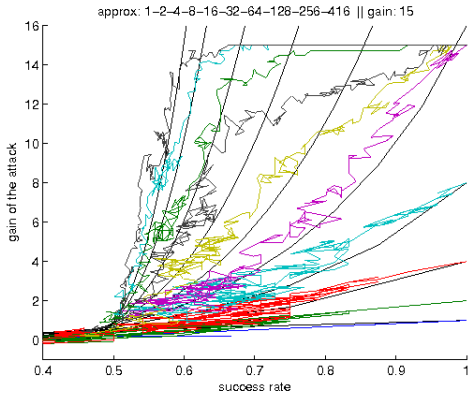

Fig. 8: Evolution of the gain *w.r.t.* the success rate for various number of approximations. This number ranges from 1 to 416 when moving from the bottom curves to the top curves. The black smooth curves are the theoretical predictions.

5 Experimental attacks with *algorithm 2*

In this section, we perform experimental attacks against 5-round Serpent using multiple linear approximations with *Algorithm 2*. It allows the adversary to recover subkey bits in the first/last round of the cipher as follows. An attack against r cipher rounds deals with a linear approximation of the $r-1$ last/first rounds. For each plaintext-ciphertext pair and subkey candidate, the ciphertext is then partially en/decrypted with the subkey candidate, and the approximation is evaluated for the partial en/decryption. A counter indexed by the keyguess value is incremented/decremented according to the parity of the evaluation. For a wrong candidate, the partial en/decryption is expected to produce a random output, thus leading to an null experimental bias (meaning that its statistical evaluation is sufficiently close to zero). For the correct key guess, the experimental bias is expected to converge toward the theoretical bias. In order to speed-up the computations, we used the FFT trick proposed in [4].

5.1 Differences between *algorithms* 1 and 2

Compared with *algorithm 1*, the exploitation of multiple linear approximations with *algorithm* 2 faces an additional problem that we detail in this section. In multiple linear cryptanalysis, an adversary has a system of m linear approximations. Each approximation has a theoretical value for its bias ϵ_i and the adversary additionally obtains an experimental value for this bias ϵ_i^*. When *algorithm* 1 is used, this experimental value is used to minimize the Euclidean distance:

$$\min_{g} \sum_{i=1}^{m} (\epsilon_i - (-1)^{g(i)} \cdot \epsilon_i^*)^2, \tag{9}$$

where g is the parity guess of the linear approximations. But when *algorithm* 2 is used, this experimental bias also depends on the round subkey that is used to perform the first (or last) round partial decryption: $\epsilon_{i,k}^*$. Therefore, the following Euclidean distance has to be computed:

$$\min_{k} \big(\min_{g} \sum_{i=1}^{m} (\epsilon_i - (-1)^{g(i)} \cdot \epsilon_{i,k}^*)^2 \big) \tag{10}$$

The practical consequence of these different conditions can be explained as follows. While the condition in Equation 9 leads to a correct value for the guess, even if the theoretical bias values are underestimated, the condition in Equation 10 only leads to a correct key candidate if a good theoretical estimation of these biases is available. This is due to the key dependencies of the experimental biases in *algorithm* 2. Unfortunately, in the context of the Serpent algorithm investigated in this paper, it has been shown in Section 3 that these theoretical values are not accurate, due to the linear hull effect. As a consequence, the framework of [2] cannot be straightforwardly applied in our context. This is in contrast, *e.g.* with the DES, where such a linear hull effect is negligible [6].

5.2 Attack results

The usual solution to overcome this problem is to look at the maximum experimental bias values. For example, when multiple approximations are considered, a vector of experimental biases can be derived for each approximation. Then the average is taken over all the approximations and its maximum value is expected to indicate the correct key candidate. But the theoretical framework of Crypto 2004 is not applied anymore and such an approach is more closely related to the experiments of Kaliski and Robshaw [7]. As an illustration, Figure 9 illustrates the evolution of the experimental biases averaged over 32 4-rounds approximations, for different numbers of plaintext/ciphertext pairs and 12 bits of keyguess. Figure 10 shows a comparison between simple and multiple linear cryptanalysis. This latter picture illustrates that multiple linear approximations still allow reducing the noise level in the modified attacks which improves their efficiency.

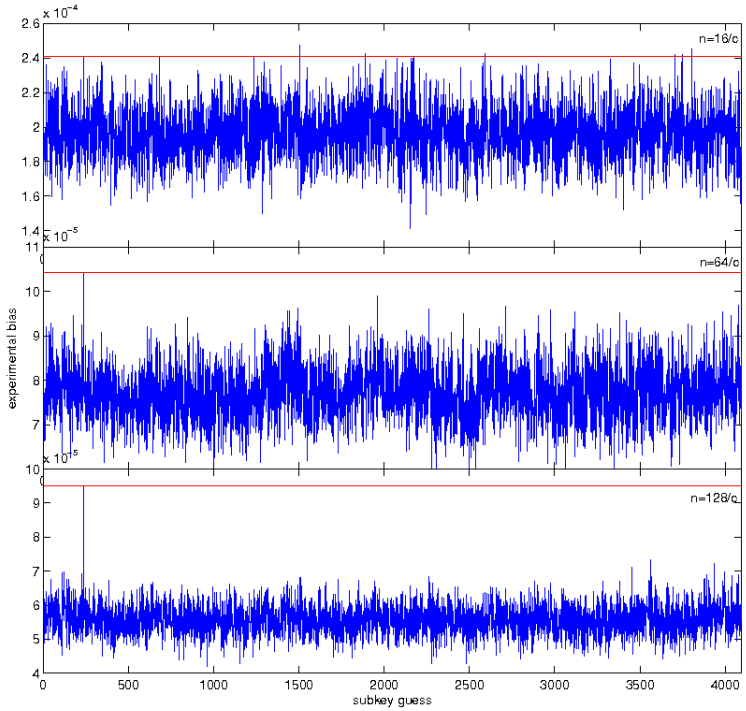

Fig. 9: Evolution of the experimental bias averaged over 32 4-rounds approximations for each guess of the 2^{12} key guesses, when the number of pairs increases ($c = 2.08 \cdot 10^{-7}$).

On the other hand, increasing the number of linear approximations does not involve reductions of the data complexity according to the capacity values as in the previous section. This is caused by the modified analysis phase, in which

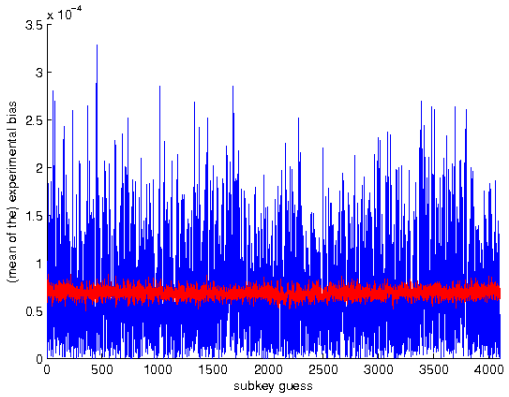

Fig. 10: Comparison of the noise levels between simple and multiple linear cryptanalysis (104 approximations vs. 1 approximation, n=64/c).

the exploitation of the information provided by the different approximations is not optimal anymore. This fact can be emphasized by investigating the gain of the attacks, for different number of approximations. For example, Figure 11 illustrates how the gain of three attacks with respectively 2, 8 and 32 approximations (having the same individual biases) increases. But the benefits are not as spectacular is in the context of *algorithm 1*. Namely, the 32-approximations gain is not increasing 16 times faster than when 2 approximations are used. Compared to the attack results with *algorithm* 1 (*e.g.* in Figure 4), it is finally interesting to notice that Figures 9 and 10 show no particular structure in the biases distribution. This is due to the partial en/decryption of one round that cancels the effects caused by close keys in the Hamming distance sense.

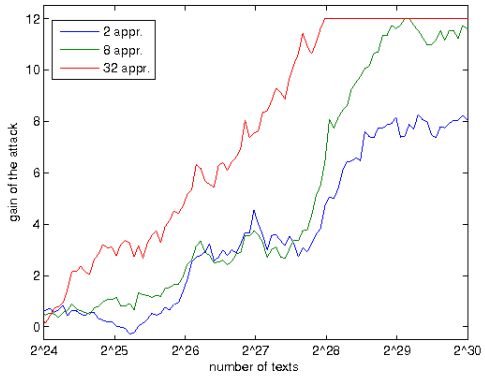

Fig. 11: Evolution of the gain of *algorithm 2* w.r.t. the number of encrypted plaintexts.

6 Conclusion and further works

This paper presented experimental results of multiple linear cryptanalysis attacks against reduced-round versions of the block cipher Serpent. It allowed us to highlight the following observations:

1. The hypotheses stated in [2] about the possible influence of dependencies (between the masks or linear trails) generally appear to be reasonably fulfilled, even for approximations of which a large part of the trail is identical.
2. Our experiments only considered a limited number of approximations. Dependencies effects could appear with more approximations. Note that the number of exploitable approximations is limited anyway, for computational reasons: because of the approximation matrix rank with *algorithm* 1 and because of the amount of partial en/decryptions to perform in *algorithm* 2.
3. By contrast with previous experiments against the DES, we observed a significant linear hull effect, with the following consequences:
   (a) Optimal attacks using Matsui's *algorithm* 1 closely followed the data complexities predicted with the capacity value (defined in [2]), even if the theoretical values of the approximation biases were underestimated.
   (b) Optimal attacks using Matsui's *algorithm* 2 did not lead to successful key recoveries because of the lack of good theoretical estimations of the bias values. Modified heuristics allowed us to take advantage of multiple approximations. But the improvement of the modified attack complexity is not following the predictions of the capacity values.
4. More generally, the analysis of Crypto 2004 leads to meaningful results as long as the branch-and-bound algorithm used to derive the linear approximations provides the adversary with the best possible biases.

In practice, our experiments finally confirmed the significant improvement of multiple linear cryptanalysis attacks compared to Matsui's original attack. Open questions include the optimal exploitation of multiple approximations using *algorithm* 2 when good estimations of the bias values are not available or the extension of these experiments towards more cipher rounds, *e.g.* using [9].

Description of the approximations

A detailed description of the linear approximations used in our experiments is available at the following address:
http://www.dice.ucl.ac.be/~fstandae/PUBLIS/50b.zip

Acknowledgements

This work is partially supported by the Walloon Region under the project Nanotic-Cosmos and by the Belgian Interuniversity Attraction Pole P6/26 BCRYPT.

References

1. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, in the proceedings of the First Advanced Encryption Standard (AES) Conference, Ventura, CA, 1998.
2. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of CRYPTO 2004, Lecture Notes in Computer Science, vol. 3152, pp.1-22, Santa Barbara, California, USA, August 2004.
3. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent*, in the proceedings of InsCrypt 2007, LNCS, pp. 47-61, Xining, China, September 2007.
4. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improving the Time Complexity of Matsui's Linear Cryptanalysis*, In K.-H. Nam and G. Rhee, editor(s), The International Conference on Information Security and Cryptology - ICISC 2007, Volume 4817 of Lecture Notes in Computer Science, pages 77-88, Springer, November 2007.
5. J. Daemen, V. Rijmen, *The Wide-Trail Strategy*, in the proceedings of IMA 2001, LNCS, vol. 2260, pp. 222-238, Cirencester, UK, December 2001.
6. P. Junod, *On the Complexity of Matsui's Attack*, in the proceedings of SAC 2001, LNCS, vol. 2259, pp. 199-211, Toronto, Ontario, Canada, August 2001.
7. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Sciences, vol. 839, pp. 26-39, Santa Barbara, California, USA, August 1994.
8. L.R. Knudsen, *Practically Secure Feistel Ciphers*, in the proceedings of FSE 1993, LNCS, vol. 809, pp. 211-221, Cambridge, UK, December 1993.
9. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, M. Schimmler, *Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker*, in the proceedings of Cryptographic Hardware and Embedded Systems - CHES 2006, Lecture Notes in Computer Science, vol. 4249, Springer, 2006.
10. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of Eurocrypt 1993, LNCS, vol. 765, pp. 386–397, Lofthus, Norway, May 1993.
11. K. Nyberg, *Linear Approximations of Block Ciphers*, in the proceedings of Eurocrypt 1994, LNCS, vol. 950, pp. 439-444, Perugia, Italy, May 1994.
12. M. Matsui, *On Correlation Between the Order of S-boxes and the Strength of DES*, in the proceedings of Eurocrypt 1994, Lecture Notes in Computer Science, vol. 950, pp. 366-375, Perugia, Italy, May 1994.
13. S. Murphy, *The Independence of Linear Approximations in Symmetric Cryptology*, IEEE Transactions on Information Theory, Vol. 52, pp. 5510-5518, 2006.

A The Serpent algorithm

The Serpent block cipher was designed by Ross Anderson, Eli Biham and Lars Knudsen [1]. It was an Advanced Encryption Standard candidate, finally rated just behind the AES Rijndael. Serpent has a classical SPN structure with 32 rounds and a block width of 128 bits. It accepts keys of 128, 192 or 256 bits and is composed of the following operations:

- an initial permutation IP,
- 32 rounds, each of them built upon a subkey addition, a passage through 32 S-boxes and a linear transformation L (excepted the last round, where the linear transformation is not applied),
- a final permutation FP.

