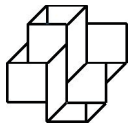

National Laboratory for Scientific Computing

Postgraduate Program in Computational Modeling

Post-Quantum Cryptography: An Efficient Differential Fault Analysis Attack and a New One-Time Signature Scheme

by

Juan del Carmen Grados Vásquez

PETRÓPOLIS, RJ - BRASIL

SEPTEMBER 2018

CRIPTOGRAFIA PÓS-QUÂNTICA: UMA ANÁLISE DIFERENCIAL EFICIENTE E UM NOVO ESQUEMA DE ASSINATURA DE USO ÚNICO

Juan del Carmen Grados Vásquez

TESE SUBMETIDA AO CORPO DOCENTE DO LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM MODELAGEM COMPUTACIONAL

Aprovada por:

Prof. Renato Portugal, D.Sc.

(Presidente)

Prof. Fábio Borges de Oliveira, Dr.-Ing.

Prof. Jauvane C. de Oliveira, Ph.D.

Prof. Lisandro Zambenedetti Granville, D.Sc.

Prof. Luis Menasché Schechter, D.Sc.

PETRÓPOLIS, RJ - BRASIL
OUTUBRO 2018

Grados Vásquez, Juan del Carmen

XXXX criptografia pós-quântica: uma análise diferencial eficiente e um novo esquema de assinatura de uso único / Juan del Carmen Grados Vásquez. Petropólis, RJ. : National Laboratory for Scientific Computing, 2018.

xx, yy p. : il.; 29 cm

Orientador: Renato Portugal

Tese (D.Sc.) – National Laboratory for Scientific Computing, 2018.

1. Hash-Based Cryptography. 2. Symmetric Cryptography. 3. Code-based Cryptography. 4. DFA. I. Portugal, Renato. II. LNCC/MCT. III. Título.

CDD XXX.XXX

I know that I know nothing. *(Socrates)*

To my wife, my daughter, my parents

and my brothers

for their patience.

Acknowledgements

I thank my wife, my daughter, my parents, my brothers and my sister, who support me to achieve my Ph.D. degree and many other goals in my life. To my wife and my daughter, for their love and their patience. To my parents, for all the commitment and dedication to the education of my brothers, my sister and me, always serving me as an example of overcoming and success. To my brothers and my sister, because they always gave me the strength to finish this work.

I thank my advisor, Prof. Renato Portugal, for all the knowledge, encouragement, and especially for always teaching me to seek the excellence. I also thank him for the patience and countless times that he has supported me within the path of the doctorate.

I would also like to thank Prof. Fabio Borges for his academic help and the strength he has always given me to complete my doctoral work. I also thank Prof. Pedro Lara for attending and participating in discussions with my advisor.

To friends, thank you for your patience and motivation during difficult times. I also thank you for all the celebrations, in moments not so difficult. To the other professors and Ph.D. fellows, thank you for helping me in this journey.

I acknowledge CAPES for the Ph.D. scholarship and the Clavis company for always supporting me to finish my doctorate.

Abstract of Thesis presented to LNCC/MCT as a partial fulfillment of the requirements for the degree of Doctor of Sciences (D.Sc.)

POST-QUANTUM CRYPTOGRAPHY: AN EFFICIENT DIFFERENTIAL FAULT ANALYSIS ATTACK AND A NEW ONE-TIME SIGNATURE SCHEME

Juan del Carmen Grados Vásquez

Outubro 2018

Advisor: Renato Portugal, D.Sc.

Cryptography is present in many fields in our daily life such as bank transactions, e-commerce, military communications, among others. Cryptography scientists have dedicated enormous effort to develop efficient and secure cryptographic schemes. The most common and successful ones are based on problems of number theory, for example, RSA and elliptic-curve cryptography. However, in 1994 Peter Shor of Bell laboratories managed to develop a quantum algorithm that can break RSA and other public-key cryptosystems based on number theory by using quantum computers. Symmetric primitives are also threatened by the arrival of quantum computers. Nonetheless, they are more resistant than the asymmetric primitives. In fact, the best-known quantum algorithm for attacking symmetric primitives is the Grover algorithm, which has a quadratic improvement over the best classical algorithm. So, according to the literature, it is enough to double the size of the keys to resist attacks of this algorithm. Efforts are concentrated in developing public and private key cryptographic schemes that resist the quantum attacks. These schemes are usually classified as 1) hash-based schemes, 2) code-based schemes, 3) lattice-based schemes, 4) multivariate-quadratic-equation schemes, and 5) secret-key schemes.

We study schemes of the classes 1), 2), and 5); and we divide the thesis into

three parts. In the first part, we introduce coding theory and provide an overview of code-based cryptography focusing mainly on the digital signature of Courtois, Finiasz, and Sendrier. In the second part, we study one-time signature schemes that resist quantum attacks. These schemes belong to the hash-based and code-based classes. Our contribution in this part is a new code-based one-time signature scheme. In the third part, we give an overview of differential fault analysis, and we study one scheme proposed by NSA in 2013 — Simon. Our contribution in this part is an efficient differential fault analysis on Simon.

Resumo da Tese apresentada ao LNCC/MCT como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

CRIPTOGRAFIA PÓS-QUÂNTICA: UMA ANÁLISE DIFERENCIAL EFICIENTE E UM NOVO ESQUEMA DE ASSINATURA DE USO ÚNICO

Juan del Carmen Grados Vásquez

Outubro 2018

Orientador: Renato Portugal, D.Sc.

A criptografia está presente em muitos campos da nossa vida cotidiana, como transações bancárias, comércio eletrônico, comunicações militares, entre outros. Pesquisadores em criptografia dedicaram um enorme esforço para desenvolver esquemas criptográficos eficientes e seguros. Os mais comuns e bem sucedidos são baseados em problemas da teoria dos números, por exemplo, RSA e criptografia de curvas elípticas. No entanto, em 1994, Peter Shor, dos laboratórios Bell, conseguiu desenvolver um algoritmo quântico que pode quebrar o RSA e outros sistemas de chave pública baseados na teoria dos números, usando computadores quânticos. Primitivas simétricas também estão ameaçadas pela chegada de computadores quânticos. No entanto, elas são mais resistentes do que as primitivas assimétricas. De fato, o algoritmo quântico mais conhecido para atacar primitivas simétricas é o algoritmo de Grover, que possui um ganho quadrático. Então, segundo a literatura, basta dobrar o tamanho das chaves para elas ficarem resistentes aos ataques deste algoritmo. Esforços estão concentrados no desenvolvimento de esquemas criptográficos de chave pública e privada que resistem aos ataques quânticos. Esses esquemas são geralmente classificados como: 1) esquemas baseados em funções hash, 2) esquemas baseados em códigos, 3) esquemas baseados em reticulados, 4) esquemas baseados em equações quadráticas multivariadas e 5) esquemas de chave

privada.

Estudamos esquemas das classes 1), 2) e 5); e nós dividimos a tese em três partes. Na primeira parte, introduzimos a teoria de codificação e fornecemos uma visão geral da criptografia baseada em códigos, focando principalmente na assinatura digital de Courtois, Finiasz e Sendrier. Na segunda parte, estudamos esquemas de assinatura de uso único que resistem aos ataques quânticos. Esses esquemas pertencem às classes baseadas em funções hash e baseadas em códigos. Nossa contribuição nesta parte é um novo esquema de assinatura única baseada em códigos. Na terceira parte, damos uma visão geral da análise diferencial de falhas, e estudamos um esquema proposto pela NSA em 2013 — Simon. Nossa contribuição nesta parte é uma análise diferencial eficiente de falhas no Simon.

Contents

List of Figures

Figure

List of Tables

Table

List of Symbols

- T: total number of rounds in the cipher. For instance, $T = 32$ for Simon 32/64.
- $\left(L^{i-1}, R^{i-1}\right)$: $2n$-bit input of the $i-th$ round of the cipher, $i \in \{0, \cdots, T-1\}$.
- $\left(L^{i+1}, R^{i+1}\right)$: $2n$-bit output of the $i-th$ round of the cipher, $i \in \{0, \cdots, T-1\}$.
- L^{i^*},R^{i^*} wrong left half input and wrong right half input respectively at position i.
- P: plaintext.
- C: ciphertext.
- C^*: faulty ciphertext.
- K^i: n-bit round key used in the i^{th} round of the cipher, $i \in \{0 \cdots T-1\}$.
- $x <<< y$: circular left rotation of x by y bits.
- x_l: l^{th} bit of the bit string x.
- $\oplus$: logical operator xor.
- $\odot$: logical operator and.
- $a\%b$: $a \mod b$.
- M^T: Transposition of a matrix M.
- v^T: Transposition of a vector v.
- $(x||y)$: Concatenation of vector x followed by vector y.
- $x \xleftarrow{\$} X$: Variable x is uniformly sampled at random from set X.

Chapter 1

Introduction

Since very remote times, humans needed to hide information. One of the first examples is found in ancient Greece. The Spartan ephors used a system called Scythian (Holden, 2017), which consisted of two wooden poles of the same thickness, and a parchment. One rod was for the sender and the other for the receiver. To send some secret information, the sender had to wrap a parchment in the wooden stick and to write a secret message on top of it. In this way, when the parchment was released, the letters composing the message were transposed. To decrypt the message, the receiver had to wrap the parchment in a stick of the same thickness. This technique, that alters the order of the letters is known as transposition. Another technique to hide information is substitution. In this technique, the letters that make up the message are replaced by other letters of the alphabet. For example, the Hebrews swapped the first letter of its alphabet with the last; the second with the penultimate, and so on. Sophisticated variations of the substitution technique were also used by Emperor Cesar, Vernam, among others.

For many years, the mentioned techniques were used mainly in the military area. For example, during World War I, Germany used encrypted telegrams to urge Mexico to invade the United States. Nevertheless, Great Britain decrypted the content of these telegrams and advised the United States of German intentions (Kahn, 1996). According to history, this was one of the reasons why the

United States decided to enter this World War I. Another example of the use of substitution and transposition techniques is also found in World War II. Germany used a machine, called Enigma, to encrypt its messages. Again, Great Britain managed to decrypt the messages cryptographed by this machine contributing to the end of the Second World War.

The techniques used to build encryption methods gave rise to a part we now know as cryptography, and the techniques used to break the security of such methods gave rise to what we now know as cryptanalysis. Until 1976, cryptography methods used only one key to encrypt as much as to decrypt. For example, in the system used by the Hebrews, the key was to know that the first letter of the alphabet should be replaced by the last letter, the second by the penultimate, and so on. Cryptography that uses the same key to encrypt and decrypt is known as symmetric cryptography or private-key cryptography (depicted in Figure 1.2). One problem with this type of cryptography is the exchange of keys. In fact, suppose we want to send a secret message from Brazil to China. How do we send the key to the receiver? With symmetric cryptography, we have no choice but to send it through an insecure channel or meet personally to share our key. Thus, we pay a high price for establishing secure communication with this type of cryptography.

In 1976, a theoretical work (Diffie and Hellman, 1976) managed to solve the problem of key exchange, giving rise to asymmetric or public-key cryptography (depicted in Figure 1.1). The first public-key cryptography system, which is used to date, was proposed by Rivest, Shamir, and Adleman (Rivest, Shamir, and Adleman, 1978). This system is call RSA. After RSA, other works proposed new public-key or asymmetric cryptography schemes, for example, ElGamal (Elgamal, 1985). All these works are based on difficult problems of the number theory. For instance, RSA is based on the practical difficulty of factoring a number that is the product of two large prime numbers. In the work of Rivest et al., a system to sign and verify digital documents was also presented (depicted in Figure 1.3). This system is also based on the practical difficulty of the product factorization of

two large prime numbers.

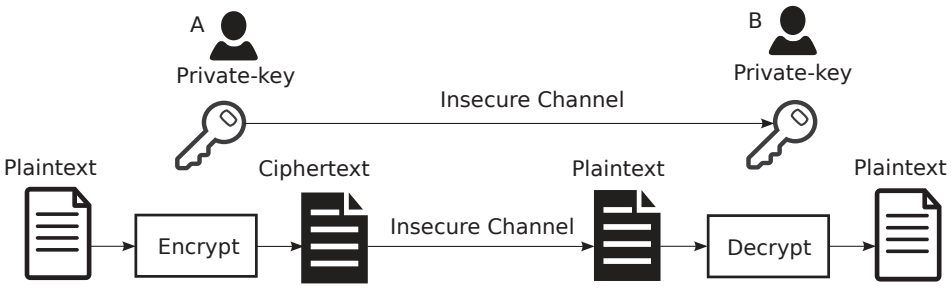

Figure 1.1: Private-key cryptography.

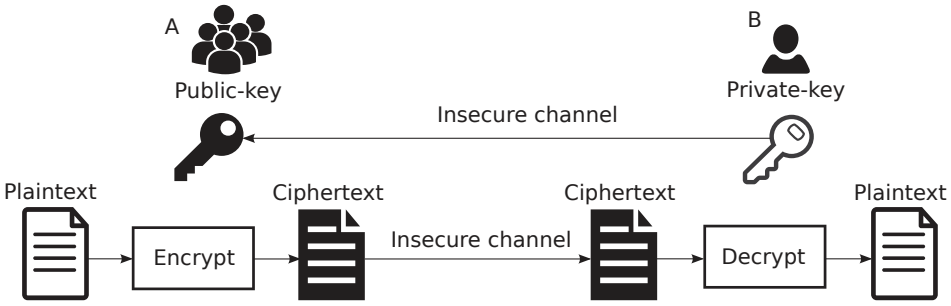

Figure 1.2: Public-key cryptography.

Actually, RSA is one of the most deployed cryptosystems, and as other cryptosystems, it is used to secure bank transactions, e-commerce, military communication among others. However, in 1997, Peter Shor, from Bell laboratories, developed a quantum algorithm, which bears his name, that can be used to break RSA and other public-key systems based on number theory (Shor, 1997). Shor's algorithm can factorize a number N in time $O((\log N)^3)$ and space $O(\log N)$. Due to this result, scientists in the area searched for cryptographic systems that resist these types of attacks. Some forgotten cryptographic systems were considered again due to their resistance against the quantum attacks, for instance, the McEliece cryptographic scheme. Besides, new schemes that resist these attacks have appeared.

Symmetric primitives also suffer from a reduced security in the quantum world, but this security reduction is much less drastic than for many asymmetric primitives. So far, the main quantum attack on symmetric algorithms follows from Grover's algorithm (Grover, 1996) for searching an unsorted database of size N in $O(N^{1/2})$ time. It can be applied to any generic exhaustive key search, but

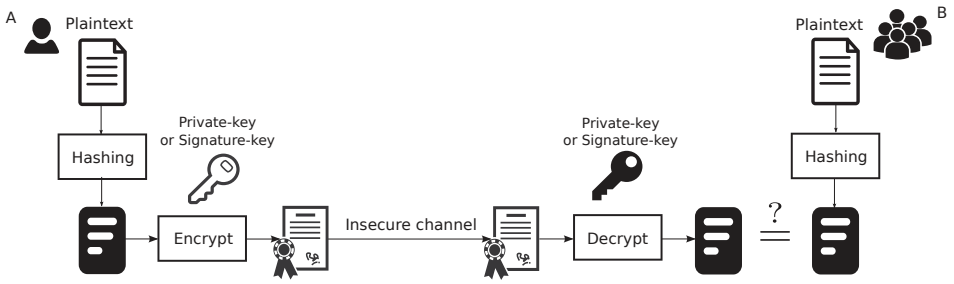

Figure 1.3: Signature scheme.

merely offers a quadratic speed-up compared to a classical attack. Therefore, the consensus is that key lengths should be doubled to restore the security.

The systems that are believed to be resistant against attacks from quantum computers are known as post-quantum systems, and Bernstein, Buchmann, and Dahmen (2008) classified them as follows:

- Hash-based cryptography;
- Code-based cryptography;
- Lattice-based cryptography;
- Multivariate-quadratic-equations cryptography;
- Secret-key cryptography.

Motivated in both the search for new cryptographic schemes that deal with post-quantum attacks and in the search for cryptographic attacks against post-quantum systems, we will focus on signatures schemes of the first two classes, and on a DFA (Biham and Shamir, 1997) attack against the last class. Thus, we divide our thesis into three parts.