In each round R_i, only one S-box is used 32 times in parallel. The cipher uses 8 distinct S-boxes S_i $(0 \leq i \leq 7)$ successively along the rounds and consequently, each S-box is used in exactly four different rounds. Finally, the linear diffusion transform is entirely defined by *XOR*s ($\oplus$), *rotations* ($\lll$) and left *shifts* ($\ll$). Its main purpose is to maximize the avalanche effect within the cipher. If one indicates by X_0, X_1, X_2, X_3 the $4 \cdot 32$ bits at the input of the linear transformation, it can be defined by the following operations:

$$
\begin{array}{r}
\text{input} = X_0, X_1, X_2, X_3 \\
\hline
X_0 = X_0 \lll 13 \\
X_2 = X_2 \lll 3 \\
X_1 = X_1 \oplus X_0 \oplus X_2 \\
X_3 = X_3 \oplus X_2 \oplus (X_0 \ll 3) \\
X_1 = X_1 \lll 1 \\
X_3 = X_3 \lll 7 \\
X_0 = X_0 \oplus X_1 \oplus X_3 \\
X_2 = X_2 \oplus X_3 \oplus (X_1 \ll 7) \\
X_0 = X_0 \lll 5 \\
X_2 = X_2 \lll 22 \\
\hline
\text{output} = X_0, X_1, X_2, X_3
\end{array}
$$

9. EXPERIMENTING LINEAR CRYPTANALYSIS

Experimenting Linear Cryptanalysis

Baudoin Collard*, François-Xavier Standaert**

UCL Crypto Group, Microelectronics Laboratory, Université catholique de Louvain.
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium.

e-mails: baudoin.collard; fstandae@uclouvain.be

Introduction

Since the publication of linear cryptanalysis in the early 1990s, the precise understanding of the statistical properties involved in such attacks has proven to be a challenging and computationally intensive problem. As a consequence, a number of strategies have been developed, in order to design block ciphers secure against cryptanalysis, under reasonable assumptions. In this context, a good assessment of the hypotheses used for the evaluation of linear cryptanalysis and a careful measurement of the distance between actual constructions and theoretical expectations are of particular interest. In this chapter, we present a number of illustrative experiments that allow discussing these issues. Based on a concrete instance of block cipher with small block size, we first evaluate the distance between the so-called practical and provable security approaches for designing block ciphers. Then, we challenge the assumptions of key independence and key equivalence that are frequently used in linear cryptanalysis. Third, we put forward the difficulty of obtaining precise estimations of the distributions within a secure block cipher when the number of rounds increases. We also discuss the consequences of this observation for the key ranking strategies used in order to extract information from actual statistical biases. Finally, we provide systematic experiments of linear cryptanalysis using single and multiple approximations in order to confirm a number of intuitive views that can be found in former papers.

Summarizing, this chapter provides an experimental survey of the basic assumptions in linear cryptanalysis and its consequences for the design of modern block ciphers. It is structured as follows. Section 1 contains background information, including notations, definitions, related works, a specification of our target cipher and a description of Matsui's second algorithm for linear cryptanalysis. Section 2 contains an empirical evaluation of different assumptions in linear cryptanalysis. It discusses the linear hull effect, the pros and cons of the practical security approach, and the key independence and key equivalence hypotheses. Section 3 contains experiments on the test of key-dependent linear biases in different scenarios. Finally, Section 4 briefly browses though the consequences of our experiments and observations for more advanced statistical attacks.

* Work supported by the project Nanotic-Cosmos of the Walloon Region.
** Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

1 Background

1.1 Notations & definitions

The following notations and definitions are standard in linear cryptanalysis. We borrow them from Liam Keliher's and Vincent Rijmen's PhD theses [28, 47].

Definition 1. *An iterated block cipher is an algorithm that transforms a plaintext block of a fixed size n into a ciphertext of identical size, under the influence of a key k, by a repeated application of an invertible transformation ρ, called the round transformation. Denoting the plaintext with x_0 and the ciphertext with x_R, the encryption operation can be written as:*

$$x_{r+1} = \rho_{k_r}(x_r), \qquad r = 1, 2, \ldots, R, \tag{1}$$

where the different k_r are the subkeys generated by a key scheduling algorithm.

For simplicity, we will consider n-bit keys and subkeys in the rest of the paper.

Definition 2. *Let $F : \{0,1\}^n \to \{0,1\}^n$ be a bijection and $\boldsymbol{a}$,$\boldsymbol{b}$ be two masks $\in \{0,1\}^n$. If $X \in \{0,1\}^n$ is a uniformly distributed random variable, then the linear approximation bias $LB(\boldsymbol{a}, \boldsymbol{b})$ is defined as:*

$$LB(\boldsymbol{a}, \boldsymbol{b}) = \Pr_X\{\boldsymbol{a} \bullet X = \boldsymbol{b} \bullet F(X)\} - \frac{1}{2}, \tag{2}$$

where $\bullet$ denotes the scalar product. If F is parametrized by a key K, we write $LB(\boldsymbol{a}, \boldsymbol{b}; K)$ and the expected linear bias $ELB(\boldsymbol{a}, \boldsymbol{b})$ is defined as:

$$ELB(\boldsymbol{a}, \boldsymbol{b}) = \mathop{\boldsymbol{E}}_K \left(LB(\boldsymbol{a}, \boldsymbol{b}; K)\right). \tag{3}$$

The linear bias can be computed for different transformations, e.g. a single S-box, a round function or a complete block cipher. Quite naturally, computing it precisely becomes computationally intensive as the transformation size increases.

Definition 3. *A one-round characteristic for the round i of an iterated block cipher is a pair of n-bit vectors $\langle \boldsymbol{a}_i, \boldsymbol{b}_i \rangle$ respectively corresponding to the input and output masks for this round. A R-round characteristic for rounds $1 \ldots R$ is a $(R+1)$-tuple of n-bit vectors $\Omega = \langle \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{R+1} \rangle$, where $\langle \boldsymbol{a}_i, \boldsymbol{a}_{i+1} \rangle$ respectively correspond to the input and output masks for the round i.*

Definition 4. *A Markov cipher is a block cipher in which the linear (and differential) biases of different rounds are independent of each other, assuming that uniformly random subkeys are used in the different rounds.*

Lemma 1 (Piling up lemma). *Given a vector of independent subkeys $\tilde{K}$ and a Markov cipher, the linear characteristic bias of a characteristic Ω is defined as:*

$$LCB(\Omega, \tilde{K}) = 2^{R-1} \prod_{i=1}^{R} LB(\boldsymbol{a}^i, \boldsymbol{a}^{i+1}; k_i). \tag{4}$$

Note that in an iterated block cipher using a bitwise XOR key addition, the absolute value of $LCB(\Omega, \tilde{K})$ is independent of the subkey vector, but not its sign.

Definition 5. *Given input and output masks* $\boldsymbol{a}$,$\boldsymbol{b}$*, the linear hull* $LH(\boldsymbol{a}, \boldsymbol{b})$ *is the set of all R-round characteristics having* $\boldsymbol{a}$ *as input mask for round 1 and* $\boldsymbol{b}$ *as output mask for round R. The approximated linear hull* $ALH(\boldsymbol{a}, \boldsymbol{b}, N_h)$ *is the subset of* N_h *characteristics in* $LH(\boldsymbol{a}, \boldsymbol{b})$ *having the largest bias* $LB(\boldsymbol{a}, \boldsymbol{b})$*.*

Note that we slightly modified Nyberg's definition of linear hull [43], in order to take into account the possibility of using subsets of characteristics. Finally, the following definition is introduced to reflect the need of estimating the linear bias and expected linear bias from a subset of plaintexts (smaller than the codebook).

Definition 6. *Let* $\mathcal{X}$ *be the set of all plaintexts of a block cipher (aka the codebook) and* $\mathcal{Y}_i \subset \mathcal{X}$*, with* $1 \leq i \leq N_s$*, be different subsets of* $\mathcal{X}$ *containing* N_p *plaintexts chosen uniformly and independently. Then the linear bias sampled with* N_s *sets of* N_p *plaintexts* $SLB(\boldsymbol{a}, \mathbf{b}, N_s, N_p)$ *is defined as:*

$$SLB(\boldsymbol{a}, \mathbf{b}, N_s, N_p) = \sum_{i=1}^{N_s} \Pr\{i\} \cdot \left(\widehat{\Pr_{Y_i}}\{\boldsymbol{a} \bullet X = \boldsymbol{b} \bullet F(X)\} - \frac{1}{2} \right). \tag{5}$$

If F *is parametrized by a key* K*, we write* $SLB(\boldsymbol{a}, \boldsymbol{b}, N_s, N_p; K)$*. If we additionally denote a uniformly selected subset of keys of cardinality* N_k *by* $\mathcal{L} \subset \mathcal{K}$*, the sampled expected linear bias* $SELB(\boldsymbol{a}, \boldsymbol{b}, N_s, N_p, N_k)$ *is defined as:*

$$SELB(\boldsymbol{a}, \boldsymbol{b}, N_s, N_p, N_k) = \sum_{K \in \mathcal{L}} \Pr\{K\} \cdot SLB(\boldsymbol{a}, \boldsymbol{b}, N_s, N_p; K). \tag{6}$$

The SLB is usually referred to as the experimental bias in the literature.

1.2 Related works

Design approaches to prevent linear cryptanalysis

Worst case security. Let K denote the master key of a target block cipher. In the worst case, an adversary would perform linear cryptanalysis using a pair of masks $(\mathbf{a}, \mathbf{b})$ such that:

$$(\mathbf{a}, \mathbf{b}) = \underset{(\mathbf{x}, \mathbf{y})}{\operatorname{argmax}} \ LB(\mathbf{x}, \mathbf{y}; K). \tag{7}$$

Following [36], it directly yields the approximated data complexity of the attack[1]:

$$N \approx \frac{c}{\max \ LB(\mathbf{a}, \mathbf{b}; K)^2}, \tag{8}$$

where c is a small constant value. However, as will be detailed next, this worst case strategy cannot be directly exploited by actual adversaries. In practice, the

[1] More sophisticated approximations are discussed in [6, 23, 24, 51].

direct computation of Equation (7) is generally infeasible, both because of an unknown key and for computational reasons (estimating the bias of an n-bit permutation requires $n \cdot 2^{2n}$ operations, i.e. more than exhaustive key search).

The key equivalence hypothesis. As a consequence of the previous limitations, design strategies to prevent linear cryptanalysis usually start by assuming key equivalence, as introduced by Harpes et al. in [17]. That is, they assume that the linear bias will be close to its average value for the vast majority of keys:

$$LB(\mathbf{a}, \mathbf{b}; \tilde{K}) \approx ELB(\mathbf{a}, \mathbf{b}). \tag{9}$$

Practically secure block ciphers. Next to the key equivalence hypothesis, arguing about security against linear cryptanalysis requires to find ways to evaluate the linear biases in a computationally tractable manner. As shown by Lemma 1, a simple solution for this purpose is to use the concept of characteristic. For a Markov cipher and assuming independent round keys, the probability of a R-round characteristic can be computed as a product of 1-round characteristic probabilities. Hence, a designer can run an algorithm to search the characteristic Ω_{max} such that $LCB(\Omega_{max})$ is maximal and then assume:

$$ELB(\mathbf{a}, \mathbf{b}) \approx LCB(\Omega_{max}). \tag{10}$$

Lars Knusden calls a block cipher practically secure if the data complexity determined by this method is prohibitive [29]. Obviously, such an approach is only valid up to a certain extent and it may give rise to false intuitions. For example, increasing the number of rounds in a block cipher always reduces the linear characteristic bias, while the actual expected linear bias of a cipher cannot be decreased below a certain threshold, depending on its block size. Positively, the practical security approach is a simple way to estimate the number of rounds required for a block cipher to become hard to distinguish from a random permutation. It is also the basis of the *wide-trail strategy* [12], that has been successful for designing many modern block ciphers, most notably the AES Rijndael [13]. In the wide-trail strategy, one essentially ensures that each characteristic involves a large number of S-boxes in each of the block cipher rounds, and that these S-boxes do not have highly probable linear approximations.

Provable security. In contrast with the practical security approach, the theory of provable security against linear cryptanalysis attempts to compute expected linear biases, i.e. to consider all the characteristics in a linear hull rather than only the best one[2]. An example of design strategy based on such a theory was proposed by Matsui in [38], and gave rise to the design of the block cipher Misty [39]. A similar line of work was followed by Keliher et al. in [26] and applied to the AES Rijndael in [27]. The main benefit of provable security is to provide security guarantees that only rely on the key equivalence hypothesis, and assuming independent round keys. Its main limitation is the computational difficulty of finding tight bounds, when the number of block cipher rounds increases.

[2] In a very similar way, provable security against differential cryptanalysis considers the concept of differential rather than the one of differential characteristic [44].

Decorrelation theory. Vaudenay proposed an alternative solution for designing block ciphers with provable security against a large class of attacks in [52]. It essentially aims at preventing the use of the key equivalence hypothesis in an attack. For this purpose, decorrelation modules are used, that are key-dependent transformations making the linear bias (and differential probability) of a given approximation highly key-dependent, so that any attack that chooses the input and output masks a priori will fail. Examples of block ciphers based on the decorrelation theory include the cipher C [2] and the Krazy Feistel Cipher [3].

Experimental evaluations of assumptions and attacks

Following Matsui's experiments on the DES [36], various publications contain empirical evaluations of the linear cryptanalysis and its underlying assumptions. All these empirical works are tightly connected with our following analyzes. We list a few of them for illustration. First, Junod reported new results on the linear cryptanalysis of the DES in 2001 [22], together with a discussion of the attack's complexity. Rouvroy et al. performed similar experiments, exploiting the computational power of reconfigurable hardware [49]. Extensions of these works, considering the use of multiple linear approximations can be found, e.g. in [5, 10, 20]. Second, the assumption of independent round keys is discussed and tested experimentally by Knudsen and Mathiassen in [32]. They show that the key scheduling algorithms used in practical block cipher constructions has an impact on this assumption, but that the conclusions drawn when independent round keys are used should still reasonably hold with a good key schedule. Third, the relevance of the practical security approach is analyzed by Selçuk in [50], with experiments conducted against reduced versions of RC5 and Feistel ciphers. This paper also discusses the impossibility to accurately evaluate linear approximations in block ciphers by means of statistical sampling. A very similar approach was followed in [46], in which experiments are performed against small scale block ciphers, in order to review the practical security approach and the key equivalence hypothesis for various block sizes, S-boxes and diffusion layers. A counterpart of these experiments in the context of differential cryptanalysis can be found in [7]. Finally, Daemen and Rijmen investigated the statistical distributions of fixed-key linear probabilities, both theoretically and empirically, in [14]. Their results allow replacing the key equivalence hypothesis by a precise understanding of the fixed-key vs. average behavior in block ciphers.

1.3 Target cipher

The experiments in the next sections will be performed against the block cipher `SmallPresent`, with 16-bit block size, described in [33]. `SmallPresent` is an iterated block cipher with R-rounds, each of them made of a key addition layer, a substitution layer and a permutation layer. The key addition layer is a bitwise XOR, the substitution layer applies four 4-bit S-boxes to the state and the permutation layer is a simple wire crossing (or bit permutation).

1.4 Matsui's second algorithm

Matsui's original description of the linear cryptanalysis comes with two algorithms that allow exploiting linear approximations in iterated block ciphers [35]. In this paper, we focus on the second one, that is intuitively pictured in Figure 1.

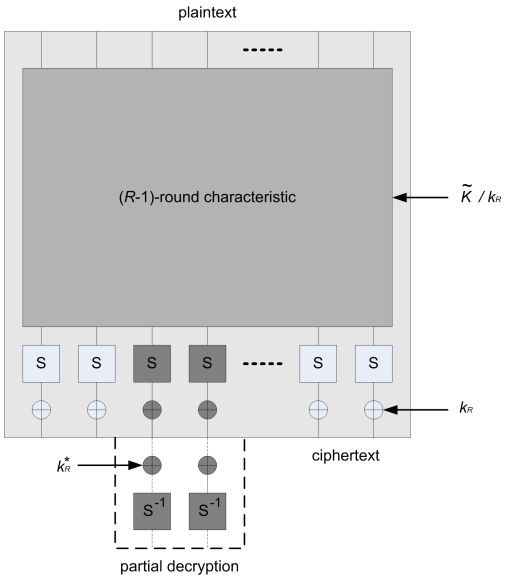

Fig. 1: Linear cryptanalysis with Matsui's M2 algorithm.

In this setting, an adversary exploits a $(R-1)$-round characteristic with good LCB. This characteristic ideally connects to a limited number of active S-boxes in the last block cipher round (e.g. the two dark grey boxes in Figure 1), so that by guessing a few key bits of k_r, it is possible to do a partial decryption of the ciphertext for these S-boxes[3]. As a result, the adversary is able to sample the linear bias of round $(R-1)$ under the various key hypotheses for k_r. We denote this sampled bias as $SLB(\mathbf{a}_1, \mathbf{a}_R; k_r)$. Eventually, the adversary selects the key candidate that maximizes some function of this sampled bias. Summarizing, a linear cryptanalysis using Matsui's M2 algorithm requires: (1) a way to generate good characteristics and (2) a way to test the key-dependent sampled biases.

1. Generation of characteristics. This step usually exploits a variety of heuristics, starting with the inspection of the non-linear components in block ciphers (e.g. S-boxes) and trying to extend partial approximations from one round to multiple rounds. For example, a branch-and-bound algorithm can be used for this purpose [37]. Intuitively, such an algorithm concatenates all possible 1-round approximations, and compares the resulting biases (estimated thanks to the piling

[3] The complexity of such a partial decryption for a n-bit key guess is in $\mathcal{O}(n \cdot 2^n)$ [9].

up lemma) with a lower bound, in order to reject "bad" approximations. For certain ciphers, e.g. the DES, it allows to find the best characteristics. For other ones, e.g. SERPENT, the amount of possible approximations grows in such a way that maintaining a list of all possibly optimal characteristics becomes impossible as the number of rounds increases, because of memory constraints [8]. The precise description of these heuristics is out of the scope of this paper. Nevertheless, we mention that, as we investigate a 16-bit cipher, it was always possible to exhaustively find the best characteristics in the next sections.

2. Testing the key-dependent sampled biases. Again, various solutions are possible, of which the precise description is out of the scope of this paper. To keep it short, we will refer to two general types of approaches. The first one, requires that a good estimation of the linear bias LB is available to the adversary, e.g. using the linear characteristic bias LCB. In this case, it is possible to apply a maximum likelihood approach. Under certain assumptions detailed in [24], this leads to the simple rule to select the key candidate that minimizes the euclidean distance between the sampled and estimated bias values:

$$\hat{k_r} = \underset{k_r^*, \mathsf{P}(\Omega,\tilde{K})}{\operatorname{argmin}} \left(SLB(\mathbf{a}_1, \mathbf{a}_R; k_r) - LCB(\Omega, \tilde{K}) \right)^2, \tag{11}$$

where $\mathsf{P}(\Omega, \tilde{K})$ denotes the parity of the subkey bits used in the characteristic Ω. The second type of approach is used if a good estimation of the bias is not available. In this case, one can rely on different types of heuristics, of which a classical one is simply to select the key that maximizes the sampled linear bias:

$$\hat{k_r} = \underset{k_r^*}{\operatorname{argmax}} \left(SLB(\mathbf{a}_1, \mathbf{a}_R; k_r) \right). \tag{12}$$

The next sections provide different experiments in order to illustrate these two steps. Namely, Section 2 discusses the difference between the linear characteristics that can be efficiently generated by actual adversaries and the corresponding linear approximations. Then, Section 3 discusses the exploitation of these characteristics and their test with the previously described maximum likelihood and heuristic approaches, including experiments using multiple approximations.

2 Evaluation of characteristics and approximations

2.1 Computing linear hulls

The first experiments we performed relate to the notion of linear hull introduced by Nyberg in [43]. Thanks to the limited block size of our target cipher, it was possible to generate the entire linear hull for a given pair of input/output masks $(\mathbf{a}, \mathbf{b})$, with a branch-and-bound algorithm. In addition, it was also possible to set the lower bound of the branch-and-bound arbitrarily low, in order to generate all characteristics with non-zero LCB. As expected, the number of characteristics in the hull increased exponentially with the number of rounds.

In practice, we generated the linear hull for the 100 best characteristics of the cipher. The size of the linear hull was between 2 and 77 after three rounds, between 54 and 51,388 after four rounds and between 991 and 1,826,043 after five rounds. For more than five rounds, we could not generate the complete linear hull, because of limited computational resources. While far from being exhaustive, these experiments suggest that, for a fixed number of rounds, the number of characteristics in a linear hull varies considerably according to the choice of the input/output masks (in particular, the number of active S-boxes in these input/output masks has a significant impact in this respect).

Cheating with the full codebook. Considering small block ciphers allows to generate the full codebook. An interesting consequence of this possibility is that one can consider the block cipher with a fixed master key as a large 16-bit S-box. And for any pair of masks $(\mathbf{a}, \mathbf{b})$, we can then compute the exact linear bias $LB(\mathbf{a}, \mathbf{b}; K)$. There exists two equivalent ways allowing to perform this task. The first one is to compute the FFT of the S-box directly, e.g. using the technique described in [48], which is possible for any number of rounds. The second one is to exploit the knowledge of the linear hull, as long as it is available (i.e. for a reduced number of rounds), and to compute:

$$LB(\mathbf{a}, \mathbf{b}; K) = \sum_{\Omega \in LH(\mathbf{a}, \mathbf{b})} LCB(\Omega, \tilde{K}). \tag{13}$$

It is important to note that this equality does not straightforwardly lead to good approximations of the linear bias for the adversaries, as it only holds if the parity of all the subkey bits that are involved the characteristics (i.e. the bias signs) are taken into account [15]. As the number of such parities increases with the size of the hull, it rapidly becomes intensive to guess by any practical adversary.

Also, Equation (13) is only tight if the linear bias is evaluated with the full codebook and linear hull. In this respect, an interesting experiment is to evaluate how the quality of the estimation for $LB(\mathbf{a}, \mathbf{b}; K)$ degrades when the number of plaintexts and characteristics in the hull decreases. For this purpose, we generated the complete linear hull for the linear approximation $(\mathbf{a}, \mathbf{b}) = (0x0770, 0x0111)$, for 5 rounds of `SmallPresent-`[16]. We found 916,841 characteristics corresponding to these masks. Their linear characteristic bias ranged from 2^{-9} for the best approximation to 2^{-24} for the 69,632 worst ones[4].

Figure 2 shows the impact of sampling with less plaintexts than the full codebook: every line in the picture represents one random key. We see that the distance between the exact linear bias $LB(\mathbf{a}, \mathbf{b}; K)$ and its sampled value $SLB(\mathbf{a}, \mathbf{b}, 1, N_p; K)$ only converges to zero when N_p equals 2^{16}. Figure 3 shows the impact of sampling with an approximated linear hull $ALH(\mathbf{a}, \mathbf{b}, N_h)$: every point in the picture represents a random key and its color scale indicates the size of the approximated hull N_h. We see that the approximation only converges towards the correct value of $LB(\mathbf{a}, \mathbf{b}; K)$ when N_h gets close to $916,841 \approx 2^{20}$.

[4] That is, the approximations with the smallest non-zero bias.

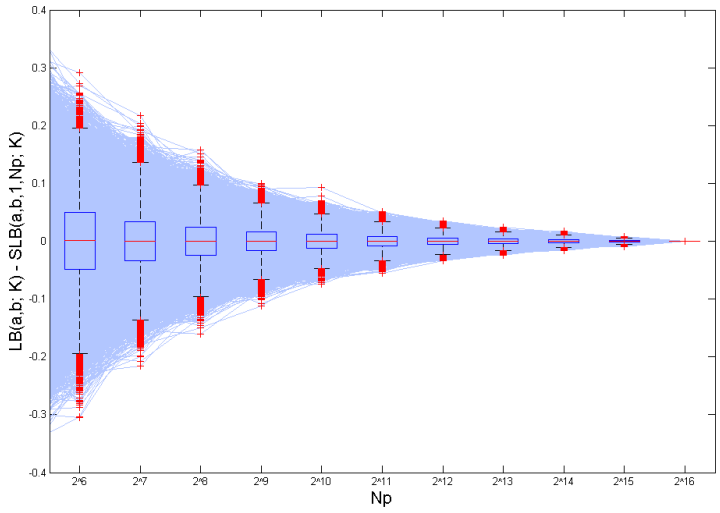

Fig. 2: Estimation of $LB(\mathbf{a}, \mathbf{b}; K)$ with the complete linear hull, in function of the number of plaintexts used in the sampling ($2^5 \leq N_p \leq 2^{16}$, $N_s = 1$), for different keys.