In the first part, we introduce coding theory in Chapter 2, and give an overview of code-based cryptography in Chapter 3, where we introduce the signature of Courtois, Finiasz, and Sendrier (CFS) (Courtois, Finiasz, and Sendrier, 2001). This scheme is a well-known alternative signature scheme based on the hardness of decoding random linear codes. We did a review of that scheme. The

the best of our knowledge, it was the first practical digital signature scheme based on codes and thus one of the most studied post-quantum schemes, and not only that, this system helped us to understand how to build a digital signature system based on codes.

In the second part, we give an overview of hash-based cryptography, in Chapter 4, and we present the main schemes in this area such as the Merkle scheme (Merkle, 1990) and W-OTS$^+$ (Hülsing, 2013b). In Chapter 5, we present two code-based signature schemes — KKS (Kabatianskii, Krouk, and Smeets, 1997) and BMS-OTS (Barreto, Misoczki, and Simplicio Jr., 2011). These are one-time schemes, resistant to quantum attacks, and based on the hardness of decoding random linear codes. An advantage of these proposals is that they use generic codes for which no efficient decoder is known, rather the ones that have trapdoors. In addition, BMS-OTS has a security proof in the random oracle model. Thus, a drawback of one-time signatures schemes based on codes is the lack of a security proof in the standard model. Another drawback of KKS and BMS-OTS is their large verification-key sizes. In fact, KKS and BMS-OTS use two matrices as verification key, these two matrices are large when instantiated with random matrices. Thus, we introduce a new one-time signature scheme based on codes. This proposal is hash-free and, when instantiated with random matrices, it has smallest verification key. Also, since it is hash-free, a proof in the standard model could be done.

In the third part, we give an overview of the differential fault analysis (DFA). This part is divided into two chapters. In Chapter 6, we present fundamentals of fault attacks and the fault attack procedure. In Chapter 7, we present the Simon family cipher (Beaulieu, Shors, Smith, Treatman-Clark, Weeks, and Wingers, 2013). This family of lightweight block ciphers was proposed by NSA in 2013 and has been optimized for performance in hardware implementations. The Simon family uses the Feistel structure, which consists of various identical rounds. There are several DFA attacks against the Simon family, which use a bit, a byte, or a

random model. All these models, that appeared until before the publication of our attack, use several rounds to perform their attacks. Due that, we introduce an efficient one-bit model for differential fault analysis against Simon (Grados, Borges, Portugal, and Lara, 2015), which use half of rounds in comparison with those attacks that appeared until before the publication of our attack. In particular, with a single round, we get the entire key of two members of the Simon family. This work was presented in FDTC, Fault Diagnosis and Tolerance in Cryptography, which is one of the most recognized workshops in the area.

Part I

Code-based Schemes

Chapter 2

Preliminaries on Coding Theory

In this chapter, we will give an overview of the principal definitions and theorems that we will need in the thesis, for more information please refer to Lint (1998) and MacWilliams and Sloane (1977).

Shannon (1948) established the basis of the coding and information theory in his work "A Mathematical Theory of Communication". Initially, Shannon studied how to encode messages before transmitting them on a noisy communication channel. Coding theory has been developed throughout the years, and now it incorporates the study of codes, including error-detecting and error-correcting codes. Several areas use coding theory, for example, data compression, cryptography, error correction, and networking. The following example is classic in the error-correcting area, and it will help to understand concepts about codes that will appear later.

In 1971, the spacecraft Mariner 71 traveled to Mars to send data and pictures to Earth. The pictures were transmitted using a fine grid. In each square was quantified the blackness degree of the image with a number between 0 and 63. A binary string of length 6 represented these numbers. The bits 0 and 1 were represented and transmitted as two different signals to a receiver station on Earth. However, sometimes, the signals arrived weak due to big distances causing bit-flip errors.

The spacecraft Mariner 71 used the Reed Solomon code. This code, as others error-correcting codes, incorporates some redundancy, so that, if some of the

symbols in a transmitted message were changed, we could still figure out the sent message. The Reed Solomon code encodes its messages in bit strings consisted of k bits of the original data and t bits with redundancy data. The Reed Solomon code used in Mariner 71 was the $(6, 4)$ code. In that code, length of the codeword is $n = 6$ bits where $k = 4$ bits are original data and $t = 2$ bits are redundancy data. This code can detect up to 2 errors.

2.1 Linear Codes

Linear codes are the cornerstone for almost all code-based cryptosystems. Traditionally linear codes are partitioned into two cases: block codes and convolutional codes. All cryptosystems described in this thesis use block codes. This kind of codes encodes the data to be transmitted into blocks of equal size. Also, each one of such blocks can be coded and decoded independently. The blocks resulted of the codification are called *codewords*, and we call *error-correcting code* the set of all these codewords. The messages are coded using an alphabet, and generally, is assumed to be a finite field. In this chapter, we use the Galois Field $\mathbb{F}_2$ as alphabet and the vectors would always be represented as line vectors and that the matrices would always be multiplied to the right. The following paragraphs describe a formal definition of linear codes.

Definition 2.1 (38, Misoczki (2010))**.** *Let $\mathbb{F}_2$ a finite field with two elements. A linear subspace with dimension k, of the vector space $\mathbb{F}_2^n$, is a binary linear code $[n, k] - \mathcal{C}$ with length n and dimension k.*

The code used in the spacecraft Mariner 71 mentioned earlier can correct up to 2 errors because it recognizes whether at most 2 bits were changed from the right codeword. The number of bits that were changed leads to the following definition.

Definition 2.2 (33, Lint (1998))**.** *If $x \in \mathbb{F}_2^n$ and $y \in \mathbb{F}_2^n$, then the distance $d(x, y)$*

of x and y is defined by

$$d(x, y) \coloneqq |\{i \mid 1 \leq i \leq n, x_i \neq y_i\}|,$$

and the weight of x $w(x)$ is defined by

$$w(x) \coloneqq d(x, 0).$$

The distance defined above is also called Hamming distance. Notice that the Hamming distance measures the required number of substitutions to change a codeword into another. Also, $\mathbb{F}_2^n$ is a metric space because d satisfies the properties of a typical distance.

Definition 2.3 (1, Sudan (2001))**.** *For any vector $x \in \mathbb{F}_2^n$,*

$$B(x, e) = (y \in \mathbb{F}_2^n | d(x, y) \leq e).$$

Definition 2.4 (34, Lint (1998))**.** *The minimum distance of a code $\mathcal{C}$ is defined by*

$$\min \{d(x, y) | x \in \mathcal{C}, y \in \mathcal{C}, x \neq y\}.$$

For a linear code the minimum distance is the minimal weight of a non-zero codeword. In other words, the minimum distance of a linear code $\mathcal{C}$ is

$$d = \min \{w(x) | x \in \mathcal{C}, x \neq 0\}$$

Later, we will see that the minimum distance of a code is important because it determines the ability to detect, or correct, wrong codewords.

Hereafter, we shall use $[n, k, d] - \mathcal{C}$ as notation for a k-dimensional linear code $\mathcal{C}$ of length n and minimum distance d.

In linear codes, the process of encoding and decoding is made using linear maps, which can be represented using matrices. The linear map for the encoding

process is known as generator matrix. The linear map for the decoded process is known as parity check matrix.

Definition 2.5 (35, Lint (1998))**.** *A generator matrix* G *for a linear code* $\mathcal{C}$ *is a matrix with dimensions* $k \times n$ *where its rows are a basis of* $\mathcal{C}$*.*

A generator matrix G is in standard form (or reduced echelon form) if $\mathsf{G} = (I_k | P)$, where I_k is the identity matrix with dimensions $k \times k$. If G is in standard form, then the first k symbols are called information symbols and the remaining parity-check symbols.

Now, we are going to see how to obtain a parity check matrix from a generator matrix, but before, we need the following definitions.

Definition 2.6 (4, Lint (1998))**.** *Given* $a \coloneqq (a_1, a_2, \cdots, a_n)$ *and* $b \coloneqq (b_1, b_2, \cdots, b_n)$*. The inner product* $\langle x, y \rangle$ *is defined by*

$$\langle x, y \rangle \coloneqq a_1 b_1 + a_2 b_2 + \cdots + a_n b_n.$$

Definition 2.7 (36, Lint (1998))**.** *If* $\mathcal{C}$ *is a* $[n, k]$ *code, we define the dual code* $\mathcal{C}^{\perp}$ *by*

$$\mathcal{C}^{\perp} \coloneqq \{ y \in \mathbb{F}_2^n | \forall_{x \in \mathcal{C}} [\langle x, y \rangle = 0] \}.$$

The dual code $\mathcal{C}^{\perp}$ is clearly a linear code, namely $[n, n-k] - \mathcal{C}^{\perp}$. If $\mathsf{G} = (I_k | P)$ is a generator matrix in standard form of the code $\mathcal{C}$, then $\mathsf{H} = (-P^T | I_{n-k})$ is a generator matrix for $\mathcal{C}^{\perp}$. This follows from the fact that $\mathsf{G}\mathsf{H}^T = 0$ implies that every codeword $a\mathsf{G}$ has inner product 0 with every row of H, i.e.,

$$x \in \mathcal{C} \iff x\mathsf{H}^T = 0.$$

The matrix H is called the parity check matrix of $\mathcal{C}$.

Definition 2.8 (36, Lint (1998))**.** *If* $\mathcal{C}$ *is a linear code with parity check matrix* H*, then for every* $x \in \mathbb{F}_2^n$ *we call* $x\mathsf{H}^T$ *the syndrome of* x*.*

The set of all codewords with the same syndrome is called coset. More formally

Definition 2.9. *Let $\boldsymbol{s} \in \mathbb{F}_2^R$. We denote the set of words in $\mathbb{F}_2^n$ with syndrome $\boldsymbol{s}$ by*

$$S_{\mathsf{H}}^{-1}(\boldsymbol{s}) = \{\boldsymbol{y} \in \mathbb{F}_2^N | \boldsymbol{y}\mathsf{H}^T = \mathbf{s}\}.$$

By definition, we have $S_{\mathsf{H}}^{-1}(0) = \mathcal{C}$ for any parity check matrix H of $\mathcal{C}$. The sets $y + \mathcal{C}$, for all y in $\{0,1\}^n$, are called the cosets of $\mathcal{C}$. There are exactly 2^{n-k} different cosets which form a partition of $\{0,1\}^n$.

If two words x and y are in the same coset, there exists two codewords c_1 and c_2 such that $x = a + c_1$ and $y = a + c_2$. We have then that $\mathsf{H}x^T = \mathsf{H}(a+c_1)^T = \mathsf{H}a^T = \mathsf{H}(a + c_2)^T = \mathsf{H}y^T$ so the two words have the same syndrome. On the other hand if two words x and y have the same syndrome, then $\mathsf{H}x^T = \mathsf{H}y^T$ and $\mathsf{H}(x - y)^T = 0$. This is the case only if $x - y$ is a codeword and therefore x and y are in the same coset. Therefore, we have the following lemma.

Lemma 2.1 (17, MacWilliams and Sloane (1977))**.** *Two words are in the same coset if and only if they have the same syndrome.*

Theorem 2.1 (67, Klima, Sigmon, and Stitzinger (2000))**.** *Let $\mathcal{C}$ be a linear code with parity matrix H and s the minimum number of linearly dependent columns in H, then s is the minimum distance of the code.*

Theorem 2.2 (11, MacWilliams and Sloane (1977))**.** *A binary linear code with minimum distance d can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.*

In linear codes, the proportion of useful data (non-redundant data) R is given by the following definition

Definition 2.10 (34, Lint (1998))**.** *The rate of $\mathcal{C}$ is defined by*

$$R \coloneqq k/n$$

For example, in the (6,4) Reed-Solomon code used by Mariner 71 quoted at the beginning of this chapter the rate is 4/6.

In code-based cryptosystems, the next theorem is important because helps to select codes that guarantee the hardness of these cryptosystems against certain types of attacks.

Theorem 2.3 (43, MacWilliams and Sloane (1977)). *A binary linear code* $[n, k, d]-\mathcal{C}$ *is ensured to exist if*

$$\sum_{j=0}^{d-2} \binom{n-1}{j} < 2^{n-k}.$$

The last expression is called the Gilbert-Varshamov (GV) bound. Random binary codes are known to meet the GV bound in the sense that the above inequality comes very close to be equality (MacWilliams and Sloane (1977)). No family of binary codes is known that can be decoded in subexponential time up to the GV bound, nor is any subexponential algorithm known that can decode general codes up to the GV bound.

Another significant bound in coding theory is the Singleton bound.

Definition 2.11 (33, MacWilliams and Sloane (1977)). *If* $\mathcal{C}$ *is* $[n, k, d]$ *code, then* $d \leq n - k + 1$.

2.2 Compact Representation Theory

To reduce both key and signature sizes several algorithms of code-based schemes use ranking functions, which have codewords as domain and integers as image. These functions are also called ranks. The rank of a particular codeword of length n and Hamming weight k is the number of codewords that precede it in the sort listing of all possible codewords of length n and Hamming weight k.

Dennis Stanton (1986) propose a ranking function widely used in code-based cryptography. The domain of this function is a set of bit-strings representing codewords in colex order. Recall that two k-subsets A and B of a set with strict linear ordering $<$ are said to be in co-lexicographical order (colex order) if, for the

sequences a and b of their elements sorted in ascending order, there exists index i such that $a_i < b_i$ and $(\forall_j)_j > i \Rightarrow a_j = b_j$.

The following definition formalizes the map presented in (Dennis Stanton, 1986).

Definition 2.12 (10, Dennis Stanton (1986))**.** *Let B be a set of bit-strings of length n. Let σ an arbitrary element of a subset C of B such that the length of the bit-strings of C is k. Let σ_i, $0 \leq i \leq k-1$, be the position where σ has an element different of zero. The ranking function*

$$r(\sigma_1\sigma_2\cdots\sigma_k) = \sum_{i=1}^{k} \binom{\sigma_i - 1}{i}$$

assigns an unique number $0 \leq a \leq \binom{n}{k}$ to any k-element of the subset C.

2.3 Decoding

Let $c = y + e \in \mathbb{F}_2^n$ be the codeword that arrives to a receiver, where y was coded using a code $[n, k, d] - \mathcal{C}$ and e is an error vector with number of ones less or equal than $\left\lfloor \frac{d-1}{2} \right\rfloor$. If we want to decode c one could use the follow *Syndrome Decoding method*.

1. Calculate and save in a table the syndrome of each possible error vector with Hamming weight less or equal than $\left\lfloor \frac{d-1}{2} \right\rfloor$,
2. Calculate the syndrome of the received message. Notice, that this message has the same syndrome as the error vector added into codeword,
3. Identify the error vector using the table calculated in step 1,
4. Sum the identified error vector in the last step to the received codeword.

As we shall see later, we need to calculate 2^{n-k} syndromes in step 1 to construct the cited table, that is, the complexity of the above method is exponential. There are other types of codes, which have trapdoors that allow "efficient decodification", for instance the Goppa code used in the McEliece cryptosystem explained

in Section 2.4.5. Those that do not have an algebraic structure to allow an "efficient decodification" are known as *General Linear Codes*.

Decode *General Linear Codes* can be stated in the following problem.

Problem 2.1 (Syndrome Decoding – SDP(N, R, n, s))**.** *Given a binary matrix* $\mathsf{H}^{R\times N}$*, a word* s *in* $\mathbb{F}_2^R$*, and a positive integer* n*, find the word* e *in* $S_{\mathsf{H}}^{-1}(s)$ *with Hamming weight* $wt(e) \leq n$*.*

Berlekamp, McEliece, and van Tilborg (1978) showed that the associated decision problem is NP-complete.

Decode *General Linear Codes* is one of the most active areas of postquantum research. Most algorithms with best performance to solve SDP are variations of the *Information Set Decoding* (ISD) algorithm (Prange, 1962). Let $\mathcal{C}$ a $[N, K]$-code. An information set is a set of K positions among the N positions of the support. Also, let SDP(N, R, n, H, s) an instance of the SD problem. The ISD algorithm consists in reducing the search space of solutions. The first step consists in transform the original problem in another equivalent by randomly permuting the columns of H using a permutation matrix P. Let $\mathsf{H}' = \mathsf{H}P$ the result of this permutation. The next step is to transform H' into a systematic form matrix $(\mathsf{I}_{N-K}|\mathsf{Q})$ where $\mathsf{Q} \in \mathbb{F}_2^{(N-K)\times K}$ and I_{N-K} is the $(N-K)$-dimensional identity matrix. Next, one calculates all linear combinations with n vectors. If the sum of at least one of these linear combinations with s is 0, then the last K columns of the matrix forms an information set and then we found a solution for SDP(N, R, n, H, s).