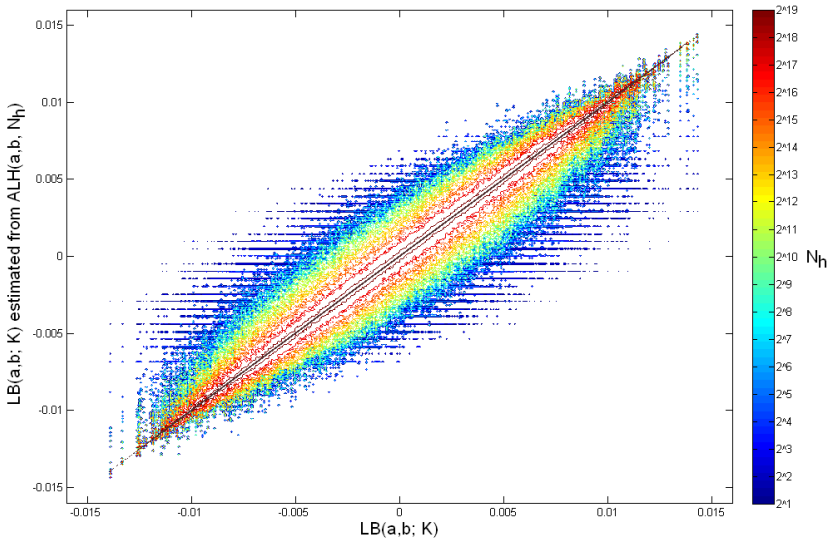

Fig. 3: Estimation of $LB(\mathbf{a}, \mathbf{b}; K)$ with the complete codebook, in function of the number of elements in the approximated linear hull ($2^1 \leq N_h \leq 2^{19}$), for different keys.

2.2 A note on Murphy's technical report

Sean Murphy recently posted a technical report about the existence of the so-called "linear hull effect" [41], first described by Nyberg in 1994. As this discussion closely relates to the experiments in this paper, we briefly comment on it. Summarizing, Murphy's observation relates to the following equality from [43]:

$$\mathop{\mathbf{E}}_{K}\left(LB(\mathbf{a},\mathbf{b};K)^2\right) = \sum_{\Omega \in LH(a,b)} \left(LCB(\Omega,\tilde{K})^2\right). \tag{14}$$

This equation can be illustrated with a simple example. Let us define a pair of masks $(\mathbf{a},\mathbf{b})$, such that the linear hull for a given cipher equals $LH(\mathbf{a},\mathbf{b}) = \{\Omega_1,\Omega_2\}$. Let us also assume that $LCB(\Omega_1,\tilde{K}) = \epsilon_1$ or $-\epsilon_1$, depending on the parity of $\tilde{K}$, and that $LCB(\Omega_2,\tilde{K}) = \epsilon_2$ or $-\epsilon_2$, depending on the parity of $\tilde{K}$. In this case, and assuming uniformly random keys, there will be four possible parities and the left part of Equation (14) can be simply evaluated as:

$$\begin{aligned}\mathop{\mathbf{E}}_{K}\left(LB(\mathbf{a},\mathbf{b};K)^2\right) &= \frac{1}{4}\cdot\left((\epsilon_1+\epsilon_2)^2+(\epsilon_1-\epsilon_2)^2+(-\epsilon_1+\epsilon_2)^2+(-\epsilon_1-\epsilon_2)^2\right),\\ &= \epsilon_1^2+\epsilon_2^2,\\ &= \sum_{\Omega\in LH(a,b)}\left(LCB(\Omega,\tilde{K})^2\right).\end{aligned}$$

Using this example as a case study, the issue raised by Murphy can be explained as follows. While Equation (14) is correct, it cannot be used to evaluate the average data complexity of a linear cryptanalysis. This is because the average data complexity is proportional to the average over the keys of the inverse of the squared linear bias. And this quantity is not equal to the inverse of the average over the keys of the squared linear bias. That is, in our example:

$$\frac{1}{\mathbf{E}_K\left(LB(\mathbf{a},\mathbf{b};K)^2\right)} = \frac{1}{\epsilon_1^2+\epsilon_2^2}, \tag{15}$$

$$\mathop{\mathbf{E}}_{K}\frac{1}{\left(LB(\mathbf{a},\mathbf{b};K)^2\right)} = \frac{\epsilon_1^2+\epsilon_2^2}{(\epsilon_1+\epsilon_2)^2\cdot(\epsilon_1-\epsilon_2)^2}. \tag{16}$$

As these two values are related by Jensen's inequality, Murphy correctly concludes that Equation (15) can only provide a lower bound for the data requirements of a linear cryptanalysis. The practical impact of this observation can easily be seen with the particular case in which $\epsilon_1 = -\epsilon_2$, leading to a zero value for the bias $LB(\mathbf{a},\mathbf{b};K)$. Clearly, such a situation is only interpreted correctly by Equation (16), resulting in an infinite data complexity. A more intuitive view of this result is that, by squaring the bias values, one looses the sign information that is crucial in combining the different approximations constructively.

We note that, while the previous observation is sound, it does not contradict the existence of a linear hull effect which, as will be discussed in the next section, relates to a trivial difference between a linear characteristic bias, estimated with

the piling up lemma, and the linear bias of its corresponding masks. Also, it remains that Nyberg's relation in Equation (14) can be used to compute a lower bound for the data complexity of a linear cryptanalysis, in a less computationally intensive manner than with the direct computation of the linear biases.

2.3 Pros & cons of the practical security approach: the linear hull effect

In our following experiment, we ran a branch-and-bound algorithm in order to find the best characteristic Ω_{max}, for different number of block cipher rounds, and denoted the corresponding pairs of input and output masks as $\mathbf{a}_{max}, \mathbf{b}_{max}$. Then, we evaluated the following quantities in Figure 4:

$$\mathop{\mathbf{E}}_{K} \; LB(\mathbf{a}_{max}, \mathbf{b}_{max}; K),$$
$$\mathop{\mathbf{var}}_{K} \; LB(\mathbf{a}_{max}, \mathbf{b}_{max}; K).$$

In this figure, the continuously decreasing line represents the linear characteristic bias $LCB(\Omega_{max}, \tilde{K})$, estimated with the piling up lemma. And the bounded line represents the corresponding biases $LB(\mathbf{a}_{max}, \mathbf{b}_{max}; K)$, computed for different keys (dots representing the sample mean, crosses representing individual experiments with different keys). These results clearly allow to give an informal description of the linear hull effect. Namely, as soon as the number of characteristics in the hull $LH(\mathbf{a}_{max}, \mathbf{b}_{max})$ increases, the distance between $LCB(\Omega_{max}, \tilde{K})$ and $LB(\mathbf{a}_{max}, \mathbf{b}_{max}; K)$ also increases, resulting in an overestimation of the attack data complexity with $LCB(\Omega_{max}, \tilde{K})$. This clearly emphasizes the pros and

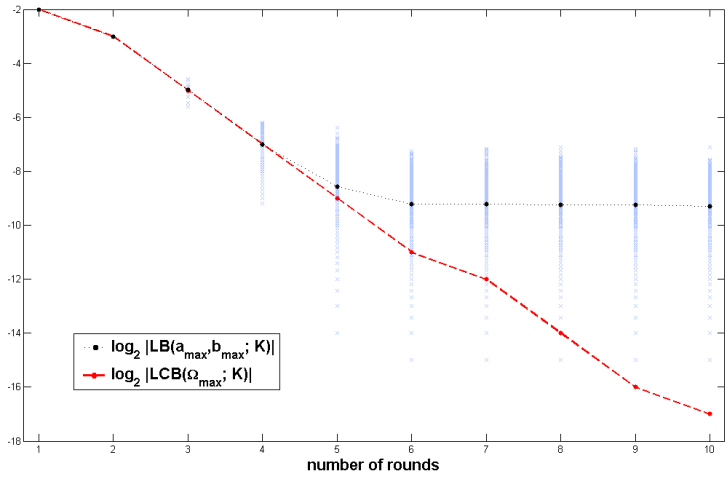

Fig. 4: Comparison between the linear characteristic bias $LCB(\Omega, \tilde{K})$ estimated with the piling up lemma and the corresponding bias $LB(\mathbf{a}_{max}, \mathbf{b}_{max}; K)$, for different keys.

cons of the practical security approach. On the one hand, the relevance of single characteristics trivially vanishes as the number of rounds increases. This "linear hull effect" is in fact dominant as soon as the cipher is practically secure (according to Knudsen's definition). On the other hand, the pair of masks $(\mathbf{a}_{max}, \mathbf{b}_{max})$ generated with the branch-and-bound algorithm and exploited by the adversary no longer corresponds to the best approximation in this case. After a sufficient number of rounds, this best characteristic behaves like a random one: the mean of $LB(\mathbf{a}_{max}, \mathbf{b}_{max}; K)$ is smaller than $2^{-\frac{n}{2}}$ and its variance over the keys is large. It prevents the application of successful attacks exploiting these characteristics.

2.4 Best, worst and average cases

The previous section investigated the practical security approach and an adversary who can only find linear characteristics with a time complexity below the one of exhaustive key search, e.g. with a branch-and-bound algorithm. We now tackle the more theoretical situation in which the best linear approximations can be found, for each key. For this purpose, we again considered `Small-Present-`[16] with various number of rounds and computed the maximum, minimum[5] and average values for $LB(\mathbf{a}, \mathbf{b}; K)$, averaged over the keys. That is, we computed:

$$\begin{aligned} \mathop{\mathbf{E}}_{K} \max_{\mathbf{a},\mathbf{b}} \; & LB(\mathbf{a}, \mathbf{b}; K), \\ \mathop{\mathbf{E}}_{K} \min_{\mathbf{a},\mathbf{b}} \; & LB(\mathbf{a}, \mathbf{b}; K), \\ \mathop{\mathbf{E}}_{K} \mathop{\mathbf{E}}_{\mathbf{a},\mathbf{b}} \; & LB(\mathbf{a}, \mathbf{b}; K), \end{aligned}$$

and the corresponding variances. The results are in Figure 5 and illustrate that:

- After a few first rounds for which the block cipher is practically insecure, the value of the maximum, minimum and average linear biases stabilizes.
- Contrary to the case where the masks are fixed and determined by a branch-and-bound algorithm (as in the previous section), the variance of these quantities over the keys is small (and decreases with the block size [46]).
- Most importantly, the maximum bias value is between 2^{-6} and 2^{-7}.

This last point emphasizes that in theory, for a given cipher and key, a low data complexity (i.e. lower than exhaustive key search) linear cryptanalysis is always possible. However, this does not mean that the practical security approach is not good for designing ciphers. The goal of the practical security approach is not to prevent the existence of good linear approximations, but to make them hard to actually find and exploit, for computational reasons. This is because, for practically secure ciphers, the best approximations cannot be found anymore by chaining small approximations. Finding these best approximations requires (1) to enumerate the full linear hull and (2) to take the sign of each characteristic bias into account. In both cases, this implies computationally intensive tasks.

[5] That is, again, the approximations with the smallest non-zero bias.

Additionally, as far as Matsui's second algorithm is concerned, it is also necessary that these best approximations only imply a limited number of active S-boxes in the first/last rounds, so that an efficient key guessing can be performed.

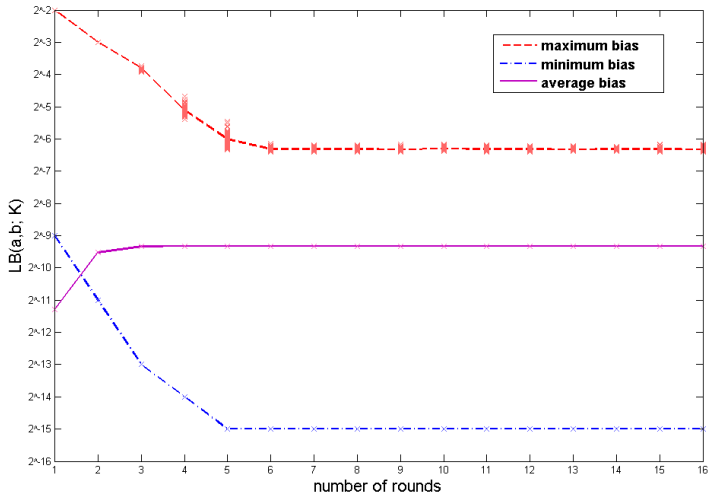

Fig. 5: Maximum, minimum and average values of the linear bias $LB(\mathbf{a}, \mathbf{b}; K)$.

2.5 Key independence and key equivalence hypotheses

Before moving to the empirical evaluation of Matsui's second algorithm, this section briefly discusses the key independence and key equivalence hypotheses. First, although we did not perform as intensive experiments as Knudsen and Mathiassen in [32], we mention that considering independent round keys or the key scheduling algorithm in [33] did not lead to significant differences in our previous experiments and conclusions. In general, it seems that the assumption of independent round keys is reasonably fulfilled by modern ciphers.

Our observations on the key equivalence hypothesis are more contrasted. Clearly, Figure 4 shows that, given a fixed pair of masks $(\mathbf{a}, \mathbf{b})$, the key equivalence hypothesis of Equation (9) is not respected as soon as the linear hull effect increases. In other words, and independently of Murphy's observation, averaging the linear bias $LB(\mathbf{a}, \mathbf{b}; K)$ over the keys is not very significant in this case. In [34, 45], the authors assume that the bias ϵ of a fixed linear approximation under variable key follows a normal distribution with mean zero. In our experiments, we evaluated the distribution of the bias of several approximations under up to 65536 keys. Our results suggest that the distribution converges to a normal distribution when the number of rounds increases, just as the linear

hull effect becomes increasingly important. In order to test the normality of the distribution, we used three different approaches:

- A simple histogram plot of the experimental bias,
- A Q-Q plot comparing the experimental distribution with a normal distribution by plotting their quantiles against each other,
- A *Jarque-Bera* test measuring the departure from normality.

Both approaches suggests that the distribution becomes close to normal after 5-6 rounds, when the encryption enters the *stationary area* (see section 3). An illustration of the histogram and Q-Q plot approaches is given in figure 13 (see appendix E). According to our experiments, the mean value converges towards zero when the number of rounds increases.

On the other hand, Figure 5 shows that once a block cipher has a sufficient number of rounds for this linear hull effect to be dominant, the best linear approximations computed for each key have a minimum, maximum and average bias that does not strongly depend on the keys. Intuitively, once the number of rounds in the cipher is sufficient, changing the key is equivalent to adding one round. And the resulting cipher is hard to distinguish from a 16-bit random S-box, of which the maximum, minimum and average biases essentially depend on the block size. In order to confirm this intuition, we ran a last experiment in which we directly computed the distribution of the biases $LB(\mathbf{a}, \mathbf{b}; K)$ for independent round keys. The results in Figure 6 show that after a few rounds, the shape of this distribution does not change anymore, and is in fact hard to distinguish from the one of a random 16-bit S-box, as theoretically expected.

3 Test of key dependent sampled biases

The previous section discussed experiments related to the best selection of linear characteristics and their differences with linear approximations. In this section, we complement this evaluation by looking at the exploitation of these characteristics in a linear cryptanalysis. For this purpose, we will investigate how the two procedures to test sampled linear biases described in Section 1.4 perform with different linear characteristics and approximations. In addition, we will consider an extension of linear cryptanalysis using multiple linear approximations, which is a frequently considered solution to decrease the data complexity of the attack, first introduced in [25]. For this purpose, we will assume an adversary who has determined the set of m best characteristics of an $R-1$-round cipher. Then, we define the gain of a linear cryptanalysis attack as proposed in [5]:

Definition 7. *If an attack is used to recover an n-bit key and is expected to return the correct key after having checked on the average M candidates in the sorted list, then the gain of the attack, expressed in bits, is defined as:*

$$\gamma = -\log_2 \frac{2 \cdot M - 1}{2^n}. \tag{17}$$

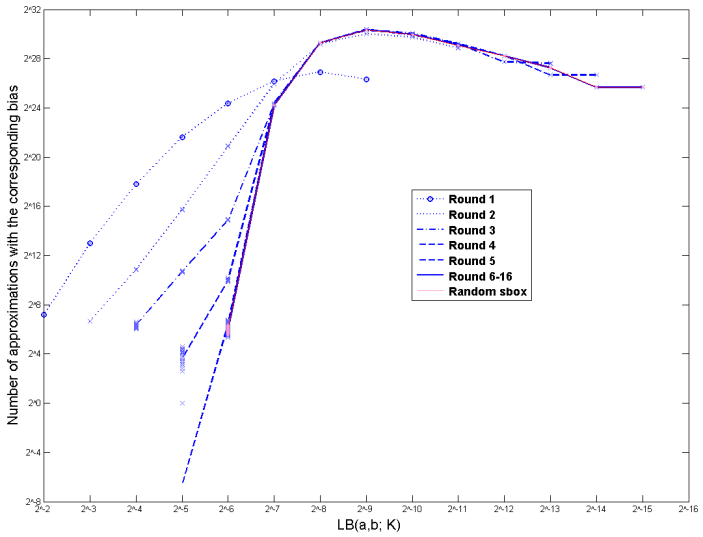

Fig. 6: Distributions of the linear biases for different number of rounds.

For any set of approximation, we also use Biryukov et al.'s definition of capacity:

$$C(\mathbf{a}^{1:m}, \mathbf{b}^{1:m}; K) = 4 \cdot \sum_{i=1}^{m} LB(\mathbf{a}^i, \mathbf{b}^i; K)^2 \tag{18}$$

In [5], the data complexity of a linear cryptanalysis using multiple linear approximations is estimated with[6] $1/C(\mathbf{a}^{1:m}, \mathbf{b}^{1:m}; K)$. Finally, when considering multiple approximations, the test of key-dependent sampled biases is generalized as follows. First, the maximum likelihood key testing of Equation (11) becomes:

$$\tilde{k_r} = \underset{k_r^*}{\operatorname{argmin}} \sum_{i=1}^{m} \left(|SLB(\mathbf{a}_1^i, \mathbf{a}_R^i; k_r^*)| - |LCB(\Omega^i, \tilde{K}|) \right)^2 . \tag{19}$$

It corresponds to a straightforward extension of Equation (11), where an absolute difference operator is added, in order to remove the need to guess the parities for multiple approximations (which becomes computationally hard as m increases). Next, the heuristic key testing procedure of Equation (12) becomes:

$$\tilde{k_r} = \underset{k_r^*}{\operatorname{argmax}} \left(\sum_{i=1}^{m} \left| SLB(\mathbf{a}_1^i, \mathbf{a}_R^i; k_r^*) \right| \right) . \tag{20}$$

[6] As detailed in [42], this can lead to an overestimate of the attack gain, because of an argument similar to the one discussed in Section 2.2. However, as will be detailed next, the main limitation of this estimated data complexity is due to the imprecise knowledge of the linear biases in actual linear cryptanalysis.

In practice, we performed systematic experiments against 4-round, 6-round and 8-round of Small-Present, corresponding to contexts where the linear hull effect is negligible, starts to play a role and is eventually dominant. For each of these contexts, we computed the gain of different linear cryptanalysis attacks using multiple approximations, with $m = 2^i$ and $i \in [0; 13]$. For clarity purpose, the results of these experiments have been reported in Appendices A, B, C, D. They correspond to four different scenarios that we now detail.

Best characteristics generated with a branch-and-bound algorithm. This scenario, of which the results are given in Appendix A, Figure 9, is the most realistic one, as it corresponds to the only strategy applicable by an actual adversary who does not know the key and has limited computational power to find good characteristics. The figure confirms two important intuitions. First, the maximum likelihood approach only works as long as the linear characteristic bias $LCB(\Omega, \tilde{K})$ is a good approximation of the sampled linear bias $SLB(\mathbf{a}, \mathbf{b}, 1, N_p; K)$. But as soon as the linear hull effect appears (i.e. for 6 rounds in our experiments), it does not allow reaching high gains anymore. Second, the heuristic approach is successful for a few more rounds than the maximum likelihood one (i.e. up to 6 rounds in our experiments), as it does not require such a precise estimations of the bias. But when increasing the number of rounds, the best characteristics eventually behave as random approximations, as pointed out in Section 2.4, Figure 4. Hence, maximizing the sampled linear bias does not allow recovering secret information anymore in this case.

Best $(R-1)$-round approximations. First, let us mention again that this (and the next) scenario(s) do not correspond to realistic adversaries, as finding the best $(R-1)$-round approximations and having precise estimations of their biases is usually not possible for practically secure ciphers. However, as the previous experiments suggest that having a good estimation of the linear biases is critical for the success of a linear cryptanalysis, it is interesting to observe the behavior of the key testing procedures in an artificial context, where the adversary knows these best $(R-1)$ approximations and their exact bias, for each key. The results of our experiments in this case are plotted in Appendix B, Figure 10, and highlight the following facts. First, the gains of the two key ranking procedures is higher than when using characteristics and remains high even when increasing the number of rounds. Second, the impact of using multiple approximations is stronger when applying the heuristic key ranking. This is in fact directly related to the results of Section 2.1, Figure 2: for the sampled linear bias $SLB(\mathbf{a}, \mathbf{b}, 1, N_p; K)$ to be close to the actual bias $LB(\mathbf{a}, \mathbf{b}; K)$, we need to sample with a number of plaintexts N_p that is close to the full codebook. Hence, the maximum likelihood key ranking strategy only starts to extract information when N_p is large enough, while a heuristic key ranking is less sensitive to this need of good estimates for $LB(\mathbf{a}, \mathbf{b}; K)$. Note that, as discussed in [10], this last observation is particularly strong when using Matsui's second algorithm, because of the partial decryption process which maps the sampled biases at

round R towards sampled biases at round $R-1$, with a non-linear S-box. It would be partially relaxed if Matsui's first algorithm was considered.

Random ($R-1$)-round approximations. The previous experiments suggest that the main parameter allowing a successful linear cryptanalysis with a maximum likelihood key ranking is the knowledge of a good estimated bias. In order to confirm this intuition, we launched another set of experiments, with randomly selected input and output masks, for which the exact value of $LB(\mathbf{a}, \mathbf{b}; K)$ was provided to the adversary. As illustrated in Appendix C, Figure 11, this context leads to gains for the maximum likelihood approach that are very close to the ones obtained with the best approximations in the previous paragraph. By contrast, the heuristic key ranking is totally ineffective in this case as the random masks selected do not correspond to maximum linear bias values.

($R-1$)-round approximations with null bias. Eventually and for completeness, we launched a set of experiments with approximations having null bias. Figure 12 in Appendix D shows that this scenario can also lead to successful key recoveries. Interestingly, it can even work with a modified heuristic approach, provided that one selects the key candidate that minimizes the sampled linear bias (rather than the one maximizing it). Hence, attacks exploiting such approximations could actually be mounted against actual ciphers and constitute a counterpart to the use of impossible differentials in cryptanalysis [4].

We conclude this section with two additional observations.

1. A note on the estimated data complexity of linear cryptanalysis with multiple approximations. As mentioned in footnote 4, the estimation of the gain in function of the data complexity in a linear cryptanalysis using multiple approximations is a difficult problem. One important issue is that most estimates published in the literature, e.g. [5], are based on a maximum likelihood key ranking procedure. But as witnessed by the previous experiments, such an approach is hardly applicable as soon as the linear hull effect in the cipher increases. Unfortunately, this is also the context in which exploiting multiple approximations would be useful in practice, in order to compensate the limited gain of a single approximation. In order to quantify the impact of multiple approximations in linear cryptanalysis, we additionally plotted the gain of attacks based on a heuristic key ranking, using the best approximations (as in Appendix B), in function of the product between the number of plaintexts used in the attack N_p and the capacity of the set of approximations $C(\mathbf{a}^{1:m}, \mathbf{b}^{1:m}; K)$, in Figure 7. If the data complexity was properly estimated with $1/C(\mathbf{a}^{1:m}, \mathbf{b}^{1:m}; K)$, the 14 curves in Figure 7 should be identical. For example, in cases where all the approximations have the same linear bias, it would imply that doubling the number of approximations in the attack would be equivalent to doubling the number of plaintexts, which is clearly not the case in our experiments. On the other hand, the figure also shows that we need approximately $2^5/c$ plaintexts to reach the maximum gain with the best approximation. And we need $2^{10}/c$ plaintexts to

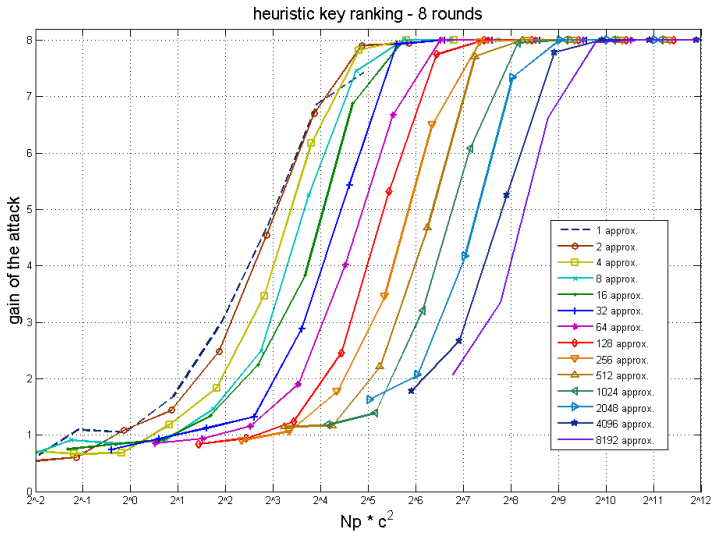

Fig. 7: Gain of linear cryptanalysis using multiple approximations with Matsui's second algorithm, in function of the number of texts N_p multiplied by the capacity $C(\mathbf{a}^{1:m}, \mathbf{b}^{1:m}; K)$, using the 1st to 8192th best approximations generated from the full codebook, for 8-round PRESENT (maximum likelihood key ranking).

reach this gain with all the 2^{13} approximations. If the use of multiple approximations was completely ineffective, we would need $2^{18}/c$ plaintexts in this case (i.e. we would have a factor 2 between each of the curves in Figure 7). Summarizing, the improved data complexity of linear cryptanalysis attacks using multiple approximations falls between these two extremes. In this respect, we note again that these observations are mainly due to the use of Matsui's second algorithm with a heuristic key ranking procedure. But as discussed in [10], experiments performed with Matsui's first algorithm have a data complexity that is much closer to their estimates with $1/C(\mathbf{a}^{1:m}, \mathbf{b}^{1:m}; K)$.