2.4 Special Codes

In this section, we define some of the codes that we will use in the following chapter.

2.4.1 Equidistant Codes

Definition 2.13 (27,Kaski and Östergård (2005))**.** *A code is called constant weight if all codewords have the same weight.*

Definition 2.14 (29,Kaski and Östergård (2005)). *A code is called equidistant if the distance between any two distinct codewords is equal to a given parameter d.*

Notice, that in virtue of Definition 2.4 equidistant codes are equivalent to constant weight codes.

2.4.2 BCH Codes

The Bose–Chaudhuri–Hocquenghem codes (BCH) form a class of cyclic error-correcting codes that are constructed using polynomials over a finite field. BCH codes were invented in 1959 by French mathematician Alexis Hocquenghem, and independently in 1960 by Raj Bose and D. K. Ray-Chaudhuri.

Definition 2.15 (MacWilliams and Sloane (1997)). *A cyclic code of length n over $\mathbb{F}_q$ with generator polynomial $g(x)$ is a BCH code of designed distance δ if, for some integer $b \geq 0$, $g(x)$ is the monic polynomial of lowest degree over $\mathbb{F}_q$ having $\alpha^b, \alpha^{b+1}, \cdots, \alpha^{b+\delta-2}$ as zeros.*

Its parity check matrix is

$$\mathsf{H} = \begin{pmatrix} 1 & \alpha^{b} & \alpha^{2b} & \vdots & \alpha^{b(n-1)} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \vdots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{b+\delta-2} & \alpha^{2(b+\delta-2)} & \dots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix}.$$

The next lemma is a known bound on the weight of the codewords that belong to binary BCH codes. As we shall see in Section 5.1, this bound was used by a signature scheme to implement signatures with practicable sizes.

Lemma 2.2 (Carlitz-Uchiyama Bound). *Let $\mathcal{C}$ be the dual of a binary BCH code of length $n = 2m - 1$ and designated distance $\delta = 2s + 1$. Then for any $x \in \mathcal{C}$:*

$$\left|\mathrm{wt}(x) - \frac{n+1}{2}\right| \leq (s-1)\sqrt{n+1}. \tag{2.1}$$

2.4.3 GRS Codes

Reed-Solomon codes were introduced in 1960 by I. Reed and G. Solomon. In this section, we will define the generalization of these.

Definition 2.16. *Let $F = \mathbb{F}_q$ be a finite field, where $q \in \mathbb{N}$ is a prime number. Given $v_1, ..., v_n \in F$ non-zero elements and $\alpha_1, ...\alpha_n \in F$. Let $\boldsymbol{v} = (v_1, ..., v_n)$ and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$. For $0 \leq k \leq n$, Generalized Reed Solomon Codes are defined as:*

$$GRS_{n,k}(\boldsymbol{\alpha}, \boldsymbol{v}) = \{(v_1 f(\alpha_1), v_2 f(\alpha_2), ..., v_n f(\alpha_n)) | f(X) \in F[X]_k\}$$

with minimum distance $k + 1$.

In the previous definition, $F[X]_k$ represents the set of polynomials in the ring $F[X]$, of degree less than k. It is easy to see that this set is a vector space of dimension k over F

2.4.4 Alternant Codes

Alternant codes are defined by making a constraint in the GRS codes. The idea of this restriction is to select the portion of the GRS code that is also in a subfield of the original field. This restriction is known as subfield subcode.

Definition 2.17. *Let $\mathbb{F}_{q^m}$ be a finite field extension of $\mathbb{F}_q$. Given $\mathcal{C} \subseteq (\mathbb{F}_{q^m})^n$ a code over $\mathbb{F}_{q^m}$.*

$$\mathcal{C}|_{\mathbb{F}_q} := \mathcal{C} \cap \mathbb{F}_q^n$$

is called of subcode subfield of $\mathcal{C}$.

Definition 2.18. *Let $\mathbb{F}_q$ be a subfield of $\mathbb{F}_{q^m}$, then the alternant code of $GRS_{n,k}(\alpha, v)$ in the subfield $\mathbb{F}_q$ is $GRS_{n,k}(\boldsymbol{\alpha}, \boldsymbol{v}) \cap \mathbb{F}_q$.*

2.4.5 Goppa Codes

Among the alternant codes, as presented in MacWilliams and Sloane (1997), there is a subclass of asymptotically good codes, in the sense that they reach the

GV bound. These codes are interesting too because it is easy to calculate their minimum distance. The creation of these codes is motivated by the BCH codes and can be described as a function of a generator polynomial $g(X)$, called the Goppa polynomial, and a set of elements, L, which is contained in a finite field where the Goppa code has its symbols (see section 10.5 Hefez and Villela, 2008).

Definition 2.19. *Let F be a finite field, extension of a finite field K. Given $g(X) \in F[X]$ and $L = \{\alpha_0, \alpha_1, ..., \alpha_{n-1}\} \subset F$, where α_i are two by two distinct and such that $g(\alpha_i) \neq 0$ for $i = 0, ..., n-1$. It is defined as Goppa code the set*

$$\Gamma_K(L, g) = \left\{ (c_0, ..., c_{n-1}) \in K^n; \sum_{i=0}^{n-1} c_i g(\alpha_i)^{-1} = 0 \right\}.$$

It can be easily shown that the above set is a linear code, and it is said: classic Goppa code over K associated to the set L and the polynomial $g(X)$.

2.4.5.1 Parity Check Matrix

Below, we will present one of the ways for constructing of the parity check matrix for the Goppa codes.

Let

$$g(X) = \sum_{j=0}^{\delta} g_j X^j, \text{ with } g_\delta \neq 0 \in F,$$

then

$$\frac{g(X) - g(\alpha)}{X - \alpha} = \sum_{j=0}^{\delta} g_j \cdot \frac{X^j - \alpha^j}{X - \alpha}.$$

Therefore, $(c_0, c_1, ...c_{n-1}) \in \Gamma_K(L, g)$ if and only if

$$\sum_{i=0}^{n-1} (g(\alpha_i)^{-1} \sum_{j=t+1}^{\delta} g_j \alpha_i^{j-1-t}) c_i = 0, \ 0 \leq t \leq \delta - 1.$$

From the above expression we have $c \in \Gamma_K(L,g) \iff \mathsf{B}c^t = 0$, where:

$$\mathsf{B} = \begin{pmatrix} g(\alpha_0)^{-1}g_\delta & \dots & g(\alpha_{n-1})^{-1}g_\delta \\ g(\alpha_0)^{-1}(g_{\delta-1} + g_\delta \cdot \alpha_0) & \dots & g(\alpha_{n-1})^{-1}(g_{\delta-1} + g_\delta \cdot \alpha_{n-1}) \\ \vdots & \dots & \vdots \\ g(\alpha_0)^{-1}\sum_{j=1}^{\delta} g_j \alpha_0^{j-1} & \dots & g(\alpha_{n-1})^{-1}\sum_{j=1}^{\delta} g_j \alpha_{n-1}^{j-1} \end{pmatrix}$$

and

$$c^T = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{\delta-1} \end{pmatrix}.$$

Doing $l_j \longleftarrow l_j - (g_\delta \sum_{i=1}^{j-1} g_{\delta-i}\alpha_0^{j-i-1})l_1$ for $1 \leq j \leq \delta - 1$, where l_j are rows of B, we obtain the following matrix

$$\hat{\mathsf{H}} = \begin{pmatrix} g(\alpha_0)^{-1} & \dots & g(\alpha_{n-1})^{-1} \\ g(\alpha_0)^{-1}\alpha_0 & \dots & g(\alpha_{n-1})^{-1}\alpha_{n-1} \\ \vdots & \dots & \vdots \\ g(\alpha_0)^{-1}\alpha_0^{\delta-1} & \dots & g(\alpha_{n-1})^{-1}\alpha_{n-1}^{\delta-1} \end{pmatrix}.$$

Therefore $c \in \Gamma_K(L,g) \iff \hat{\mathsf{H}}c^t = 0$. Notice that the entries of this parity check matrix $\hat{\mathsf{H}}$ are in the field F. One can consider F as a K - vector space of dimension m. Thus we can write each input of the matrix $\hat{\mathsf{H}}$ as a vector of length m with coordinates in K, obtaining a matrix H' of dimensions $(m \times \delta) \times n$ with coefficients in K. Therefore, $c \in \Gamma_K(L,g) \iff \mathsf{H}'c^t = a$.

The minimum distance d of this code is at least δ. To demonstrate this, we can see that by creating the matrix $\hat{\mathsf{H}}'$ from any $\delta - 1$ columns of $\hat{\mathsf{H}}$, we have:

$$\det(\hat{\mathsf{H}}') = \prod_{i=0}^{\delta-1} g(\alpha_i)^{-1} \cdot \det\left(\begin{pmatrix} 1 & \dots & 1 \\ \alpha_0 & \dots & \alpha_{\delta-1} \\ \vdots & \dots & \vdots \\ \alpha_0^{\delta-1} & \dots & \alpha_{\delta-1}^{\delta-1} \end{pmatrix}\right).$$

The right-hand matrix of the above expression is the Vandermonde matrix, in which we have $\det(\hat{\mathsf{H}}) \neq 0$. Thus, any $\delta - 1$ columns of $\hat{\mathsf{H}}$ are linearly independent. This result and the Theorem 2.1 demonstrate that the minimum distance of this Goppa code is at least δ.

The matrix B can also be described as the product of three matrices: Toepliz, T, Vandermonde, V and a diagonal matrix D, (Section 2.4.2.1 Hoffmann, 2011). Like this,

$$\mathsf{B} = \mathsf{T} \cdot \mathsf{V} \cdot \mathsf{D},$$

where

$$\mathsf{T} = \begin{pmatrix} g_\delta & 0 & 0 & \dots & 0 \\ g_{\delta-1} & g_\delta & 0 & \dots & 0 \\ g_{\delta-2} & g_{\delta-1} & g_\delta & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \dots & g_\delta \end{pmatrix}, \qquad \mathsf{V} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{\delta-1} & \alpha_2^{\delta-1} & \dots & \alpha_{n-1}^{\delta-1} \end{pmatrix}$$

$$\mathsf{D} = \begin{pmatrix} 1/g(\alpha_0) & 0 & \dots & 0 \\ 0 & 1/g(\alpha_1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1/g(\alpha_{n-1}) \end{pmatrix}.$$

2.4.5.2 Generator Matrix

As explained in Section 2.1, the generator matrix G of the code, can be calculated from H, computing a basis for the null space of the linear transformation induced by the matrix H.

For a better understanding of the construction of the G and H matrices, let's see an example of its construction.

Example. We begin by constructing an extension $F = \mathbb{F}_{2^3}$ of a finite field $K = \mathbb{F}_2$, with the residual class ring $K[X]_{P(X)}$, where $P(X) = X^3 + X + 1$ is an irreducible polynomial. Let $g(X) = X^2 + X + 1$ Goppa's polynomial, so the minimum distance d is $\delta = \deg(g(X)) = 2$. Let a be a primitive element of F and $L = F \setminus \{0\}$ be the support of code. We obtain:

$$\mathsf{T} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},$$

$$\mathsf{V} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & a & a^2 & a+1 & a^2+a & a^2+a+1 & a^2+1 \end{pmatrix},$$

$$\mathsf{D} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a^2+a & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a^2+a \end{pmatrix}$$

and, thus,

$$\mathsf{B} = \begin{pmatrix} 1 & a^2 & a^2+a & a^2 & a & a & a^2+a \\ 0 & a^2+a+1 & a+1 & a+1 & a^2+1 & a^2+a+1 & a^2+1 \end{pmatrix}.$$

Expressing each entry of B as a vector of dimension 3, we get:

$$\mathsf{H}' = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Following the procedure of Example 3.5 as given in Klima et al. (2000), we obtain the generating matrix:

$$\mathsf{G} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

2.4.6 Binary Goppa Codes

We will now consider a special class of Goppa codes, called binary Goppa codes, because the characteristic of the field, where the code has its symbols, is two.

Definition 2.20. *The constraints $K = \mathbb{F}_2$ and $F = \mathbb{F}_{2^m}$ in Definition 2.19, generates binary Goppa codes.*

Proposition 2.1. *(Hoffmann (2011)) If $\Gamma(L, g)$ is a binary Goppa code, such that $g(X)$ is square free, then:*

$$\Gamma(L, g) = \Gamma(L, g^2).$$

Note that the above property causes the minimum code distance to be at least $2 \cdot \delta$, that is to say the correction of δ errors is ensured.

Chapter 3

Courtois, Finiasz and Sendrier (CFS) Signature Scheme

In this chapter, we introduce the CFS scheme (Courtois, Finiasz, and Sendrier, 2001) after reviewing two relevant public key cryptosystems based on error-correcting codes, which are the basis for the CFS scheme. These two cryptosystems are the McEliece (McEliece, 1978) and the Niederreiter schemes (Niederreiter, 1986).

3.1 The McEliece Scheme

This public scheme, which uses error-correcting codes, was invented by McEliece (1978). McEliece used the binary Goppa codes (reviewed in Section 2.4.5). One of the most important property of this scheme is its performance for both encryption and decryption in comparison with another well-known schemes, for example RSA. The main disadvantage is that the sizes of its keys are larger in comparison with the ones used RSA. This fact has motivated scientists in this area to study how to reduce these keys, for example Misoczki and Barreto (2009); Berger, Cayrel, Gaborit, and Otmani (2009); Barbier and Barreto (2011).

The McEliece scheme is based on the difficulty of solving the syndrome decoding problem, which is NP-complete problem.

Next, we describe the algorithms of this scheme.

3.1.1 Key generation

The following algorithms are described using the fictional characters Alice and Bob. Here we are assuming that Bob wants to send a secret message to Alice.

1. Alice selects a binary Goppa code Γ with parameters (n, k) such that it can be correct up to δ errors;
2. Alice computes the generator matrix $\mathsf{G}^{k\times n}$ of the code Γ;
3. Alice selects a random non-singular matrix $\mathsf{S}^{k\times k}$;
4. Alice selects a permutation matrix $\mathsf{P}^{n\times n}$;
5. Alice computes a matrix $\hat{\mathsf{G}} = \mathsf{S}\cdot\mathsf{G}\cdot\mathsf{P}$;
6. The public key of Alice is $(\hat{\mathsf{G}}, \delta)$ and its private key is the triple $(\mathsf{S}, \mathsf{G}, \mathsf{P})$.

3.1.2 Encryption

Assume Bob wants to send a message x to Alice whose public key is $(\hat{\mathsf{G}}, \delta)$:

1. Bob computes $c' = x\hat{\mathsf{G}}$;
2. Bob selects a random vector $e \in \mathbb{F}_2^n$ and such that $\mathrm{wt}(e) = \delta$;
3. Bob computes the ciphertext as $c = c' + e$.

3.1.3 Decryption

At this point, Alice with its private key and the ciphertext c performs the next steps

1. Alice computes the inverse of P, i.e. P^{-1};
2. Alice computes $\hat{c} = c\mathsf{P}^{-1}$;
3. Alice using an efficient decoding algorithm for the code Γ decodes $\hat{c}$ in $\hat{x}$;
4. Alice computes $x = \hat{x}\mathsf{S}^{-1}$.

We can see that the encryption and decryption processes are consistent because of the step (2) of Decryption, $\hat{c} = c\mathsf{P}^{-1} = (c' + e)\mathsf{P}^{-1} = x\hat{\mathsf{G}}\mathsf{P}^{-1} + e\mathsf{P}^{-1} = x\mathsf{S} \cdot \mathsf{G} \cdot \mathsf{P} \cdot \mathsf{P}^{-1} + e\mathsf{P}^{-1} = x\mathsf{S} \cdot \mathsf{G} + e\mathsf{P}^{-1}$. Notice that $\mathrm{wt}(e\mathsf{P}^{-1}) = \delta$ because P is a matrix that permutes the entries of e. In the last expression, applying a decoding algorithm for the code Γ, $\hat{\mathsf{c}}$ is decoded in $\hat{x} = x\mathsf{S}$. Multiplying both sides of the last by S^{-1}, we obtain $\hat{\mathsf{x}}\mathsf{S}^{-1} = x$.

3.2 Niederreiter scheme

Niederreiter (1986) proposed a modification to McEliece scheme, which has an equivalent security (Ding, Wolf, and Yang (2007)). This scheme has a smaller private key in comparison with the McEliece scheme. However, both the encryption algorithm and the decryption algorithm are expensive.

Next, we describe the algorithms of this scheme.

3.2.1 Key generation

1. Alice selects a binary Goppa code Γ with parameters (n, k) and such that it can be correct up to δ errors;
2. Alice computes the parity check matrix $\mathsf{H}^{k\times n}$ of Γ;
3. Alice selects a random non-singular matrix $\mathsf{S}^{k\times k}$;
4. Alice selects a permutation matrix $\mathsf{P}^{n\times n}$;
5. Alice computes the matrix $\hat{\mathsf{H}} = \mathsf{S} \cdot \mathsf{H} \cdot \mathsf{P}$;
6. The public key of Alice is $(\hat{\mathsf{H}}, \delta)$ and its private key is $(\mathsf{S}, \mathsf{H}, \mathsf{P})$.

3.2.2 Encryption

Assume Bob wants to send a message x to Alice whose public key is $(\hat{\mathsf{H}}, \delta)$:

1. Bob encodes the message x as binary vector of length n such that $\mathrm{wt}(x) = \delta$;

2. Bob computes the vector $c = \hat{\mathsf{H}}x^T$.

3.2.3 Decryption

1. Alice computes the inverse of S,i.e. S^{-1};

2. Alice computes $\mathsf{S}^{-1}c = \mathsf{HP}x^T$;

3. Alice uses an efficient decoding algorithm to retrieve $\mathsf{P}x^T$;

4. Alice computes the original message x with $x^T = \mathsf{P}^{-1}\mathsf{P}x^T$.

3.3 CFS Signature

In the 90's, there were attempts to construct signatures schemes using asymmetric cryptography based on codes (Stern (1993, 1994)). The algorithms used in this scheme need a decodable syndrome. Which is a problem in their performance. Courtois et al. (2001) proposed a practical scheme based on codes inspired on the Niederreiter scheme. The signature scheme called CFS solves the above problem allowing to correct an additional number of errors λ. Thus, to decode a syndrome corresponding to a vector with de Hamming weight $\delta + \lambda$, λ random columns are added to the parity check matrix H, and the process of decoding is performed again with the new syndrome. If the decoding process has success, then the decoded vector is used in the signature generation. Otherwise, new columns are generated randomly. This process is repeated until finding a decodable syndrome.

3.3.1 Key Generation

1. Choose a Goppa code $\Gamma(L, g(X))$;

2. Compute both the generator matrix $\mathsf{G}^{k\times n}$ and the parity check matrix $\mathsf{H}^{(n-k)\times n}$ of Γ;

3. Select a random non-singular matrix $\mathsf{S}^{k\times k}$;

4. Select a permutation matrix $\mathsf{P}^{n\times n}$;

5. Compute $\mathsf{V} = \mathsf{SHP}$;

6. The private key consists of G, P and S. The public key is (V, δ).

3.3.2 Signature Generation

1. Find the least $i \in \mathbb{N}$ such that $c = h(m, i)$ and $s = \mathsf{S}^{-1}c$ be a decodable syndrome of the code Γ;

2. Compute the error vector e from s using an efficient decoding algorithm of Γ;

3. Compute $e^T = \mathsf{P}^{-1}e'^T$;

4. The signature is the pair (e, i).

3.3.3 Signature Verification

1. Compute $c = \mathsf{V}e^T$;

2. Verify if $c = h(m, i)$.

In the worst case, the signature is obtained using $t!$ steps.

As we said before, all this review we have done will serve to understand part II of this thesis.

Part II

One-Time Signature Schemes

Chapter 4

Hash-Based Cryptography

We have considered hash-based schemes because studying them, we found an inspiration to create a new one-time signature based on codes.

4.1 Hash-Based One-time Signature Schemes

One-time signatures appeared more than three decades ago and as the name indicates each key-pair of these schemes can only be used to sign a single message. However, as we will see, if combined with Merkle tree, it can be used to sign many messages with a single key, making those more efficient digital signature schemes.

The first one-time signature scheme based on hash and one-way functions that appeared in the literature is the scheme of Lamport (1979). Another old but important one-time signature is the Winternitz scheme (W-OTS). W-OTS was presented in Merkle (1990) and appeared to improve both the signature and key sizes of the Lamport scheme. In the same way as the Lamport scheme, W-OTS can be built from hash and one-way functions.

W-OTS is the inspiration for new one-time signature schemes because it is simple and has smaller signature and key sizes. Several variations of W-OTS with significant improvements in both the security and performance are described in the literature. Regarding the security, Hevia and Micciancio presented a proof in the standard model for several graph-based one-time signatures, including W-OTS. Regard performance, Buchmann, Dahmen, Ereth, Hülsing, and Rückert improve

the signature and key sizes of W-OTS. Besides, this work shows two proofs in the standard model when W-OTS is instantiated using pseudorandom functions (PRF). The first one is the existential unforgeability of W-OTS using PFR, and the second one is the strong unforgeability of W-OTS using second key resistant functions. However, the reduction proofs presented in Buchmann et al. (2011) have flaws according to Lafrance and Menezes (2017). Another improvement in W-OTS, called W-OTS$^+$, is described by Hülsing (2013b), who achieves smaller key and signature sizes than the variation presented in Buchmann et al. (2011). The work of Lafrance and Menezes (2017) does not affect W-OTS$^+$.

In this section, we review the Lamport scheme, W-OTS, and W-OTS$^+$.

4.1.1 Lamport Signature

Lamport (1979) created the first one-time signature scheme studied within hash-based digital signatures. This scheme uses a one-way function and a hash function as primitives. Despite huge key sizes, it supports attacks that could come from quantum computers.

4.1.1.1 Lamport Parameters

Let n be the block size in the Lamport scheme and its security parameter. Given h and g two cryptographic primitives where $h\colon \{0,1\}^n \mapsto \{0,1\}^n$ is a one-way function and $g\colon \{0,1\}^* \mapsto \{0,1\}^n$ is a cryptographic hash function, then the key generation, signature generation and signature verification are as follows.

4.1.1.2 Key-Generation

The signature key consists of $2n$ bit-strings X_i, $0 \leq i \leq 2n-1$, of size n. The verification key consists of $2n$ bit-strings Y_i, $0 \leq i \leq 2n-1$, of size n, such that $Y_i = h(X_i)$.

4.1.1.3 Signature Generation

The signature σ of a message m is calculated with the next steps:

1. Compute the digest of m, i.e. $d = g(m)$;

2. Compute $\sigma_j = X_{2j+d_j}$, $0 \leq j \leq n-1$. Thus, $\sigma = (\sigma_0, \sigma_2, \cdots, \sigma_{n-1})$.

4.1.1.4 Signature Verification

To verify the signature $\sigma = (\sigma_0, \sigma_2, \cdots, \sigma_{n-1})$ of a message m under the verification key Y, one must perform the next steps:

1. Compute the digest of m, i.e. $d = g(m)$;

2. Accept if and only if $h(\sigma_j) = Y_j$, $0 \leq j \leq n-1$.

4.1.1.5 Impossibility of multisigning

Using a toy example, we are going to explain why the key pair of the Lamport scheme cannot be used more than once. Suppose $n = 3$,

$$X = \begin{pmatrix} x_{00} & x_{01} & x_{02} & x_{03} & x_{04} & x_{05} \\ x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{20} & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \end{pmatrix}$$

and

$$Y = \begin{pmatrix} y_{00} & y_{01} & y_{02} & y_{03} & y_{04} & y_{05} \\ y_{10} & y_{11} & y_{12} & y_{13} & y_{14} & y_{15} \\ y_{20} & y_{21} & y_{22} & y_{23} & y_{24} & y_{25} \end{pmatrix}.$$

Let $d = (0, 1, 1)$ be the hash of a message m. Then, according to the signature generation algorithm presented in Section 4.1.1.3,

$$\sigma = \begin{pmatrix} x_{00} & x_{03} & x_{05} \\ x_{10} & x_{13} & x_{15} \\ x_{20} & x_{23} & x_{25} \end{pmatrix}$$

is the signature of d. If we use the same key pair to sign another message m' with hash $d' = (1, 1, 0)$, then the new signature σ' is

$$\sigma' = \begin{pmatrix} x_{01} & x_{03} & x_{04} \\ x_{11} & x_{13} & x_{14} \\ x_{21} & x_{23} & x_{24} \end{pmatrix}.$$

Notice that both signatures σ and σ' reveal pieces of the signature key, then an attacker can sign $d'' = (1, 1, 1)$ without knowing of the signature key, i.e., it is not possible multi-signing.

4.1.1.6 Security of Lamport scheme

In this section, we are going to present a security proof of the Lamport scheme in the standard model following (Bernstein et al., 2008). Specifically, we are going to present a proof where the Lamport scheme, as a one-time scheme, is EUF-1CMA secure.

To achieve EUF-1CMA the one-time function used in the Lamport scheme will be replaced by the following family of one-way functions,

$$\mathcal{F} = \{f_k \colon \{0,1\}^n \to \{0,1\}^n \,|k \in K\},$$

where K is a finite set and its elements are called keys.

Another modification in the key generation is that the Lamport scheme differs slightly from W-OTS and works as follows. On input of 1^n for a security parameter

n a key of K is chosen randomly with the uniform distribution. The key-pair is similar to W-OTS (see Section 4.1.1.2). The single difference is that k is part of the public key.

To show that Lamport scheme is EUF-1CMA, we are going to follow a classic reduction proof presented in Bernstein et al. (2008) that is, we use a forger of the Lamport scheme to construct an adversary $\mathcal{A}_{\mathsf{pre}}$ on the one-wayness of $\mathcal{F}$. Algorithm 1 shows how a forger $\mathsf{For}^{\mathsf{Sign}(\mathsf{sk},.)}(\mathsf{pk})$ for the Lamport scheme can be used to construct that adversary. The adversary simulates the signing oracle.

Algorithm 1 $\mathcal{A}_{\mathsf{pre}}$

Input: $k \xleftarrow{\$} K$ and $y = f_k(x)$, where $x \xleftarrow{\$} \{0,1\}^n$
Output: x' such that $y = f_k(x')$ or $\mathsf{failure}$

1: Generate a key-pair (X, Y) of the Lamport scheme.
2: Choose $a \xleftarrow{\$} \{0, \cdots, n-1\}$ and $b \xleftarrow{\$} \{0,1\}$.
3: Replace $y_a[b]$ by y in the verification key Y of the Lamport scheme.
4: Run $\mathsf{For}^{\mathsf{Sign}(\mathsf{sk},.)}(\mathsf{pk})$.
5: When $\mathsf{For}^{\mathsf{Sign}(\mathsf{sk},.)}(\mathsf{pk})$ asks its only oracle query with $M = (m_{n-1}, \cdots, m_0)$:
6: a) if $m_a = (1-b)$ then sign M and respond to the forger $\mathsf{For}^{\mathsf{Sign}(\mathsf{sk},.)}(\mathsf{pk})$ with the signature σ.
7: b) else return $\mathsf{failure}$.
8: When $\mathsf{For}^{\mathsf{Sign}(\mathsf{sk},.)}(\mathsf{pk})$ outputs a valid signature $\sigma' = (\sigma'_{n-1}, \cdots, \sigma'_0)$ for a message $M' = (m'_0, \cdots, m'_{n-1})$:
9: a) if $m'_a = b$ then return σ'_a as preimage of y.
10: b) else return $\mathsf{failure}$.

The goal of the adversary $\mathcal{A}_{\mathsf{pre}}$ is to obtain a x' such that $y = f_k(x')$ for a given pair k, y provided as input. $\mathcal{A}_{\mathsf{pre}}$ begins by generating a regular key pair of the Lamport scheme, and by choosing a random position a and random binary value b (Lines 1,2). Then $\mathcal{A}_{\mathsf{pre}}$ uses y to replace it in Y using the random elements calculated previously. Specifically, $\mathcal{A}_{\mathsf{pre}}$ replaces $y_a[b]$ by y in the verification key Y of the Lamport scheme (Line 3). Next, $\mathcal{A}_{\mathsf{pre}}$ calls the forger and waits for it to ask an oracle query. As we have said, $\mathcal{A}_{\mathsf{pre}}$ plays the role of signing oracle. The forger query can only be answered if the message does not contain the value b in the position m_a, in case it happens then $\mathcal{A}_{\mathsf{pre}}$ returns failure (Lines 5, 6, 7). The forgery produced by $\mathsf{For}^{\mathsf{Sign}(\mathsf{sk},.)}(\mathsf{pk})$ is only valid if $m'_a = b$, in this case the output of Algorithm 1 is σ'_a, otherwise the output is failure (Lines 8, 9, 10).

We now compute the success probability of $\mathcal{A}_{\mathsf{pre}}$. We denote by ϵ the forger's success probability for producing an existential forgery of the Lamport scheme and by t its running time. By t_{Gen} and t_{Sig} we denote the times the Lamport scheme requires for key and signature generation, respectively. Since b is selected randomly with the uniform distribution, the probability for m_a does not contain the value b in the position a is $1/2$. Since M' must be different from the queried message M, there exists at least one index c such that $m'_c = 1 - m_c$. $\mathcal{A}_{\mathsf{pre}}$ is successful if $c = a$, which happens with probability at least $1/2n$. Hence, the adversary's success probability for finding a pre-image in time $t_{ow} = t + t_{Sig} + t_{Gen}$, is at least $\epsilon/4n$. We have proved the following theorem.

Theorem 4.1. *Let $n \in N$, let K be a finite parameter set, let t_{ow}, ϵ_{ow} be positive real numbers, and $\mathcal{F} = f_k \colon \{0,1\}^n \to \{0,1\}^n \,|k \in K$ be a family of (t_{ow}, ϵ_{ow}) one-way functions. Then the LD-OTS variant that uses F is $(t_{ots}, \epsilon_{ots}, 1)$ existentially unforgeable under an adaptive chosen message attack with $\epsilon_{ots} \leq 4n\epsilon_{ow}$ and $t_{ots} = t_{ow} - t_{Sig} - t_{Gen}$ where t_{Gen} and t_{Sig} are the key generation and signing times of Lamport scheme, respectively.*

4.1.2 Winternitz

The Winternitz scheme (Merkle (1990)) solves the problem of large keys and signature sizes of the Lamport scheme. In the same way as the Lamport scheme, the Winternitz scheme (W-OTS) assumes the existence of a hash function $g \colon \{0,1\}^* \to \{0,1\}^m$ and a one-way function resistant to pre-image $h \colon \{0,1\}^n \to \{0,1\}^n$. Using these primitives, we describe below the W-OTS algorithms.

4.1.2.1 W-OTS Key Generation

Instead of signing a digest bit by bit as in the Lamport scheme, the W-OTS signature algorithm divides the digest into equal parts and then signs these parts. The steps for the key generation are

1. Select an integer $w \geq 2$, which would be the size of the partitions of the

message to be signed;

2. Compute the number of partitions

$$t_1 = \left\lceil \frac{r}{w} \right\rceil;$$

3. Compute

$$t_2 = \left\lceil \frac{\lfloor \log_2 t_1 \rfloor + 1 + w}{w} \right\rceil,$$

which will be used to prevent a type of attack against W-OTS. Let $t = t_1 + t_2$;

4. The signature key is $X = (x_0, x_1 \cdots, x_{t-1}) \in \{0,1\}^{n,t}$, where the bit-strings x_i with $0 \leq i \leq t-1$ are randomly chosen;

5. The verification key Y is computed by applying g $2^w - 1$ times in each bit-string x_i of the signature key, i.e., $Y = (y_0, \cdots, y_{t-1})$ with $y_i = h^{2^w-1}(x_i)$.

The size of a Winternitz signature is roughly mn/w bits and signing roughly requires $2wm/w$ hash operations, where m is the bit length of the hash value to be signed, n is the output length of the hash function used in the scheme, and w is the Winternitz. This last parameter determines a trade-off between signature size and the runtime of signature.

4.1.2.2 W-OTS Signature Generation

Compute the digest d of message m to be signed using g, i.e., $g(m) = d = (d_0, \cdots, d_{n-1})$. Add zeros to d until it is divisible by w. The partition of d into t_1 w-sized parts is

$$d = b_0||b_1|| \cdots ||b_{t_1-1}.$$

Note that each element b_i can be transformed into base-10 representation and identified with integers $\{0, \cdots, 2^w - 1\}$.