2. A note on the stationary area for block ciphers. Let us finally shortly comment on the notion of "stationary area" with respect to linear and differential cryptanalysis, discussed in [46]. Intuitively, a block cipher is in its stationary area if the distribution of its linear biases (and differential probabilities) do not vary anymore with the number of rounds. This typically corresponds to the point where adding more rounds to the block cipher is not useful anymore from the point of view of statistical attacks. As pictured in Figure 6, Small-Present-[16] becomes stationary after 6 rounds. This should imply that targeting any number of rounds larger than 6 with a given statistical attack should lead to similar

gains. For illustration, we plotted in Figure 8 the gains of attacks against 8 and 19 rounds of Small-Present-[16], with the best linear approximations. It clearly illustrates that attacks have very similar data complexities (experiments could also be launched with the best characteristics, with random approximations or approximations with zero bias and would lead to identical observations).

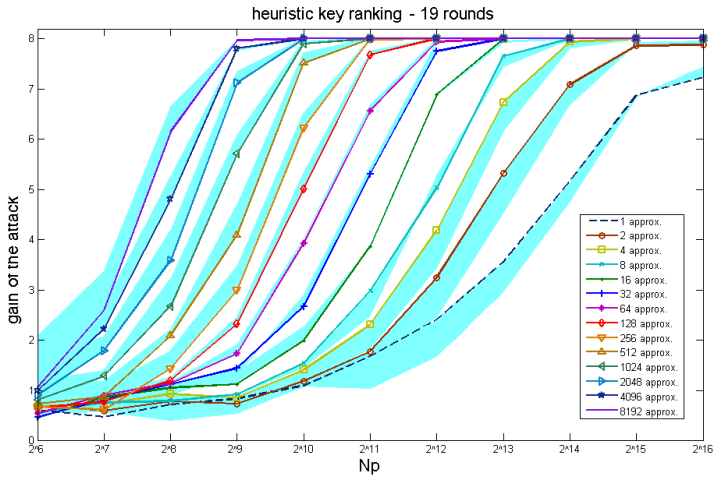

Fig. 8: Gain of linear cryptanalysis using multiple approximations with Matsui's 2nd. algorithm, in fct. of the number of texts N_p, using the 1st to 8192th best approx. generated from the full codebook, for 19-round PRESENT (maximum likelihood key ranking). Shadowed curves show the difference with the results against 8-round PRESENT).

4 Beyond linear cryptanalysis

Many variations of linear cryptanalysis have been proposed in the literature. While they essentially rely on the same principles, they introduce tweaks that allow improving the effectiveness of the attack. In this section, we briefly survey some important results and their relations with our previous experiments[7].

Next to the use of non-linear approximations [30] or chosen plaintexts attacks [31], one of the most frequently investigated ways to extend linear cryptanalysis is to take advantage of non-uniform distributions in the plaintexts or ciphertexts (rather than simply biased linear approximations). This idea was first denoted as partitioning cryptanalysis [18] and directly leads to the question of how to find these non-uniform behaviors in a cipher. Hence, in practice, it usually relies on

[7] A number of these advanced strategies are analyzed carefully in the rest of this book.

some specificities that a cryptanalysis may find, e.g. [16, 40]. Partitioning cryptanalysis has been analyzed by Baignères et al. in [1]. More recently, the idea of multidimensional cryptanalysis [19] and its application to linear cryptanalysis using multiple approximations [21] was based on very similar ideas. Quite naturally, these extensions suffer from the same limitations as linear cryptanalysis regarding the difficulty of obtaining precise estimates of the target distributions. In fact, the problem of estimating multidimensional distributions is generally more difficult than the one of estimating linear biases. Hence, these optimal attacks also need to be modified with heuristics when statistical hull effects appear. In this respect, we finally mention statistical saturation attacks as a typical example of such heuristics [11]. This attack exploits the non-uniform behavior of the diffusion layer in the block cipher PRESENT in a chosen plaintext scenario. It can be viewed as a particular case of partitioning (or multidimensional) cryptanalysis in which one simply selects the key candidate that maximizes the distance with a uniform distribution over the m bits targeted in the attack, in order to avoid the need of precise estimated distributions in the key ranking.

Conclusion

This chapter essentially highlights the difficulty of predicting and analyzing the statistical behavior of a block cipher as its number of rounds increases, both for adversaries trying to exploit key-dependent biases, and for designers trying to accurately predict security bounds. As a consequence, from these two complementary points of view, the main open problem is to find efficient solutions for estimating the target distributions in a statistical cryptanalysis. Progresses in this direction would allow to improve our understanding of provable security against such attacks and to close the gap with the practical security approach that is currently (and successfully) used for designing ciphers.

References

1. T. Baignres, P. Junod, S. Vaudenay, *How Far Can We Go Beyond Linear Cryptanalysis?*, in the proceedings of ASIACRYPT 2004, LNCS, vol 3329, pp 432-450, Jeju Island, Korea, December 2004.
2. T. Baignères, M. Finiasz, *Dial C for Cipher*, in the proceedings of SAC 2006, LNCS, vol 4356, pp 76-95, Montreal, Canada, August 2006.
3. T. Baignères, M. Finiasz, *KFC - The Krazy Feistel Cipher*, in the proceedings of ASIACRYPT 2006, LNCS vol 4284, pp 380-395, Shanghai, China, December 2006.
4. E. Biham, A. Biryukov, A. Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials*, in the proceedings of EUROCRYPT 1999, LNCS, vol 1592, pp 12-23, Prague, Czech Republic, May 1999.
5. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of Crypto 2004, LNCS, vol 3152, pp 1-22, Santa Barbara, California, USA, August 2004.

6. C. Blondeau, B. Gérard, *On the Data complexity of Statistical Attacks Against Block Ciphers*, in the proceedings of the IMA International Conference on Cryptography and Coding 2009, full version available on the IACR ePrint archive: http://eprint.iacr.org/2009/064,
7. C. Blondeau, B. Gérard, *Links Between Theoretical and Effective Differential Probabilities: Experiments on PRESENT*, in the proceedings of the workshop in Tools for Cryptanalysis, pp 109-125, Edgham, UK, June 2010. full version available on the IACR ePrint archive: http://eprint.iacr.org/2010/261.
8. B.Collard, F.X. Standaert, J.J. Quisquater, *Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent*, in the proceedings of InsCrypt 2007, LNCS, vol 4990, pp 51-65, Xining, China, September 2007.
9. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improving the Time Complexity of Matsui's Linear Cryptanalysis*, in the proceedings of ICISC 2007, LNCS, vol 4817, pp 77-88, Seoul, Korea, November 2007.
10. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent*, in the proceedings of FSE 2008, LNCS, vol 5086, pp 382-397, Lausanne, Switzerland, February 2008.
11. B. Collard, F.-X. Standaert, *A Statistical Saturation Attack against the Block Cipher PRESENT*, in the proceedings of CT-RSA 2009, LNCS, vol 5473, pp 195-210, San Francisco, USA, April 2009.
12. J. Daemen, V. Rijmen, *The Wide Trail Design Strategy*, in the proceedings of the IMA International Conference on Cryptography and Coding 2001, LNCS, vol 2260, pp 222-238, Cirencester, UK, December 2001.
13. J. Daemen, V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*, Springer 2002.
14. J. Daemen, V. Rijmen, *Probability Distributions of Correlation and Differentials in Block Ciphers*, Journal of Mathematical Cryptology, vol 1, pp 12-35, 2007.
15. J. Daemen, R. Govaerts, J. Vandewalle, *Correlation Matrices*, in the proceedings of FSE 1994, LNCS, vol 1008, pp 275-285, Leuven, Belgium, December 1994.
16. H. Gilbert, H. Handschuh, A. Joux, S. Vaudenay, *A Statistical Attack on RC6*, in the proceedings of FSE 2000, LNCS, vol 1978, pp 64-74, New York, USA, April 2000.
17. C. Harpes, G.G. Kramer, J.L. Massey, *A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-Up Lemma*, in the proceedings of Eurocrypt 1995, LNCS, vol 921, pp 24-38, Saint-Malo, France, May 1995.
18. C. Harpes, J.L. Massey, *Partitioning Cryptanalysis*, in the proceedings of FSE 1997, LNCS, vol 1267, pp 13-27, Haifa, Israel, January 1997.
19. M. Hermelin, J.Y. Cho, K. Nyberg, *Multidimensional Extension of Matsui's Algorithm 2*, in the proceedings of FSE 2009, LNCS, vol 5665, pp 209-227, Leuven, Belgium, February 2009.
20. M. Hermelin, J.Y. Cho, K. Nyberg, *Multidimensional Linear Cryptanalysis of Reduced Round Serpent*, in the proceedings of ACISP 2008, LNCS, vol 5107; pp 203-215, Wollollong, Australia, July 2008.
21. M. Hermelin, K. Nyberg, *Dependent Linear Approximations: The Algorithm of Biryukov and Others Revisited*, in the proceedings of CT-RSA 2010, LNCS, vol 5985, pp 318-333, San Franciscon California, USA, February 2010.
22. P. Junod, *On the Complexity of Matsui's Attack*, in the proceedings of SAC 2001, LNCS, vol 2259, pp 199-211, Toronto, Ontario, August 2001.
23. P. Junod, *On the Optimality of Linear, Differential, and Sequential Distinguishers*, in the proceedings of Eurocrypt 2003, LNCS, vol 2656, pp 17-32, Warsaw, Poland, May 2003.

24. P. Junod, S. Vaudenay, *Optimal Key Ranking Procedures in a Statistical Cryptanalysis* in the proceedings of FSE 2003, LNCS, vol 2887, pp 235-246, Lund, Sweden, February 2003.
25. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of CRYPTO 1994, LNCS, vol 839, pp. 26-39, Santa Barbara, California, USA, August 1994.
26. L. Keliher, H. Meijer, S.E. Tavares, *New Method for Upper Bounding the Maximum Average Linear Hull Probability for SPNs*, in the proceedings of Eurocrypt 2001, LNCS, vol 2045, pp 420-436, Innsbruck, Austria, May 2001.
27. L. Keliher, H. Meijer, S.E. Tavares, *Improving the Upper Bound on the Maximum Average Linear Hull Probability for Rijndael*, in the proceedings of SAC 2001, LNCS, vol 2259, pp 112-128, Toronto, Ontario, Canada, August 2001.
28. L. Keliher, *Linear Cryptanalysis of Substitution-Permutation Networks*, PhD Thesis, Queen's University, Kingston, Ontario, Canada, October 2003.
29. L.R. Knudsen, *Practically Secure Feistel Ciphers*, in the proceedings of FSE 1993, LNCS, vol 809, pp 211-221, Cambridge, UK, December 1993.
30. L.R. Knudsen, M.J.B. Robshaw, *Non-Linear Approximations in Linear Cryptoanalysis*, in the proceedings of EUROCRYPT 1996, LNCS, vol 1070, pp 224-236, Saragossa, Spain, May 1996.
31. L.R. Knudsen, J.E. Mathiassen, *A Chosen-Plaintext Linear Attack on DES*, in the proceedings of FSE 2000, LNCS, vol 1978, pp 262-272, New York, USA, April 2000.
32. L.R. Knudsen, J.E. Mathiassen, *On the Role of Key Schedules in Attacks on Iterated Ciphers*, in the proceedings of ESORICS 2004, LNCS, vol 3193, pp 322-334, Sophia Antipolis, France, September 2004.
33. G. Leander, *Small Scale Variants of the Block Cipher PRESENT*, Cryptology ePrint Archive, Report 2010/143, http://eprint.iacr.org/, 2010.
34. G. Leander, *On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN*, to appear in the proceedings of Eurocrypt 2011, Tallinn, Estonia, May 2011.
35. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of Eurocrypt 1993, LNCS, vol 765, pp 386-397, Lofthus, Norway, May 1993.
36. M. Matsui, *The First Experimental Cryptanalysis of the Data Encryption Standard*, in the proceedings of Crypto 1994, LNCS, vol 839, pp 1-11, Santa Barbara, California, USA, August 1994.
37. M. Matsui, *On Correlation Between the Order of S-boxes and the Strength of DES*, in the proceedings of Eurocrypt 1994, LNCS, vol 950, pp 366-375, Perugia, Italy, May 1994.
38. M. Matsui, *New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis*, in the proceedings of FSE 1996, LNCS, vol 1039, pp 205-218, Cambridge, IK, February 1996.
39. M. Matsui, *New Block Encryption Algorithm MISTY*, in the proceedings of FSE 1997, LNCS, vol 1267, pp 54-68, Haifa, Israel, January 1997.
40. M. Minier, H. Gilbert, *Stochastic Cryptanalysis of Crypton*, in the proceedings of FSE 2000, LNCS, vol 1978, pp 121-133, New York, USA, April 2000.
41. S. Murphy, *The Effectiveness of the Linear Hull Effect*, Royal Holloway University of London, Technical Report RHUL-MA-2009-19, 2009.
42. S. Murphy, *Overestimates for the Gain of Multiple Linear Approximations*, Royal Holloway University of London, Technical Report RHUL-MA-2009-21, 2009.
43. K. Nyberg, *Linear Approximations of Block Ciphers*, in the proceedings of Eurocrypt 1994, LNCS, vol 950, pp 439-444, Perugia, Italy, May 1994.