Calculate the checksum

$$c = \sum_{i=0}^{t_1-1} 2^w - b_i.$$

The maximum number of bits representing c in binary base is $\lfloor \log_2 t_1 \rfloor + w + 1$. Indeed, note that $c = t_1 2^w - \sum_{i=0}^{t_1-1} b_i \leq t_1 2^w$ and the number of bits required to represent $t_1 2^w$ is $\lfloor \log_2 t_1 2^w \rfloor + 1 = \lfloor \log_2 t_1 \rfloor + w + 1$.

Similarly, split the binary representation of c into t_2 w-sized parts, that is $c = c_0 || c_1 || \cdots || c_{t_2-1}$ and compute the signature as

$$\sigma_{\text{W-OTS}} = (\sigma_0, \cdots, \sigma_{t-1}) = (h^{b_0}(x_0), \cdots, h^{b_{t_1-1}}(x_{t_1-1}), h^{c_0}(x_{t_1}), \cdots, h^{c_{t_2-1}}(x_{t-1})).$$

Why does the signer must sign the checksum c? Suppose a Winternitz scheme with parameter $w = 2$ where the signer does not sign the checksum, then $t_1 = 2$. Using this scheme suppose that we sign the hash $d = 1101$, then $\sigma_{\text{W-OTS}} = (h^3(x_0), h^1(x_1))$. An attacker could forge this scheme intercepting d and changing it to $d = 1111$. Since, h^1 is known the attacker could change $\sigma_{\text{W-OTS}}$ by $\sigma'_{\text{W-OTS}} = (h^3(x_0), h^3(x_1))$, which is a valid signature too.

4.1.2.3 W-OTS Verification

To verify the signature $\sigma_{\text{W-OTS}}$ of the message M, compute the values b_i and c_i in the same way as described above, then compute

$$y' = (h^{2^w-1-b_0}(\sigma_0), \cdots, h^{2^w-1-b_{t_1-1}}(\sigma_{t_1-1}), h^{2^w-1-c_0}(\sigma_{t_1}), \cdots, h^{2^w-1-c_{t_2-1}}(\sigma_{t-1})),$$

and compare y' with Y. The signature is valid only if $y'_i = y_i$, $\forall i$.

4.1.2.4 Security of the Winternitz scheme

There is an extensive security proof against an EU-CMA attack and an SU-CMA attack in the standard model in (Hülsing, 2013a).

4.1.3 W-OTS$^+$

Hülsing (2013b) proposed a type of W-OTS which has two properties: tight and exact. The first property guarantees smaller signatures, and key sizes, in relation to W-OTS (about 50% less than W-OTS). The second property allows computing the exact parameters for the scheme. Also, the second property is guaranteed due W-OTS$^+$ presents a security proof in the standard model.

In the same way as W-OTS, W-OTS$^+$ is parameterized by an integer w which determines a trade-off between the signature size and the runtime of the signature generation. The parameter n determines the security of W-OTS$^+$ and m is the length of the message. W-OTS$^+$ uses two cryptographic primitives g and $\mathcal{F}_n$ where $g\colon \{0,1\}^* \mapsto \{0,1\}^m$ is a cryptographic hash function and $\mathcal{F}$ is a family of keyed functions defined as

$$\mathcal{F}_n = \{f_k\colon \{0,1\}^n \to \{0,1\}^n \,|k \in \mathcal{K}_n\}$$

where $\mathcal{K}_n$ is the key space.

A new function is introduced by Hülsing: $c_k^i(x,r)$. This function on input of value $x \in \{0,1\}^n$, iteration counter $i \in \mathbb{N}$, key $k \in K$ and randomization elements $r = (r_1, \cdots, r_j) \in \{0,1\}^{n\times j}$ with $j \geq i$, works in the following way. In case $i = 0$, c returns x ($c^{0_k}(x,r) = x$). For $i > 0$ c is defined recursively as

$$c_k^i(x,r) = f_k(c_k^{i-1}(x,r) \oplus r_i).$$

As in W-OTS, there are two parameters which allows to sign groups of bits,

$$t_1 = \left\lceil \frac{r}{w} \right\rceil,$$

$$t_2 = \left\lceil \frac{\lfloor \log_2 t_1 \rfloor + 1 + w}{w} \right\rceil,$$

where r is the length of the message to be signed. Another parameter is $t = t_1 + t_2$.

Next, we are going to describe the algorithms of W-OTS$^+$ using fictional characters Alice and Bob.

4.1.3.1 Key Generation

1. Alice selects t n-bit strings x_i, $0 \leq i \leq t$, uniformly at random;
2. Alice selects $w-1$ n-bit strings r_i, $0 \leq i \leq w-2$, uniformly at random;
3. Alice chooses a function key $k \xleftarrow{\$} \mathcal{K}$;
4. The signature key (X, r) consists of the first t selected n-bit strings x_i, i.e. $X = (x_0, \cdots, x_t)$ and the last $w-1$ selected $n-$bit strings r_i. i.e. $r = (r_0, \cdots, r_{w-2})$;
5. The verification key is computed as $Y = (y_0, \cdots, y_l)$ where $y_0 = (r, k)$ and $y_i = c_k^{2^w-1}(x_i, r)$, $1 \leq i \leq l-1$.

4.1.3.2 Signature Generation

To sign the message m with the private key X Alice follows the next steps.

1. Split m into t_1 $w-$sized parts. Let m_i, $0 \leq i \leq w-1$, the $i-th$ $w-$sized part of m. Let α_i, $0 \leq i \leq w-1$, the decimal representation of m_i;
2. Split $a = \sum_{i=0}^{t_1-1} 2^w - b_i$ into t_2 $w-$sized parts. Let a_i, $0 \leq i \leq w-1$, the $i-th$ $w-$sized part of a. Let β_i, $0 \leq i \leq w-1$, the decimal representation of a_i;
3. The signature is

$$\begin{aligned}\sigma_{\text{W-OTS}^+} &= (\sigma_0, \cdots, \sigma_{t-1}) \\ &= (c_k^{\alpha_0}(x_0, r), \cdots, c_k^{\alpha_{t_1-1}}(x_{t_1-1}, r), c_k^{\beta_0}(x_{t_1}, r), \cdots, c_k^{\beta_{t_2-1}}(x_{t-1}, r)).\end{aligned}$$

4.1.3.3 Signature Verification

To verify σ under the public key Y Bob follows the next steps:

1. Repeat the step 1 and the step 2 of the Signature generation;

2. Compute

$$\begin{aligned} y' &= (y'_0, y'_1, \cdots, y'_t) \\ &= ((r,k), c_k^{2^w-1-\alpha_0}(\sigma_0, r_{\alpha_0,w-2}), \\ &\quad \cdots, \\ &\quad c_k^{2^w-1-\alpha_{t_1-1}}(\sigma_{t_1-1}, r_{\alpha_{t_1-1},w-2}), \\ &\quad c_k^{2^w-1-\beta_0}(\sigma_{t_1}, r_{\beta_0,w-2}), \\ &\quad \cdots, c_k^{2^w-1-\beta_{t_2-1}}(\sigma_{t-1}, r_{\beta_{t_2-1},w-2})); \end{aligned}$$

3. Accept if $y_0 = (r,k)$ and if $y'_i = y_i$, $1 \leq i \leq t$.

4.1.3.4 Security of W-OTS$^+$

There is an extensive security proof in the standard model in (Hülsing, 2013a).

4.2 Merkle Signature

The Merkle scheme (Merkle, 1990) lies in the hash-function-based digital signature scheme and uses trees (called Merkle trees) to sign several messages with a single public key. The Merkle scheme employs both a hash function and a one-time signature scheme. The original Merkle scheme has a huge private key and its key-generating and signing time are long. Before 2015, attempts to solve those problems can be found in Buchmann, García, Dahmen, Döring, and Klintsevich (2006a); Buchmann, Dahmen, Klintsevich, Okeya, and Vuillaume (2007); Dahmen, Okeya, Takagi, and Vuillaume (2008); Buchmann et al. (2011); Hülsing (2013b). Despite the proposed advances, a problem remained, namely, the Merkle scheme is stateful,

that is, the signer must keep track of the number of signatures. A stateful scheme cannot perform signatures in parallel. To solve the stateful problem, Bernstein, Hopwood, Hülsing, Lange, Niederhagen, Papachristodoulou, Schneider, Schwabe, and Wilcox-O'Hearn (2015) proposed to use few-time signatures instead of the one-time signature scheme.

Let k be a positive integer which is the security parameter of the Merkle signature. This scheme uses a hash function

$$g\colon \{0,1\}^* \times \mathbb{F}_2^R \to \mathbb{F}_2^k,$$

and any one-time signature as main primitives. Besides the one-time signature explained in the previous section, and those which will be present in the next section, there is a survey Villányi (2010), showing several others one-time signature schemes used in Merkle scheme.

Next, we describe their algorithms using Bernstein et al. (2008) as main reference.

Key Generation. Let $H > 1$ be an integer. Let 2^H be the number of messages that an instance of the Merkle scheme will be able to sign and verify. Then, to generate the public and private keys the signer needs to create 2^H instances of a one-time Signature (OTS) scheme. We will denote the i^{th} signature and verification keys of each instance as X_i and Y_i, respectively. Using Y_i, $0 \leq i \leq 2^H - 1$, the signer generates the public key constructing a binary tree, denoted as Merkle tree. The leaves of the Merkle tree are the digests $g(Y_i)$, $0 \leq i \leq 2^H - 1$. The inner nodes are the hash values of the concatenation of their left and right children. Hence, the root of the Merkle tree is a public key of the scheme. The private key consists of the sequence X_i, $0 \leq i \leq 2^H - 1$. To be more precise, we denote the nodes of the Merkle tree as $v_h[j]$, $0 \leq j \leq 2^{H-h}$, where $h \in \{0, \cdots, H\}$ is the height of the node. Then

$$v_h[j] = g(v_{h-1}[2j] \| v_{h-1}[2j+1]), 1 \leq h \leq H \text{ and } 0 \leq j \leq 2^{H-h}. \tag{4.1}$$

Signature Generation. To sign a message M the signer computes $d = g(M)$. After that, the signer creates an OTS signature σ_{OTS} using the key X_s. Let Y_s the corresponding verification key associated with X_s. Then, for the verifier to be convinced of the validity of Y_s, the signer should include an authentication path A_s, which allows the verifier to construct a path from the leaf $g(Y_s)$ to the root of the Merkle tree. Let h be the height of each node on the path from the leaf $g(Y_s)$ to the Merkle tree root. Then, the authentication path is computed using the following construction:

$$a_h = \begin{cases} v_h[s/2^h - 1], & \text{if } \lfloor s/2^h \rfloor \equiv 1 \mod 2 \\ v_h[s/2^h + 1], & \text{if } \lfloor s/2^h \rfloor \equiv 0 \mod 2 \end{cases}$$

for $h = 0, \cdots, H-1$. Hence, the s-th signature is

$$\sigma_s = (s, Y_s, \sigma_{\text{OTS}}, \overbrace{(a_0, \cdots, a_{H-1})}^{A_s}).$$

Signature Verification. Verifying $\sigma_s = (s, Y_s, \sigma_{\text{OTS}}, \overbrace{(a_0, \cdots, a_{H-1})}^{A_s})$ consists of two steps. In the first step, the verifier validates the one-time signature σ_{OTS} of the digest d using the one-time verification key Y_s and the verification algorithm of the OTS scheme. In the second step, the verifier validates the authenticity of the one-time verification key Y_s by constructing a path $(p_0, \cdots, p_h)$ from the leaf $v_0[l]$ to the root of the modified Merkle tree. The verifier uses the index s, the authentication path $(a_0, ..., a_{h-1})$, and applies the following construction:

$$p_h = \begin{cases} g(a_{h-1} \| p_{h-1}), & \text{if } \lfloor s/2^{h-1} \rfloor \equiv 1 \mod 2 \\ g(p_{h-1} \| a_{h-1}), & \text{if } \lfloor s/2^{h-1} \rfloor \equiv 0 \mod 2 \end{cases}$$

where $h = 1, ..., H$ and $p_0 = g(Y_s)$. The index s is used for deciding in which order the authentication path nodes and the nodes on the path from leaf $g(Y_s)$ to the Merkle tree root are to be concatenated. The authentication of the one-time verification key Y_s is successful if and only if p_H is equal to the modified Merkle

tree root, which is part of the public key.

In modern implementations of the Merkle scheme, it is used several strategies to generate the public and private keys. In the following paragraphs, we describe some of them.

To generate the public key, the *tree hash algorithm* Merkle (1990) is used . This algorithm calculates the root of the Merkle tree efficiently. The algorithm does not store the full hash tree. Instead, successively computes leaves and, whenever possible, computes their parents. To achieve this, the *tree hash algorithm* uses a stack.

In Szydlo (2004), there is a proposal to compute the authentication paths efficiently. To achieve this, the authors use an array of stacks lst. This same array is used in the *tree hash algorithm* to store the first authentication path while the root of the Merkle tree is calculated simultaneously.

To generate the private key usually, the strategy presented in Buchmann, García, Dahmen, Döring, and Klintsevich (2006b) is used. In this work, we called it as *OTS private strategy*. This strategy avoids storing the whole sequence X_i, $0 \leq i \leq 2^H - 1$, as private key. Instead a R-bit seed is used as private key. The first step of that strategy is to choose a R-bit seed SEED_0 uniformly at random, which will be a private key. To generate the whole sequence X_i, it is used SEEDOTS_i, $0 \leq i \leq 2^H - 1$. Each SEEDOTS_i is computed iteratively by using a PRNG function defined as follow:

$$\text{PRNG}(\text{SEED}_i) = (\text{SEED}_{i+1}, \text{SEEDOTS}_i + 1), 0 \leq i \leq 2^H - 1. \tag{4.2}$$

The seed SEEDOTS_i is updated during each call to the PRNG function. The same time that SEEDOTS_i is computed, the new seed SEED_{i+1} is computed too. This last seed will be used for the generation of the next signature key X_{i+1}. The use of the PRNG function in this strategy ensure the forward secrecy property, because in order to calculate X_i only knowledge of SEED_i is necessary.

Chapter 5

Code-based Cryptography

5.1 KKS signature

Kabatianskii et al. (1997) proposed a one-time signature scheme based on the intractability of decoding linear error-correcting codes. This scheme, called KKS, was motived by the lack of signature schemes based on codes and was the first to use *general linear codes* in this area. *General linear codes* help to simplify the KKS construction.

Otmani and Tillich (2011) proposed the best-known attack against KKS, however, as their authors pointed out, such attack does not compromise the security of KKS, when the parameters are carefully selected. We will check these parameters produce key sizes that are bigger than those presented in the original paper. KKS lacks a security proof, and that is why Barreto et al. (2011) created a variation which is secure in the random oracle model. We will review this variation in Section 5.2.1.

In the next paragraphs, we follow the rationale presented by Kabatianskii et al. to construct digital signature schemes based on error-correcting codes. After that, we describe three KKS variations that appeared in their original paper. In the final part of this section, we present the Otmani and Tillich (2011) attack and we also calculate the new parameters of KKS that resist this attack for several security levels.

Analyzing the Niederreiter scheme, one could think a natural way to con-

struct a signature scheme based on codes. First, let us recall the Niederreiter scheme presented in Section 3.2. To encrypt a message m, under the public key $\hat{\mathsf{H}}$, one should encode m in a vector e with Hamming weight t. Then, the ciphertext is $s = \hat{\mathsf{H}}e$.

Let $(\mathsf{P}, \mathsf{H}, \mathsf{S})$ be the private key associated to $\hat{\mathsf{H}}$. Also, let ψ be an efficient decoding algorithm for the code described by H such that it can be correct up to t errors. A natural way to construct a signature scheme inspired in the ideas of Niederreiter is to assume that s is the message we want to sign. Thus, to generate a signature for s one should compute $e = \psi\left(s\mathsf{S}^{-1}\right)\mathsf{P}^{-1}$ and then obtain the message-signature pair (s, e). Obviously, the problem is that not all messages s are decodable with ψ. Then it is necessary to construct an application between the set of messages M and the set of decidable vectors $S_{\mathsf{H}} = \left\{\mathsf{H}e \colon e \in \mathbb{F}_2^{n,t}\right\}$.