44. K. Nyberg, L.R. Knudsen, *Provable Security against Differential Cryptanalysis*, Journal of Cryptology, vol 8, num 1, pp 27-37, 1995.
45. K. Ohkuma, *Weak Keys of Reduced Round PRESENT for Linear Cryptanalysis*, in the proceedings of SAC 2009, LNCS, vol 5867, pp 249-265, Calgary, Alberta, Canada, August 2009.
46. G. Piret, F.-X. Standaert, *Provable Security of Block Ciphers Against Linear Cryptanalysis - a Mission Impossible?*, Designs, Codes and Cryptography, vol 50, num 3, pp 325-338, March 2009.
47. V. Rijmen, *Cryptanalysis and Design of Iterated Block Ciphers*, PhD Thesis, Katholieke Universiteit Leuven, Belgium, October 1997.
48. T. Ritter, *Measuring boolean function nonlinearity by Walsh transform*, `http://www.ciphersbyritter.com/ARTS/MEASNONL.HTM`.
49. G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-D. Legat, *Efficient Uses of FPGAs for Implementations of DES and Its Experimental Linear Cryptanalysis*, IEEE Transactions on Computers, vol 52, num 4, pp 473-482, 2003.
50. A.A. Selçuk, *On Bias Estimation in Linear Cryptanalysis*, in the proceedings of Indocrypt 2000, LNCS, vol 1977, pp 52-66, Calcutta, India, December 2000.
51. A.A. Selçuk, *On Probability of Success in Linear and Differential Cryptanalysis*, Journal of Cryptology, vol 21, num 1, pp 131-147, 2008.
52. S. Vaudenay, *Decorrelation: A Theory for Block Cipher Security*, Journal of Cryptology, vol 16, num 4, pp 249-286, 2003.

A Best characteristics

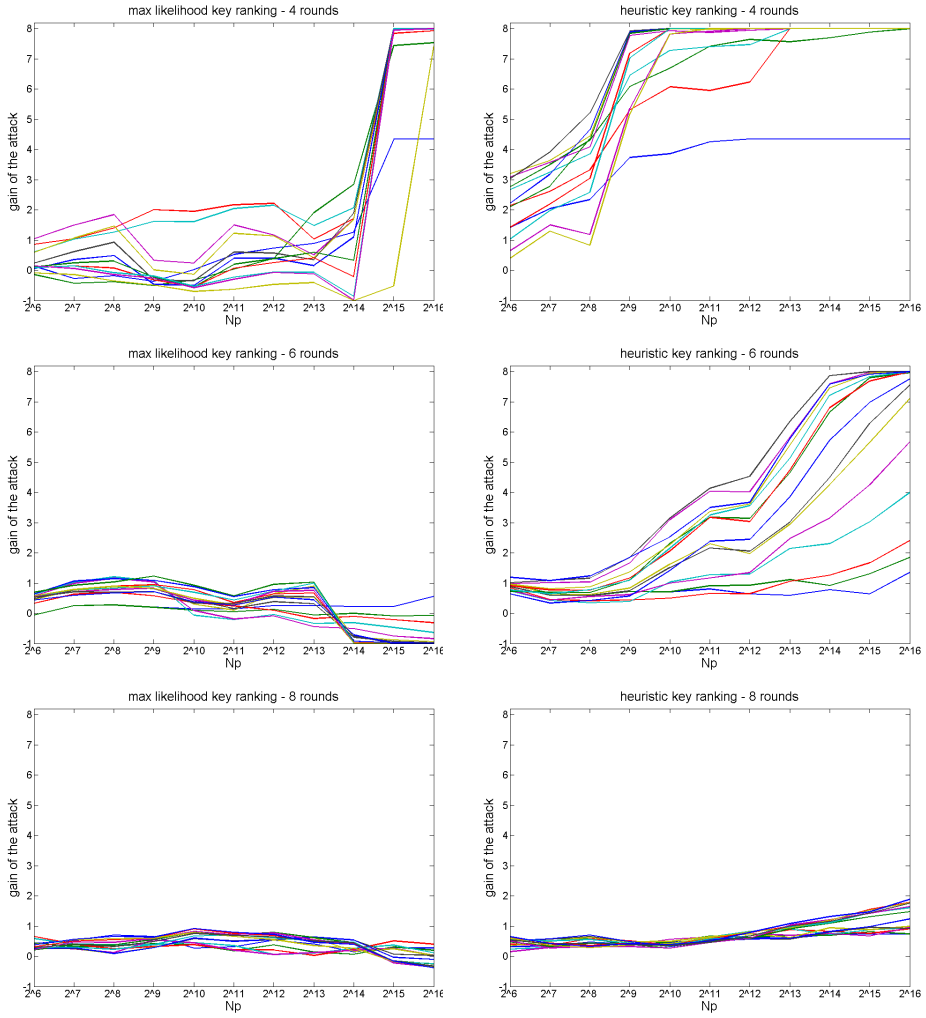

Fig. 9: Gain of linear cryptanalysis using multiple approximations with Matsui's second algorithm, in function of the number of texts N_p, using 1 to 8192 characteristics generated with a branch-and-bound algorithm, for 4-round, 6-round and 8-round PRESENT. Left: maximum likelihood key ranking. Right: heuristic key ranking.

B Best approximations

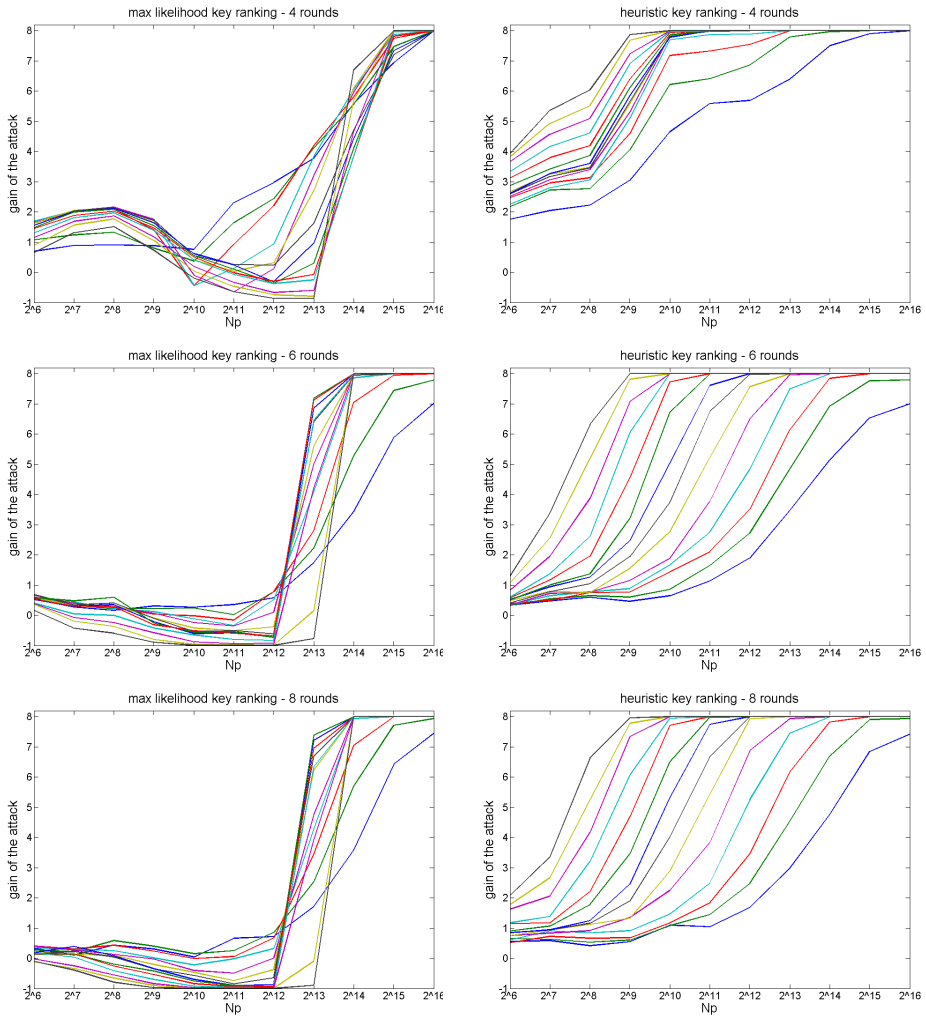

Fig. 10: Gain of linear cryptanalysis using multiple approximations with Matsui's second algorithm, in function of the number of texts N_p, using the 1st to 8192th best approximations generated from the full codebook, for 4-round, 6-round and 8-round PRESENT. Left: maximum likelihood key ranking. Right: heuristic key ranking.

C Random approximations

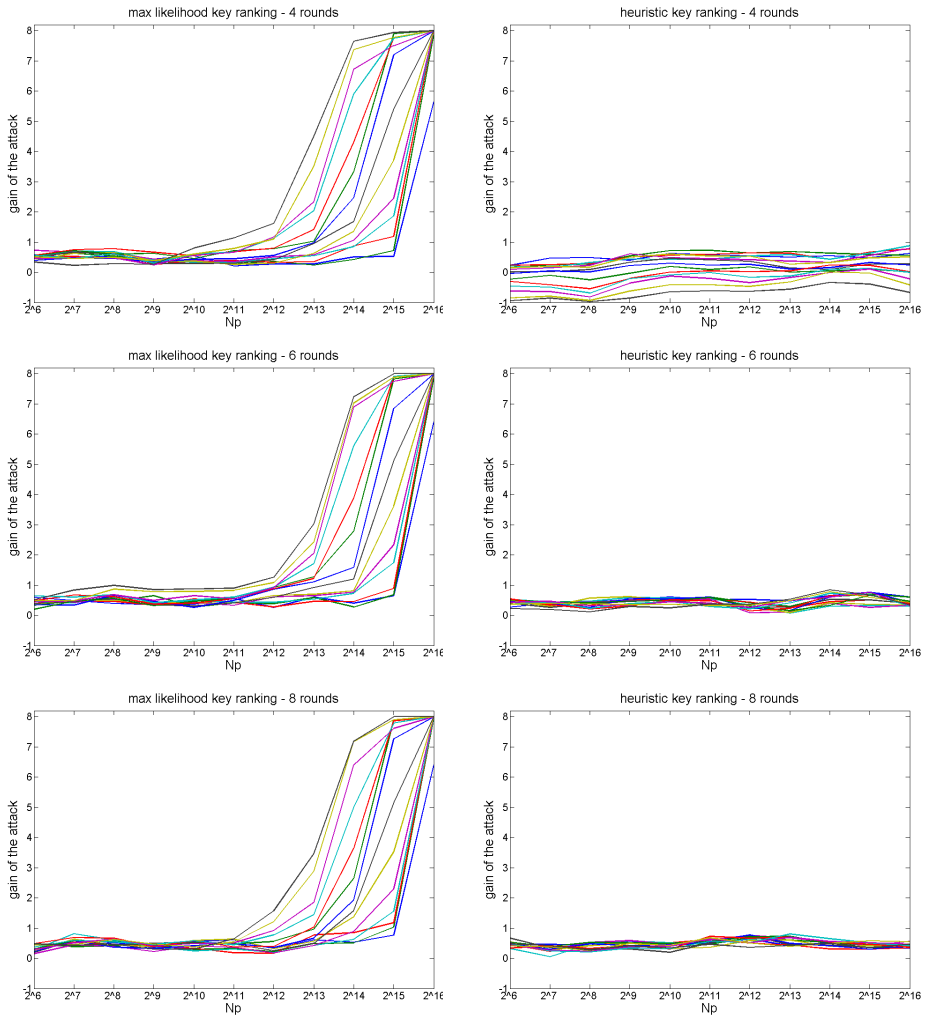

Fig. 11: Gain of linear cryptanalysis using multiple approximations with Matsui's second algorithm, in function of the number of texts N_p, using 1 to 8192 random approximations with biases estimated from the full codebook, for 4-, 6- and 8-round PRESENT. Left: maximum likelihood key ranking. Right: heuristic key ranking.