A first attempt to construct this application is $\phi'(i) = \hat{\mathsf{H}}\beta(i)$, where $\beta(i)$ is an encoding function, see for example the ranking function defined in Section 2.2. Clearly this construction can be broken easily, because anyone is able to generate a message-signature pair $(s(m), \beta(m))$. Another approach could be to pick randomly an application ϕ from the set of $M!$ applications $M \to S_{\mathsf{H}}$. The problem with this new approach is that it is necessary $M\log(M)$ bits to store ϕ (Cayrel, Otmani, and Vergnaud, 2007).

Kabatianskii et al. (1997) created three ways to construct the application ϕ efficiently. Following the classification of (Cayrel et al., 2007), we present them in the next subsections.

5.1.1 KKS-1

In KKS-1, the application ϕ is constructed in the following way. Given two codes $[N, K, d > 2t] - \mathcal{C}_{public}$ and $[N', K', t] - \mathcal{C}_{hidden}$, where $N > N'$, $\mathcal{C}_{public}$ is a random binary code and $\mathcal{C}_{hidden}$ is a equidistant code (see Section 2.4.1). Let $\mathsf{H} \in \mathbb{F}_2^{N\times(N-K)}$ be the parity check matrix of $\mathcal{C}_{public}$ and let $\mathsf{G} \in \mathbb{F}_2^{N'\times K'}$ be the generator matrix of $\mathcal{C}_{hidden}$. Let $J \in \{0, \cdots, N\}$ such that $|J| = N'$. Let $F_\phi = \mathsf{H}_J\mathsf{G}^T$ be

the matrix representation of ϕ and let $\sigma \in \mathbb{F}_2^N$ such that all entries are zero. To sign a message m, firstly one computes $m\mathsf{G}$ and places the entries of the result on $\sigma \in \mathbb{F}_2^N$ in the positions indicated by J. To accept σ as a valid signature one needs to verify that

$$\mathsf{V}m = \mathsf{H}\sigma. \tag{5.1}$$

Notice that σ is the unique vector with Hamming weight t, otherwise the minimum distance of $\mathcal{C}_{public}$ must be less or equal than $2t$.

The problem to use equidistant codes is that the minimum distance of $\mathcal{C}_{hidden}$ must be $t = 2^{K'-1}$ and its length must be $2^{K'} - 1$ (see Section 2.4.1). These constraints mean that KKS-1 is inapplicable in practice. To show this, suppose that to reduce the size of the messages that will be signed, we use hash functions with a security level of 256-bit. Then $K' = 256$ and $N' = 2^{256} - 1$. Since $N > N'$, both the sizes the signature and public key are too large.

5.1.2 KKS-2

As we have indicated above, the use of equidistant codes is not suitable to construct $\mathcal{C}_{hidden}$ because those codes generate keys of large sizes. To solve it, Kabatianskii et al. proposed to use the dual code $\mathcal{U}$ of a binary BCH code with length $N' = 2^l - 1$ and designed distance $\delta = 2s+1$. With this kind of codes, $t_1 \neq t_2$ but the hardness of solving the equation (5.1) is still guaranteed. Specifically, the values of t_1 and t_2 are the extremes of the expression

$$\left|\mathrm{wt}(u) - \frac{N'+1}{2}\right| \leq (s-1)\sqrt{N'+1},$$

where $u \in \mathcal{U}$. That expression is the Carlitz-Uchiyama bound on the weight of the codewords that belong to BCH codes and was presented in Lemma 2.2.

The Example 1 and Example 2 in (Kabatianskii et al., 1997) show a comparison of the key sizes between KKS-1 and KKS-2, i.e., a comparison of the key sizes using equidistant codes and BCH codes respectively. In that example, the

improvement on the public key size is at least of 10^9.

5.1.3 KKS-3

Another way to reduce even more the sizes of the keys is to choose random linear codes. The next theorem guarantees that the weight of the codewords of random linear codes is inside a certain interval. Thus, the uniqueness of equation (5.1) is still satisfied.

Theorem 5.1 (33, Kabatianskii et al. (1997))**.** *Let $[n', k] - \mathcal{C}$ be a random binary code and $v \in \mathcal{C}$, then*

$$\Pr\left[\frac{n'}{2}(1-\delta) \leq \mathrm{wt}(v) \leq \frac{n'}{2}(1+\delta)\right] = 1 - 2^{-r'+n'H_2(\delta)+1},$$

where $r' = n' - k$, $0 < \delta < 1$ and $H_2(x) = -x\log(x) - (1-x)\log(1-x)$ is the binary entropy function.

Following, we present the algorithms of KKS-3.

5.1.3.1 Key Generation

For the key generation, the signer Alice follows the next steps

1. Alice chooses N, K, n, t_1 and t_2 according to the required security level;
2. Alice draws a random $(N-K) \times N$ matrix H and randomly picks a subset J of $\{1, \cdots, N\}$ of cardinality n;
3. Alice randomly picks a random $k \times n$ generator matrix G that defines a code $\mathcal{C}_{hidden}$ such that, with high probability, $t_1 \leq |u| \leq t_2$ for any non-zero codeword $u \in \mathcal{C}_{hidden}$;
4. Alice computes $\mathsf{V} = \mathsf{H}_J\mathsf{G}^T$ where H_J is the restriction of H to the columns in J;
5. The private key consists of (J, G) and the public key consists of (V, H).

5.1.3.2 Signature Generation

The signature σ of a message $m \in \mathbb{F}_2^k$ of Alice is the unique vector σ of $\mathbb{F}_2^N$ such that $\sigma_i = 0$ for any $i \notin J$ and $\sigma_J = m\mathsf{G}$.

5.1.3.3 Signature Verification

To verify the signature σ of a message m, which was signed under the public key (V, H), Bob checks if $t_1 \leq |\sigma| \leq t_2$ and if $\mathsf{H}\sigma^T = \mathsf{V}m^T$. In case the two conditions are satisfied the signature is valid, otherwise it is rejected.

5.1.4 Security of KKS

There are three kinds of attacks against KKS: 1) Attack that recovers G using an ISD-based algorithm; 2) Attack that obtains a system of equations when KKS is used as a few-time scheme; 3) Attack that uses a sophisticated method presented in Otmani and Tillich (2011).

To recover the private key G is equivalent to decode $\mathcal{C}_{public}$. In fact, the columns of V are linear combinations of the columns of H. Each one of these linear combinations has Hamming weight between t_1 and t_2. Thus, from the point of view of solving $\mathsf{V} = \mathsf{H}_J\mathsf{G}^T$, KKS is resistant against the first attack as long as their parameters are selected carefully.

There are two papers that exploit the weakness of KKS when used as a few-time signature scheme. The first is (Cayrel et al., 2007) and the second is (Barreto et al., 2011). In the next section, we present the attack described by Barreto et al. against a variant of KKS. The attack presented in (Barreto et al., 2011) is an improvement of (Cayrel et al., 2007).

As we have said, Otmani and Tillich (2011) proposed an attack against KKS scheme. However, as their authors pointed out, such attack does not compromise the security of KKS when the parameters are carefully selected. It just points out that the region of weak parameters is much larger than previously thought. Here, we study this attack and which we call Otmani-Tillich attack.

The Otmani-Tillich attack uses the components of the public key, $\mathsf{H}^{R\times N}$ and $\mathsf{V}^{R\times k}$, to construct the parity check matrix $\hat{\mathsf{H}} = \big[\mathsf{H}|\mathsf{V}\big]$. Although, H and V are generated at random, the code described by the parity check matrix $\hat{\mathsf{H}}$ has the following property: there are enough codewords with low Hamming weight, which can be used in algorithms that find low-weight codewords, when using the original parameters.

More specifically, note that the code $\mathcal{C}_{pub}$ described by the parity check matrix $\hat{\mathsf{H}}$ has length $N+k$ and dimension $N+k-R$. Also, note that valid message-signature pairs in KKS satisfy the following fact

Fact 1. *Let (h,σ) a valid message-signature pair where $h \in \mathbb{F}_2^k$ and $\sigma \in \mathbb{F}_2^N$, then*

1. $(h||\,(\sigma))\,\hat{\mathsf{H}}^T = 0.$
2. $t_1 \leq \mathrm{wt}(\sigma) \leq t_2.$

The elements that satisfy both conditions of Fact 1 describe a code $\mathcal{C}_{hidden} \subset \mathcal{C}_{pub}$, which has length $N+k$ and dimension k. Let $J^* \subset [1,\cdots,N]$ be the support of e and set $J' = J \cup J^*$, then

$$\mathcal{C}_{hidden} = \left\{(h||\sigma) \in \mathbb{F}_2^{k+N} \colon h \in \mathbb{F}_2^k, \sigma \in \mathbb{F}_2^N, \sigma_{J'} = h\mathsf{P} + e, \sigma_{[1,\ \cdots,\ N]\setminus J'} = 0\right\}.$$

$\mathcal{C}_{hidden}$ has dimension k and according to Theorem 5.1 the expected Hamming weight of σ is $n/2+n$. Otmani and Tillich claim that $n/2+n$ is much smaller than the total length N. This strongly suggests using well-known algorithms for finding low weight codewords to reveal codewords in $\mathcal{C}_{hidden}$ and therefore valid message-signature pairs. Otmani-Tillich attack uses an algorithm proposed by Dumer (1991a), which is a variation of Stern's algorithm. This algorithm has success because the code described by $\hat{\mathsf{H}}$ does not behave as a random code. In particular:

- There are many low-weight codewords $\mathcal{C}_{hidden} \subset \mathcal{C}_{pub}$;
- The support of the codewords is limited to a minimal subset of positions (of size $n+k$ in the one-time variant);

- Part of the support is already known to the attacker (the rightmost k positions).

The complexity of Otmani-Tillich attack is given by

$$O\left(N^3\right) + O\left(\binom{(K+k+l)/2}{p}\right) + O\left(\frac{1}{2^l}\left(N-K-l\right)^2\binom{(K+k+l)/2}{p}^2\right),$$

where l is an integer ≤ 40 and p is an integer between 1 and 4.

Table 5.2 shows several parameters of KKS so that it resists the Otmani-Tillich attack. The first column represents the work factor of the attack. The second and third columns are the dimensions of G. The fourth and fifth columns are the codimension and the length of H, respectively. The columns p and l refer to the equation (5.1.4). The four last columns are 1) the GV bound, 2) the size of the matrix V, 3) the size of the signature and 4) the work factor of one of the best-known attacks against the syndrome decoding problem (Problem 2.1). The best-known attack is the one presented in (Becker, Joux, May, and Meurer, 2012), which has complexity $O(2^{n/20})$, but for simplicity in its formula, we use the one presented in (Dumer, 1991b).

Table 5.1: Suggested parameters for standard security levels

λ	k	n	$R' = \lvert H \rvert$	N'	l	p	GV	$\lvert\mathsf{V}\rvert$	$\lvert h, \sigma \rvert$	SD_{H}
80	160	380	5000	10000	8	3	6	800160	4044	383
112	224	550	3700	7400	8	5	11	1657824	4717	563
128	256	630	9400	18800	8	5	11	4812800	6930	638
192	384	940	8750	17500	8	6	13	6720000	15382	946
256	512	1280	9450	18900	8	12	24	9676800	11333	1302

5.2 BMS-OTS

Barreto, Misoczki, and Simplicio Jr. (2011) introduced a new one-time signature scheme in a paper entitled "One-time signature scheme from syndrome

decoding over generic error-correcting codes" (BMS-OTS). BMS-OTS is based on the hardness of solving the syndrome decoding problem (SDP), and unlike other signature schemes, based on that problem, BMS-OTS can be instantiated on general linear error-correcting codes, rather than the restricted families such as Goppa codes. This fact is important because the signer and the verifier do not spend resources disguising or exposing the code. This resources reduction contributes to the simplicity of this scheme.

BMS-OTS was inspired in both Schnorr (Schnorr (1990)) and KKS schemes (Kabatianskii et al. (1997)). The Schnorr scheme is based on the difficulty to solve the discrete logarithm problem, and the KKS scheme is based on the difficulty to solve SDP. The signature and verification algorithms of BMS-OTS have steps that are similar to the ones of the Schnorr scheme, see (Schnorr, 1990, p. 4) and (Barreto, Misoczki, and Simplicio Jr., 2011, p. 4). The main difference between BMS-OTS and KKS schemes is that BMS-OTS uses both a hash function and a noise vector, while KKS does not use them.

BMS-OTS uses the Pointcheval-Stern definition (Pointcheval and Stern (1996)) to construct an EUF-1CMA security proof in the random oracle model. The use of this definition is possible because, unlike KKS, BMS-OTS introduces a random error noise and a constraint on the minimum distance of a random binary code, which is part of the verification key. We describe in Section 5.2.6 how these constraints make BMS-OTS meet the Pointcheval-Stern definition.

In the literature, there is an attack against BMS-OTS. This attack is based on the *information set decoding* algorithm (ISD) (Prange (1962)) and consists of using the best-known variation of the ISD algorithm to reveal the signature key, or to retrieve the noise vector. An overview of this attack is presented is Section 5.2.7.

5.2.1 BMS-OTS Parameters

Let K, N, R, n, k, t_1 and t_2 be integers, where $t_1 \leq t_2$, let λ be the security level of BMS-OTS, and let J be a random subset of $\{1, \cdots, N\}$ of cardinality n.

The current difficulty of the best-known attack against BMS-OTS has work factor 2^{λ}

5.2.2 Key Generation

The private key, or signature key, consists of a generator matrix $\mathsf{P} \in \mathbb{F}_2^{k \times n}$ and the set J. P describes a random $[n, k]$-code $\mathcal{C}_{\text{hidden}}$ whose codewords have Hamming weight between t_1 and t_2 with high probability, that is, $t_1 \leq u \leq t_2$, where u belongs to the $\mathcal{C}_{\text{hidden}}$. If we choose the rows of P uniformly from $\mathbb{F}_2$ then by the central limit theorem the weight of the rows of P follows a normal distribution with mean $n/2$ and standard deviation $\sqrt{n}/2$. In practice the authors suggest to use $t_1 = \dfrac{n - 3\sqrt{n}}{2}$ and $t_2 = \dfrac{n + 3\sqrt{n}}{2}$ due to the 3σ rule.

The public key, or verification key, is the pair (H, V) such that $\mathsf{H} \in \mathbb{F}_2^{R \times N}$ is a parity-check matrix of an $[N, N - K, d \geq 4n + 1]$-code called $\mathcal{C}_{\text{pub}}$ and $\mathsf{V} \leftarrow \mathsf{H}_J \mathsf{P}^T \in \mathbb{F}_2^{R \times k}$, where H_J is the restriction of H to the columns in J.

One can see that the consistency of this scheme relies on the difficulty of recovering J. In fact, if J is known then the overdetermined linear system $\mathsf{V} = \mathsf{H}_J \mathsf{P}^T$ is solved in polynomial time.

5.2.3 Signature Generation

To sign a message $m \in \{0, 1\}^*$ under the private key $\mathsf{P} \in \mathbb{F}_2^{k \times n}$, the signer must follow the next steps:

1. Selects $e \in \mathbb{F}_2^N$ randomly, such that $\text{wt}(e) = n$;
2. Computes $s^T \leftarrow \mathsf{H} e^T$;
3. Computes $h \leftarrow f(m, s)$, where $f \colon \mathbb{F}_2^* \to \mathbb{F}_2^k$ is a hash function;
4. Computes $c = y + e$, where y is the unique vector in $\mathbb{F}_2^N$ such that (i) $\mathsf{supp} \subset J$ and (ii) $y_J = h\mathsf{P}$.

The signature is the pair $(h, c) \in \mathbb{F}_2^k \times \mathbb{F}_2^N$. Since the maximum weight of the code generated by P is n, clearly $\text{wt}(h\mathsf{P} + e) \leq \max_h \text{wt}(h\mathsf{P}) + \text{wt}(e) = 2n$,

and hence the legitimate signatures must satisfy $\mathrm{wt}(c) \leq 2n$. One can see that the signature size is $k + N$ bits. However, due to the compact representation theory presented in Section 2.2, it is possible to reduce that size to $k + \log \binom{N}{2n} + \log(2n)$ bits.

5.2.4 Signature Verification

To verify a signature $(h, c) \in \mathbb{F}_2^k \times \mathbb{F}_2^N$ of a message $m \in \{0,1\}^*$ under the public key $(\mathsf{H}, \mathsf{V}) \in \mathbb{F}_2^{R \times N} \times \mathbb{F}_2^{R \times k}$, the verifier checks that $\mathrm{wt}(c) \leq 2n$, computes $s^T \leftarrow \mathsf{H}c^T + Vh^T$, $v \leftarrow f(m, s)$, and accepts $(h, c) \in \mathbb{F}_2^k \times \mathbb{F}_2^N$ if and only if $v = h$.

Note that the consistency of this scheme is established by the fact that, by definition of c, s and V, we have $\mathsf{H}c^T = \mathsf{H}(y + e)^T = \mathsf{H}_J\mathsf{P}^T h^T + \mathsf{H}e^T = \mathsf{V}h^T + s^T$. Thus, $s^T = \mathsf{H}c^T + \mathsf{V}h^T$ as expected, so it follows necessarily that $v = f(m, s) = h$.

5.2.5 The impossibility of multisigning

The BMS-OTS parameters and notations presented at the beginning of this chapter are different from those presented in Barreto et al. (2011), in fact, we used the parameters and notations presented in Otmani and Tillich (2011). Next, using these new notations, we present an analysis of the impossibility of multisigning in BMS-OTS due to Barreto et al. (2011).

If an attacker knows the set J, then the attacker can to solve the overdetermined linear system $\mathsf{H}_J\mathsf{P}^T = \mathsf{V}$ to retrieve P. Barreto et al. (2011) presented an analysis to get J under the assumption that the same key-pair is used to sign l messages. In the following paragraphs, we describe this analysis using statistics.

First, suppose that in the step 4 of the signature generation, no noise vector is added to $h\mathsf{P}$, i.e. $c_J = h\mathsf{P}$. That is, each non-zero element in c reveals a position in J. Since h is the result of f evaluated in (m, s), each element of c is non-zero with probability $1/2$, and thus each column reveals, on average, half of the still unknown elements of J. Therefore, under these facts an attacker needs $\ln(n) + 1$ valid signatures to recover P.

In the last paragraph, we suppose that no noise vector is added to $h\mathsf{P}$, however, in Barreto et al. (2011), the opposite happens. Next, we calculate how many signatures l (at least) are necessary to recover J when a noise vector e is added to $h\mathsf{P}$. This calculation is due to Barreto et al. (2011), who used a single position of J to simplify the problem.

The goal of Barreto et al. (2011) is to find an expression in function of l using a counting procedure. The expression is obtained when the average of the chosen signatures revealing an entry of J is greater than the average of the chosen signatures falling outside J. First, we describe how to calculate the number of times that an element of $h\mathsf{P}$ is in average 1. Next, we describe how to calculate the number of times that an element of e is in average 1 too. After that, we calculate the number of times that the combined contributions reveal or do not reveal an entry of J.

Let X_i, $0 \leq i \leq l-1$ be random variables that describe the outcomes of an entry of $c = y + e$. Note that the j-th entry of $h\mathsf{P}$ contributes with probability $1/2$ to the j-th entry of c to be 1; and the j-th entry of e contributes with probability n/N to the j-th entry of c to be 1. By the central limit theorem, the sum of l independent variables that are randomly sampled with probability δ has mean $l\delta$ and standard deviation $\sqrt{l\delta(1-\delta)}$. Thus, the number of times that an element from $h\mathsf{P}$ assumes in average the value 1 has mean $\mu_{\mathsf{P}} = l/2$ and standard deviation $\sigma_{\mathsf{P}} = \sqrt{l}/2$. And, the number of times that an element from e assumes in average the value 1 has mean $\mu_e = ln/N$ and deviation standard $\sigma_e = \sqrt{ln/N(1-n/N)}$.

As we have said, the goal of Barreto et al. (2011) is to find an expression as a function of l. That expression comes from the following sentence: The average of the chosen signatures revealing an entry of J is greater than the average of the selected signatures falling outside J. We denote the average of the chosen signatures revealing an entry of J as A and the average of the chosen signatures falling outside J as B.

To obtain l, the counting procedure consists in counting the number of sig-

natures until revealing all columns of J in average, that is, $A(l) > B(l)$, where $A(l)$ is the minimum number of signatures revealing all columns of J, and $B(l)$ is the maximal number of signatures that does not reveal J. Let m_0 be the number of standard deviations. Then, $A(l) = (\mu_{\mathsf{P}} - m_0\sigma_{\mathsf{P}}) - (\mu_e + m_0\sigma_e)$ and $B(l) = \mu_e + m_0\sigma_e$. Reducing the expression $A(l) > B(l)$, we obtain

$$l \geq m_0^2 \left(\frac{1 + 4\sqrt{(n/N)(1 - n/N)}}{1 - 4n/N} \right)^2 .$$

5.2.6 EUF-1CMA security

The authors of BMS-OTS said that they were inspired by Schnorr scheme. The latter scheme has a proof in the random oracle model, which was developed by Pointcheval and Stern (1996). To formalize this proof, they modified the Schnorr scheme slightly, and also created a class of signature schemes called Pointcheval-Stern. The definition of this scheme is used in BMS-OTS to obtain a proof in the random oracle model. The following paragraphs describe this definition and why BMS-OTS achieves that definition.

Definition 5.1 (Pointcheval-Stern)**.** *Given a message m, the Pointcheval-Stern scheme produces a triple (σ_1, h, σ_2). This triple needs to meet the following rules. σ_1 must be randomly sampled from a large set, h must be the hash value of (m, σ_1), and σ_2 only depends on σ_1, the message m, and h.*

In the case of BMS-OTS, the triple is (s, h, c). s is clearly sampled from a large set $\binom{N}{n}$ and h is indeed the hash value of (m, s). For the third rule, c depends on s through of e because there is a unique e of weight n for a given valid s due to the fact that the minimum distance of the code is $d > 4n + 1$. Thus, BMS-OTS meets the Pointcheval-Stern definition. To show that there is a unique e such that $e\mathsf{H}^T = s$ and $\mathrm{wt}(e) = n$, one can suppose that there exist another e' such that $e'\mathsf{H}^T = s$ and $\mathrm{wt}(e') = n$. In this case, we have $(e + e')\mathsf{H} = 0$, that is, $e + e' \in \mathcal{C}_{\mathrm{pub}}$.

Because the Hamming weight of both e and e' is n, the vector $(e+e')$ has Hamming weight at most $2n$, but this is contradictory with $d > 4n + 1$.

Another great contribution to cryptography presented in Pointcheval and Stern (1996) is the Forking lemma. This lemma helps to demonstrate that the Pointcheval-Stern signature schemes are provably secure in the random oracle model. Below, we present that lemma and also, we describe how it is used to prove that BMS-OTS is provably secure in the random oracle model.

Lemma 5.1 (The Restricted Forking Lemma)**.** *Let $\mathcal{A}_{\mathsf{pre}}$ be a probabilistic polynomial time turing machine, given only the public data as input. Let q_H be the number of queries that $\mathcal{A}_{\mathsf{pre}}$ can ask to the random oracle. If $\mathcal{A}_{\mathsf{pre}}$ can find, with non-negligible probability, a valid signature $(m, \sigma_1, h, \sigma_2)$ within time bound T and probability $\epsilon \geq 7q_H/2^{\lambda}$, then there is another machine which has control over $\mathcal{A}_{\mathsf{pre}}$ and produces two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2)$ with non-negligible probability, with the same random tape and a different oracle such that $h \neq h'$ in expected time $T' \leq 84480q_HT/\epsilon$.*

Proof. (Pointcheval and Stern, 2000, p. 16)

Collorary 1. *In the conditions of the Restricted Forking Lemma, for any given l, there is a machine A_l that can produce l valid signatures $(m, \sigma_1, h_j, \sigma_2, j)$, $j = 1 \cdots l$, such that h_j are all distinct, in expected time $T' \leq 84480lq_HT/\epsilon$.*

Proof. (Barreto et al., 2011, p. 8)

Theorem 5.2 (The General Forking Lemma)**.** *Let $\mathcal{A}_{\mathsf{pre}}$ be a probabilistic polynomial time Turing machine whose input only consists of public data. Let q_H and q_S be the number of queries that $\mathcal{A}_{\mathsf{pre}}$ can ask to the random oracle and the number of queries that $\mathcal{A}_{\mathsf{pre}}$ can ask to the signer, respectively. Assume that, within a time bound T, $\mathcal{A}_{\mathsf{pre}}$ produces, with probability $\epsilon > 10(q_S + 1)(q_S + q_H)/2\lambda$, a valid signature $(m, \sigma_1, h, \sigma_2)$. If the triples (σ_1, h, σ_2) can be simulated without knowing the secret key, with an indistinguishable distribution probability, then there is another*

machine which has control over the machine obtained from $\mathcal{A}_{\mathsf{pre}}$ *replacing interaction with the signer by simulation and produces two valid signatures* $(m, \sigma_1, h, \sigma_2)$ *and* $(m, \sigma_1', h', \sigma_2')$ *such that* $h \neq h'$*in expected time* $T' \leq 6120686 q_H T/\epsilon$.

Proof. (Pointcheval and Stern, 2000, p. 19).

The following theorem states the BMS-OTS is secure against no-message attacks. i.e. attacks where the adversary can query the hash oracle but not the signer oracle.

Theorem 5.3. *Assume that, within a time bound* T*, an attacker* $\mathcal{A}_{\mathsf{pre}}$ *performs an existential forgery under a no-message attack against BMS-OTS, with probability* $\epsilon > 7qH/2\lambda$ *where* q_H *denotes the number of queries that* $\mathcal{A}_{\mathsf{pre}}$ *can ask to the random oracle. Then the overdetermined linear system* $\mathsf{V} = \mathsf{H}_J \mathsf{P}^T$ *can be solved in expected time* $T' \leq 684480 l q_H T/\epsilon$ *where* $l = O(\lambda)$.

Proof. (Barreto et al., 2011, p. 9)

In the General Forking Lemma, it is required that triples (s, h, c) can be simulated without knowing the private key P. The following indistinguishability result claims that.

Theorem 5.4. *The triples* (s, h, c) *of the proposed scheme can be simulated without knowing the private key* P*, in the sense of being indistinguishable from legitimate triples unless the adversary is able to solve the overdetermined linear system* $\mathsf{V} = \mathsf{H}_J \mathsf{P}^T$.

Proof. (Barreto et al., 2011, p. 9).

Finally, the next theorem shows that BMS-OTS is EUF1-CMA.

Theorem 5.5. *Let* $\mathcal{A}_{\mathsf{pre}}$ *be an attacker which performs, within a time bound* T*, an existential forgery under a one-chosen-message attack against BMS-OTS with probability* $\epsilon > 20(1 + q_H)/2\lambda$ *where* q_H *denotes the number of queries that* $\mathcal{A}_{\mathsf{pre}}$ *can ask to the random oracle. Then the overdetermined linear system* $\mathsf{V} = \mathsf{H}_J \mathsf{P}^T$ *can be solved within expected time* $T' \leq 6120686 l q_H T/\epsilon$ *where* $l = O(\lambda)$.

Proof. (Barreto et al., 2011, p. 10).

5.2.7 Security of BMS-OTS

Like KKS, a ISD attack can be used against BMS-OTS to recover the signature key G, or to retrieve e. The structural attack presented in Otmani and Tillich (2011) cannot be used against BMS-OTS. In fact, we would have to look for codewords (σ, h) such that

1. $\mathsf{H}\sigma^T + \mathsf{F}h^T = \mathsf{H}e^T$;
2. $\mathrm{wt}(\sigma) \leq 2n$ and $\mathrm{wt}(e) = n$.

Let us assume that we want to forge a signature for a message $x \in \{0,1\}^*$. One has to solve points 1 and 2 but in the random oracle model, finding e given h is hard (pre-image resistance of f and $\mathrm{wt}(e) = n$). Also, if we first choose e, then letting $s = \mathsf{H}e^T$ and applying the attack by considering the following problem

1. $\hat{\mathsf{H}}x' = s$ where $\hat{\mathsf{H}} = (\mathsf{H}|\mathsf{F})$ and $x' = (\sigma, h)$;
2. $\mathrm{wt}(\sigma) \leq 2n$.

It will not be sufficient because (σ, h) has to satisfy the verification equation $h = f(x, \mathsf{H}\sigma^T - \mathsf{F}h^T)$. The only way to deal with this problem is to compute $h = f(x, \mathsf{H}e^T)$ and to set $z = \mathsf{F}h^T + \mathsf{H}e^T$, and then to search for σ such that

1. $\mathsf{H}\sigma^T = z$;
2. $\mathrm{wt}(\sigma) \leq 2n$.

This is a classical syndrome decoding problem. Probably, this scheme needs to have their parameters updated to avoid attacks based on the ISD algorithm, as the one presented in Becker et al. (2012).

5.3 A new one-time signature

In this section, we present the first of our contributions, a new one-time signature. The motivation for this contribution is the small number of digital signatures based on codes and one-time that are hash free. In fact according

our knowledge the last one-time signature with these features was proposed in (Kabatianskii et al., 1997; Barreto et al., 2011). As we have studied, the one-time signatures based on codes (Kabatianskii et al., 1997) basically maps the message space to the set $S_{\mathsf{H}} = \left\{\mathsf{H}\sigma \colon \sigma \in \mathbb{F}_2^{n,t}\right\}$, where H is a parity check matrix of a linear random code $\mathcal{C}$ and $t = w$ in the case of KKS and $t = 2w$ in the case of BMS-OTS. Then, the signature of a message m is a vector σ with Hamming weight less or equal than an integer w in the case of KKS (and less or equal than $2w$ in the case of BMS-OTS). To verify the signature, one verifies if σ has Hamming weight less or equal than w in the case of KKS (or if σ has Hamming weight less or equal than $2w$ in the case of BMS-OTS). Also, one verifies if $\mathsf{V}m = \mathsf{H}\sigma$ in the case of KKS (or if $s = \mathsf{V}m + \mathsf{H}\sigma$ in the case of BMS-OTS). As we have seen, both constructions fall in the hardness of solving the linear syndrome decoding problem, which is proved to be NP-complete.

A disadvantage of the one-time signatures based on codes cited above is their large public key when random matrices are used in its construction. This happens because three matrices are necessary to construct the map between the message space and the set $S_{\mathsf{H}} = \left\{\mathsf{H}e \colon e \in \mathbb{F}_2^{n,t}\right\}$. Two of these matrices make up the public key, and the other, the secret key. Another disadvantage is that the existing one-time signature schemes use hash function primitives. Thus, the security proof of these schemes can be developed only in the random oracle model.

The aforementioned disadvantages motivated us to propose a new hash-free one-time signature. In this proposal, only one matrix is necessary as public key. The secret key is also a matrix, but a sparse one. To achieve this and for a selected parameter t, we modify the set S_{H} by

$$S'_{\mathsf{H}} = \left\{e_i \in \mathbb{F}_2^{n,w} \colon e_0\mathsf{H} = e_1\mathsf{H} = \cdots = e_{2t-1}\mathsf{H}\right\}.$$

Being a Lamport scheme, the drawback of our construction is the large signature sizes, but in compensation a secure proof in the standard model could be done.

5.3.1 Parameters

Choose the integers L, n, r, w' and w such that $Lw' \leq w$, and chose a security parameter λ such that the actual difficulty of $\mathrm{SDP}(Ln, Lr, w)$ meets the level 2^{λ}. Let g be a hash function where $g\colon \{0,1\}^{*} \to \{0,1\}^{t}$.

5.3.2 Key Generation

1. Construct a matrix with the following structure

$$\mathsf{H} = \begin{bmatrix} \mathsf{M}_{0,0} & \mathsf{0} & \dots & \mathsf{0} \\ \mathsf{M}_{1,0} & \mathsf{M}_{1,1} & \dots & \mathsf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{M}_{L,0} & \mathsf{M}_{L,1} & \dots & \mathsf{M}_{L-1,L-1} \end{bmatrix}$$

   Here $\mathsf{M}_{i,j}$, $0 \leq i, j \leq L-1$, is a random matrix of dimensions $r \times n$ such that $n \gg r$.