D Approximations with null bias

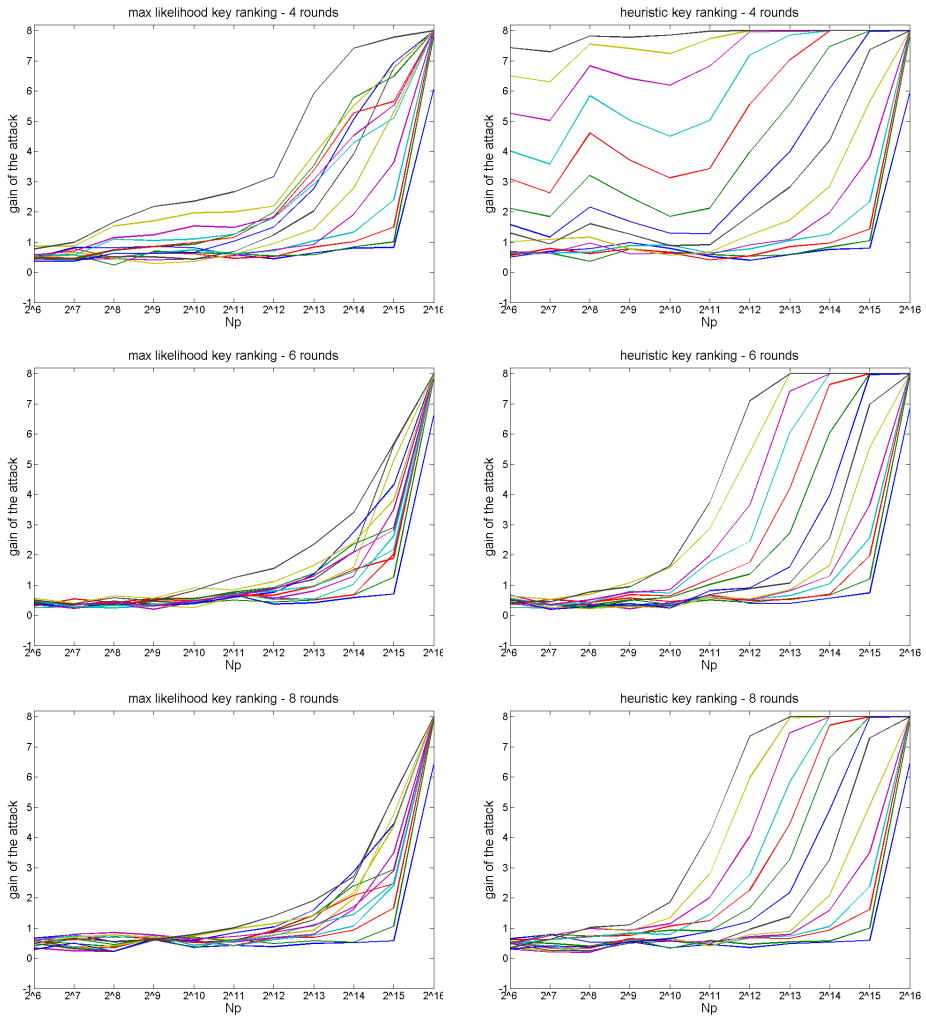

Fig. 12: Gain of linear cryptanalysis using multiple approximations with Matsui's second algorithm, in function of the number of texts N_p, using 1 to 8192 approximations with null bias estimated from the full codebook, for 4-round, 6-round and 8-round PRESENT. Left: maximum likelihood key ranking. Right: heuristic key ranking.

E Bias distribution for fixed approximation and variable key

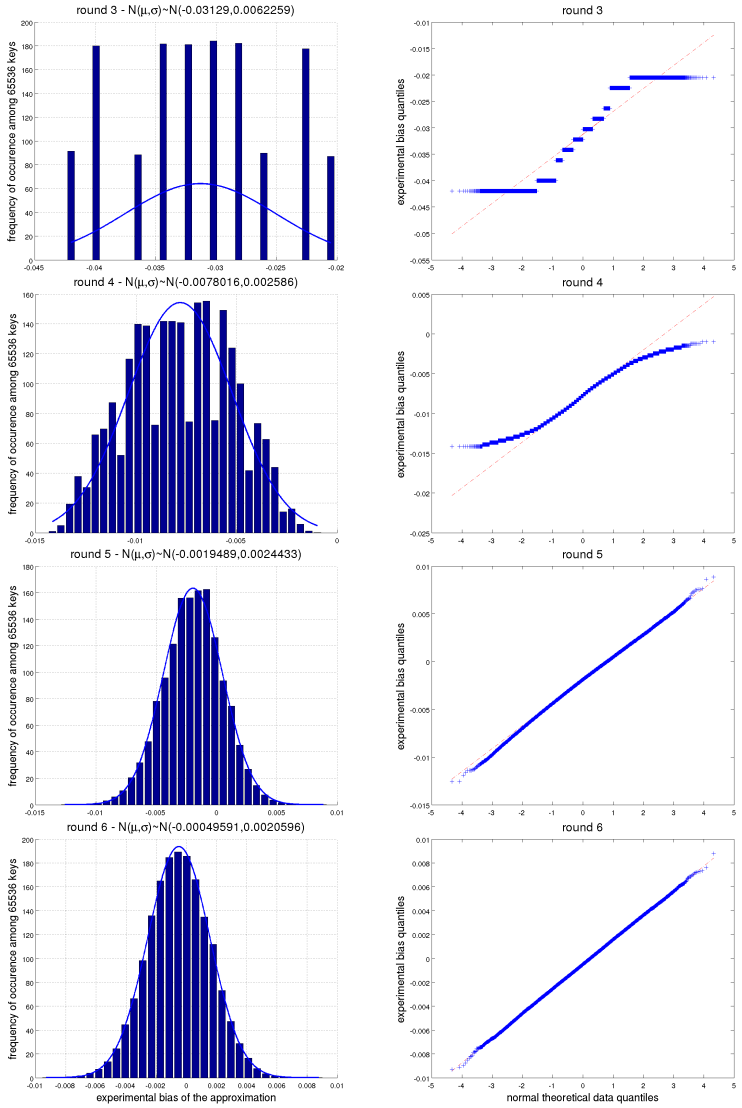

Fig. 13: Experimental distribution of the bias of an approximation on 3-round, 4-round, 5-round and 6-round PRESENT, estimated using 65536 different keys and the full codebook. Left: Histogram of the distribution. Right: Q-Q plot comparing the quantiles of the experimental distribution with the quantiles of a normal distribution.

CONCLUSION

CONCLUSION

The research in this thesis was mainly focused on the experimental evaluation of block cipher cryptanalysis and its assumptions. Individual contributions, described in the previous chapters first include the development of efficient tools for the generation of linear approximations and an FFT-based trick allowing one to speedup the partial key decryptions involved in most statistical attacks. Next, we used these tools for improving the cryptanalysis of the block cipher PRESENT. By introducing an attack inspired by partitioning cryptanalysis, we have put forward a weakness in the diffusion layer of this low cost cipher, that can be effectively exploited by an adversary. Attacks against reduced-round versions of the cipher experimentally proved to lead to significant gains up to 15 rounds (out of 31), using 2^{36} chosen plaintexts (out of 2^{64}). Theoretical aspects of this attack were also investigated in Chapter 7: while the precise estimation of the attack data complexity and gain turned out to be computationally intensive (e.g. using Markov transition matrices), we showed that a combination of heuristics could be used to explain the success of the attack for a limited number of rounds. Finally, the open questions raised by our Statistical Saturation Attack, together with the ongoing discussions on the linear hull effect, led us to re-consider the basic assumptions in cryptanalysis. In the final part of this thesis, we discussed and illustrated experimentally the relevance of certain important assumptions in cryptanalysis and block cipher design techniques, namely: the (so-called) practical and provable security approaches to prevent linear cryptanalysis, the key independence and key equivalence assumptions and the linear hull effect. Next to these individual contributions, our different pieces of work allowed us to put forward some methodological conclusions, by highlighting the pros and cons of different approaches, and to underline a number of important challenges for further research that we now detail.

In the first place, our results exhibit a tension between practical security and provable security against linear cryptanalysis. More precisely, as the estimation of a linear attack becomes computationally intensive with the number of block cipher rounds, previous techniques to argue about security in a provable way (i.e. trying to take linear hulls into account) had to rely on simplifying assumptions, e.g. by averaging the probability of success of the attacks over the key. But our experiments clearly illustrate that one important reason for practical security in actual block ciphers is the key-dependency of the best

linear approximations (and, more generally, statistical features) that becomes important when the number of rounds increases. On the positive side, this makes a case for the practical security approach, as such a variance over the keys typically appears when no single characteristic can be used to explain the experimental biases (hence, when the data complexity estimated by single characteristics becomes prohibitive). On the negative side, this raises important questions about the minimum number of rounds needed to prevent statistical attacks. Also, the existence of key dependencies for the best approximations in a block cipher raises the question whether there exist classes of keys for which a given approximation would allow successful attacks with a high probability and large number of rounds, i.e. classes of keys that are weak for a given linear hull. This approach has been applied with some success on the block cipher PRESENT [4]. A preliminary to answer these difficult questions would be to extend the computation of the probability transition matrices of a small block cipher, in order to look for such key classes experimentally. Interestingly, such seemingly theoretical questions also have very concrete consequences: answering them would allow to determine how large can a block cipher key be, if the block size is fixed. In practice, one solution to extend the power of the experimental approach could also be to consider hardware implementations of statistical attacks. Platforms such as the COPACOBANA code-breaker developed at the university of Bochum [3] would allow to gain orders of magnitude in the data complexities that one could explore (compared to the software-based approach considered in this thesis).

The adversarial counterpart of these questions about the linear hull relates to the best strategies to perform linear attacks. As discussed in this thesis, applying optimal solutions, based on maximum likelihood key recovery, usually requires that the adversary has precise estimations of the statistical biases. But these statistical biases are as difficult to find for the adversary as they are to estimate for the designer. As a consequence, our experiments showed that in a number of practical contexts, attacks based on heuristics can perform better than maximum likelihood ones, because of imperfect bias estimations. This difference can be related to the difference between profiled and non profiled attacks when using side-channels such as the power consumption. As precise models for a device's power consumption may not be available to the adversary, it frequently turns out that suboptimal (non-profiled) attacks perform better in practice. Back in the context of black box (statistical) cryptanalysis, this boils down to the question of finding the best heuristics in a given scenario. An interesting approach for the cryptanalysis of block ciphers with secret components is presented in [1].

Finally, taking these observations and questions to a more general level, one important challenge in block cipher cryptanalysis is to move towards a better formalization of the attacks. The present state-of-the-art (and this thesis in particular) are frequently based on an engineering toolbox, both for the crypt-

analysis and design of block ciphers. Unifying views of certain classes of attacks (e.g. as attempted by David Wagner in [6] and by Chabaud and Vaudenay in [2]) would help to clarify the design criteria to consider in the future. Relations between similar (e.g. statistical) attacks, such as statistical saturation attacks and multidimensional cryptanalaysis, would certainly be useful as well. Quite naturally, the investigation of security bounds against these classes of attacks, such as attempted by the decorrelation theory [5], is another interesting scope of research. In this respect, it is worth to notice that existing ciphers based on the decorrelation theory are relatively inefficient (compared to ciphers such as PRESENT). Hence, designing a low cost cipher following the principles of decorrelation theory would be an interesting topic of research too. Last, let us mention the question of the block cipher S-box size and their impact for (mainly algebraic) cryptanalysis. While large S-boxes are desirable for security reasons, small ones allow much more efficient implementations, in particular if one also has to consider physical security against side-channel or fault attacks. So it would be interesting to investigate if smaller non-linear S-boxes (e.g. 3-bit ones) can always be traded for more block cipher rounds, and to which extent.

BIBLIOGRAPHY

[1] Julia Borghoff, Lars R. Knudsen, Gregor Leander, and Soeren S. Thomsen. Cryptanalysis of present-like ciphers with secret s-boxes. Cryptology ePrint Archive, Report 2010/574, 2010. `http://eprint.iacr.org/`.

[2] Florent Chabaud and Serge Vaudenay. Links between differential and linear cryptoanalysis. In *EUROCRYPT*, pages 356–365, 1994.

[3] Sandeep Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. Breaking ciphers with copacobana - a cost-optimized parallel code breaker. In *CHES*, pages 101–118, 2006.

[4] Kenji Ohkuma. Weak keys of reduced-round present for linear cryptanalysis. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 249–265. Springer, 2009.

[5] Serge Vaudenay. Decorrelation: A theory for block cipher security. *J. Cryptology*, 16(4):249–286, 2003.

[6] David Wagner. Towards a unifying view of block cipher cryptanalysis. In *FSE*, pages 16–33, 2004.

PUBLICATION LIST

[1] B. Collard and F.-X. Standaert. Experimenting linear cryptanalysis. to appear in Advanced Linear Cryptanalysis, IOS Press, 2011.

[2] Baudoin Collard and François-Xavier Standaert. A Statistical Saturation Attack on the Block Cipher PRESENT. In *CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 4 2009.

[3] Baudoin Collard and François-Xavier Standaert. Multi-Trail Statistical Saturation Attacks. In *ACNS 2010*, volume 6487 of *Lecture Notes in Computer Science*, pages 123–138. Springer, 6 2010.

[4] Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improving the Time Complexity of Matsui's Linear Cryptanalysis. In K.-H. Nam and G. Rhee, editors, *The International Conference on Information Security and Cryptology - ICISC 2007*, volume 4817 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 11 2007.

[5] Jean-Jacques Quisquater, Baudoin Collard, and François-Xavier Standaert. Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent. In *Fast Software Encryption 2008*, volume 5086 of *Lecture Notes on Computer Science*, pages 382–397. Springer, 2 2008.

[6] François-Xavier Standaert, Jean-Jacques Quisquater, and Baudoin Collard. Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent. In *Inscrypt 2007*, volume 4990 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 9 2007.