2. Generate L random vectors s_i, $0 \leq i \leq L-1$.

3. Using the rows of H construct several linear, and underdetermined, systems in the following way.

   - For the first-row construct $\mathsf{M}_{0,0}x = s_0$ and compute $2t$ solutions $\theta_0 = \left\{a_0^0, a_1^0, \cdots, a_{2t-1}^0\right\}$ such that $\mathrm{wt}(a_k^0) \leq w'$, $0 \leq k \leq 2t-1$.
   - For the second-row construct $2t$ linear, and underdetermined, systems using the previously calculated solutions. Specifically, using a_0^0 construct the linear, and underdetermined, system $\mathsf{M}_{1,1}x = s_1 + \mathsf{M}_{1,0}a_0^0$. Using a_1^0 construct the linear, and underdetermined, system $\mathsf{M}_{1,1}x = s_1 + \mathsf{M}_{1,0}a_1^0$. Repeat this process for the remained solutions. Denote $\theta_1 = \left\{a_0^1, a_1^1, \cdots, a_{2t-1}^1\right\}$ as the set where a_0^1 is a solution of $\mathsf{M}_{1,1}x = s_1 + \mathsf{M}_{1,0}a_0^0$ with $\mathrm{wt}(a_0^1) \leq w'$, a_1^1 is a solution of $\mathsf{M}_{1,1}x = s_1 + \mathsf{M}_{1,0}a_1^0$ with $\mathrm{wt}(a_1^1) \leq w'$, and so on.
   - Repeat the previous step for the remained rows.

4. Select a singular matrix $\mathsf{S} \xleftarrow{\$} \mathbb{F}_2^{Lr \times Lr}$.

5. Compute $\hat{\mathsf{H}} = \mathsf{SH}$.

6. Compute $\hat{s} = \mathsf{S}s$, where $s = s_0||s_1|| \cdots s_{L-1}$.

7. Permute $\hat{\mathsf{H}}$ and $\hat{s}$ in the same row positions. Let $\bar{\mathsf{H}}$ the output of the permutation on $\hat{\mathsf{H}}$. Let $\bar{\mathsf{s}}$ the output of the permutation on $\hat{s}$.

8. Sort the elements of θ into the colexicographic order. Let $\hat{\theta} = \{b_0, \cdots, b_{2t-1}\}$ the output of the sorted θ.

The signature key is $\hat{\theta}$ and verification key consist of the pair $(\bar{s}, \bar{\mathsf{H}})$ and

$$Y = \{y_0, y_1, \cdots, y_{2t-1}\} = \{\mathrm{wt}(b_0), \mathrm{wt}(b_1), \cdots, \mathrm{wt}(b_{2t-1})\}.$$

5.3.3 Signature generation

Let m be a message to be signed with the signature key $\hat{\theta}$:

1. Compute the digest d of the message m, i.e. $g(m) = d$. Hereafter, we denote the k-th bit of d as d_k.

2. Compute the signature $\sigma = (\sigma_0, \sigma_2, \cdots, \sigma_{t-1})$ where

$$\sigma_k = b_{d_k}, 0 \leq k \leq t-1.$$

5.3.4 Signature verification

To verify the signature $\sigma = (\sigma_0, \sigma_1, \cdots, \sigma_{t-1})$ of a signed message m using the verification key $(\bar{\mathsf{H}}, \bar{s})$, we must follow the next steps:

1. Compute $d = g(m)$.

2. Verify that $\sigma_k \bar{\mathsf{H}}^T = \bar{s}$, $0 \leq k \leq t-1$.

3. Verify $\mathrm{wt}(\sigma_k) \stackrel{?}{=} y_{d_k}$, $0 \leq k \leq t-1$.

4. Verify that the elements of σ of the message m are sorted in colexicographic order.

Notice that it is necessary the inclusion of the Hamming weights of each entry b_i, i.e., Y. In fact, without these values any message m' can be replaced by m. Also note that if we want to sign a message with two equal contiguous bits an attacker can flip one of these bits and the signature verification algorithm would correctly verify this new message. To avoid this attack, we need to ensure that $\mathrm{wt}(a_i) \neq \mathrm{wt}(a_j)$, for $i \neq j$.

5.3.5 Running Time Complexity and Storage Space Complexity

First, we look at the sizes. A signature contains t vectors of length Ln and Hamming weight Lw'. Thus, using the compact theory representation presented in Section 2.2, we have that the signature size is

$$s_{\mathsf{Sig}} = t \log \binom{Ln}{Lw'} \tag{5.2}$$

bits. The verification key contains $\bar{s}$ and $\bar{\mathsf{H}}$. $\bar{s}$ has Lr bits. $\bar{\mathsf{H}}$ can be represented in a systematic form. This fact allows to use $L^2r(n-r)$ bits rather than L^2rn. Suppose that the number of bits used to represent an integer is s_{Y}, then Y has size $2tc_{\mathsf{Y}}$ bits. Thus, the number of bits of the verification key is

$$s_{\mathsf{PK}} = Lr + L^2(n-r)r + 2ts_{\mathsf{Y}}.$$

Last, the signature key s_{SK} contains $2t$ Ln-bit vectors. Using the compact representation theory, we have s_{SK} is $2t \log \binom{Ln}{Lw'}$.

For the running times, we only look at the worst cases. Let c_{PRNG} be the cost of generating a random element in $\mathbb{F}_2$, then the cost to generate H is $O(c_{\mathsf{PRNG}}Lrn)$. To find the $2t-1$ solutions of each linear underdetermined sub-system $s_i = x\mathsf{M}_i$, we use (Niebuhr, Cayrel, and Buchmann, 2011). This method is a variation of the generalized birthday attack (Wagner, 2002) and works fine with instances of SDP

when it has several solutions. This method has complexity of $O(2^{a+\frac{r}{a+1}})$, where $2^a = w'$.

Let c_{F} be the cost of an arithmetic operation over $\mathbb{F}_2$, then the cost c_{S} to generate S is $O(c_{\mathsf{F}}(Lr)^2 n)$. The matrix-matrix multiplication SH costs $O(L^2 r^2 n)$. The cost of the matrix-vector multiplication $\mathsf{S}s$ is $O(L^2 r^2)$. The cost to permute the rows of $\hat{\mathsf{H}}$ and $\hat{s}$ is $O(Lr)$. The cost to sort the t solutions in lexicographic order is $O(2tLn)$.

Thus, the key generation time is

$$\begin{aligned} c_{\mathsf{Kg}} &= O(c_{\mathsf{PRNG}} Lrn) + O(L(2t-1)2^{a+\frac{r}{a+1}}) + O(c_{\mathsf{F}}(Lr)^2 n) \\ &\quad + O(L^2 r^2 n) + O(L^2 r^2) + O(Lr) + O((2t-1)Ln) \\ &= O(L(2t-1)2^{a+\frac{r}{a+1}}). \end{aligned} \tag{5.3}$$

For the signature generation, we need a call to the hash function and construct the signature selecting some elements of $\hat{\theta}$. Thus, the signature generation is given by

$$c_{\mathsf{Sig}} = O(c_{\mathsf{g}}), \tag{5.4}$$

where c_{g} is the cost to compute the digest d. For the signature verification, 1) we need a call to the hash function, 2) t matrix-vector multiplications, 3) t integer comparisons and 4) verify that σ is sorted in lexicographic order. Thus, the total cost c_{Ver} of the signature verification is given by

$$\begin{aligned} c_{\mathsf{Ver}} &= O(c_{\mathsf{g}}) + O(L^2 nr) + O(1) + O(t) \\ &= O(c_{\mathsf{g}} + L^2 nr). \end{aligned} \tag{5.5}$$

5.4 Security

We begin by showing that our proposal cannot be a multisigning scheme. After, we show which types of attacks can be used against our proposal.

5.4.1 Impossibility of multisigning

Notice that like the Lamport scheme a signature in our proposal is part of the private key. This enable us to perform the next attack, for example. Let $t = 4$. Suppose the signer signs two messages with digests $d_1 = (1, 0, 1, 1)$ and $d_2 = (1, 1, 1, 0)$ using the same signature key. The signatures of these digests are $\sigma_1 = (x_3[1], x_2[0], x_1[1], x_0[1])$ and $\sigma = (x_3[1], x_2[1], x_1[1], x_0[0])$, respectively. Then an attacker knows $x_3[1], x_2[0], x_2[1], x_1[1], x_0[0], x_0[1]$ from the signature key. The adversary can use this information to generate valid signatures for messages with digests $d_3 = (1, 0, 1, 0)$ and $d_4 = (1, 1, 1, 1)$. This example can be generalized to arbitrary security parameters. Thus, the adversary is not able to generate valid signatures, as long as the parameters to construct H can be carefully selected and the key-pair can be used only one time.

5.4.2 Best Known Attacks against our Proposal

Notice that our construction is based on the hardness of solving the linear system $\bar{\mathsf{H}}x = \bar{s}$, where $\mathrm{wt}(x) \leq Lw$, i.e., our construction falls in the SDP. As we have said, the best-known ISD attack against SDP at the moment is considered to be (Becker et al., 2012). One usually measures the complexity of decoding algorithms asymptotically in the code length (Becker et al., 2012). In this way, the complexity of this method against random linear codes is $O(2^{0.0494Ln})$. For simplicity, in its general complexity formula, we to use the method proposed in (Dumer, 1991b). This method has an asymptotic complexity of $O(2^{0.0679Ln})$ but, its general complexity formula is

$$\mathrm{WF}_{\mathrm{SD}} = \frac{\binom{n}{w}}{\binom{n-k-l}{w-p}\sqrt{\binom{k+l}{p}}} \tag{5.6}$$

where

$$l = \log \sqrt{\binom{k+l}{p}} \tag{5.7}$$

The linear system $\bar{\mathsf{H}}x = \bar{s}$ with the restriction $\mathrm{wt}(x) \leq w$ is underdetermined

and has several solutions since $\binom{N}{w} > 2^R$. As we have said, an attack for this kind of system is the generalized birthday attack (GBA) (Wagner, 2002). This attack has complexity $O(2^{a+\frac{Lr}{a+1}})$, where $a = \log(w)$.

5.4.3 Parameters

According to equation (5.2), if we fix w' and allow n to grow, we can get smaller signatures. Also, note that if the values of n and r are too big, then the signer will spend a lot of time to obtain θ_i. This happens because to find at least one value of θ_i falls in SDP. In its turn, if the values of n and r are small, the sizes of the signature key and the signature will be too big. In fact, suppose we want to sign a 128-bit message m, and also we want a security level of 80-bit against both the GBA attack and the ISD variation attack (Dumer, 1991b), then for GBA we need

$$a + \frac{Lr}{a+1} = 80 \tag{5.8}$$

with $w = 2^a$ (Niebuhr et al., 2011), and for the ISD attack, we need to satisfy equation (5.10). According to (Finiasz and Sendrier, 2009) a must be approximately greater than 6 to achieve an 80-bit security level against GBA, thus from (5.8), $L > 549/r$. If r is the smallest possible, i.e., $r = 1$, then $L > 549$. To minimize w we need to set $w' = 1$. Thus, $w > 549$. To achieve a security level of 80-bit against the attack of Dumer (1991a), we need to choose the smallest L, i.e. $L \approx 549$. Thus, $n \approx 11$. Using the compact representation theory, we will need about $\log_2 \binom{5490}{549} \approx 2569$ bits to sign each bit of m. That is, for a security level of 80-bit, the signature size will be at least 328830 bits, and the key size of the signature will be at least 657660 bits. A similar behavior is expected with other practical security levels.

For the cited reasons, the choice of the parameters of the system should be done with great care because a bad choice can strongly affect the performance of the system. One should first choose the security required λ. Then, the choice of the parameters must follow the next simple rules. First, we must ensure that the

expected number of solutions of $\bar{\mathsf{H}}x = \bar{s}$ must be at least $2t$, i.e., we must ensure that $2^{-r}\binom{n}{w'} \geq 2t$. Also, we need to ensure that the running time complexity of GBA to compute the $2t$ solutions is feasible. After that, we must ensure that the selected parameters resist GBA, i.e.,

$$a + \frac{Lr}{a+1} \geq \lambda \tag{5.9}$$

with $w = 2^a$ (Niebuhr et al., 2011). Finally, we must ensure that the selected parameters resist the ISD attack. That is

$$\mathrm{WF}_{\mathrm{SD}} \geq \lambda. \tag{5.10}$$

After testing several values and in order to achieve a security level of 80-bit, we found that the parameters $w' = 8$, $r = 41$, $L = 14$, $n = 428$ are possibly the best ones. Thus, $N = 5992$, $R = 574$, $w = 112$.

Table 5.2 suggests parameters for several practical security levels. The parameters are selected taking into account the shortest signature keys.

Table 5.2: Suggested Parameters for several security levels

λ	n	r	w'	L	|sk|	|pk|	|sig|	ISD	GBA
80	428	41	8	14	141805	3126578	70902	371	80
112	422	54	16	19	330040	7174818	165020	934	120
128	422	54	16	21	214978	9122982	107489	1034	130
192	470	67	16	27	481497	19685538	240748	1231	194
256	668	92	16	26	487323	35824984	243661	1168	267

Tables 5.3 and 5.4, show a comparison between our proposal and other code-based one-time signatures at 80-bit security level. Specifically, Table 5.3 shows a comparison of our proposal with KKS and BMS-OTS. In this comparison, we assume that $\mathcal{C}_{public}$, of both KKS and BMS-OTS, is implemented with double-

circulant codes. Table 5.4, shows a comparison of our proposal with KKS and BMS-OTS, too. This time it is assumed that $\mathcal{C}_{public}$, of KKS and BMS-OTS, is implemented using random codes.

As you can see, when $\mathcal{C}_{public}$ of KKS and BMS-OTS is implemented with double-circulant codes, they have better key and signature sizes. If $\mathcal{C}_{public}$ of KKS, and respectively BMS-OTS, is implemented with random codes, the size of the verification key of our proposal is less than the verification key of KKS and BMS-OTS. Nevertheless, the size of the signature key and the private key of KKS, and BMS-OTS, are less than our proposal.

Table 5.3: Comparing our proposal with other coding-based one-time signatures that use double-circulant codes.

scheme	$\|sk\|$ bits	$\|pk\|$ bits	sig	hash free
KKS	2726	176900	4042	no
BMS-OTS	1288	496363	2062	no
Our	141805	3126578	70902	yes

Table 5.4: Comparing our proposal with other coding-based one-time signatures that use random codes.

scheme	$\|sk\|$ bits	$\|pk\|$ bits	sig	hash free
KKS	2726	25000000	4042	no
BMS-OTS	1288	9507972	2062	no
Our	141805	3126578	70902	yes

Table 5.5 shows a comparison between our proposal and two known multisign signature schemes — CFS and Stern[1] . Our proposal has private and public key sizes less than CFS but a greater signature size. In relation to Stern, our proposal

[1] We use the version presented in (Gaborit and Girault, 2007).

proposal has signature size less than Stern, but greater signature and verification key sizes.

Table 5.5: Comparing our proposal with other coding-based multisign signatures

scheme	$\|sk\|$ bits	$\|pk\|$ bits	sig	hash free
CFS	444434	5898240	180	no
Stern	694	347	120000	no
Our	141805	3126578	70902	yes

For all cases, our proposal has a lower performance to generate the keys, due to we use a exponential algorithm — GBA. However, for all cases our proposal does not use hash function, thus a proof in the standard model can be obtained.

Part III

DFA against an NSA's proposal

Chapter 6

Differential Fault Attack

It is shown that the security of cryptosystems relies on both their mathematical design and their physical implementation. Unsatisfactory features generated naturally, or intentionality, on the physical implementations of a cryptosystem are known as faults. An intentionally one is known as fault attack and is used to extract the key, or some secret of a cryptosystem. The differential fault attacks belong to the class of fault attacks and the first differential attack was proposed by Biham and Shamir against DES in the paper entitled "Differential fault analysis of secret key cryptosystems" (Biham and Shamir, 1997). In this chapter, we review some basic concepts of this type of attack. But first, we introduce fundamentals of fault attacks such as fault model and fault measurement.

6.1 Fundamentals of Fault Attacks

As we have said, a fault attack is a type of attack against the physical implementation of a cryptosystem. This type of attack can be achieved by introducing the physical design in a hostile environment, or by changing the normal state of its components. This procedure is known as fault injection. After that, the possibly wrong outputs of the cryptosystem are observed in order to exploit some leaked information.

To perform a fault attack, there exist basically two stages. The first one is the fault analysis process and the second one is the fault measurement. The fault

analysis process consists in analyzing analytically the success or failure of an attack using a fault model. As we shall see below, to construct a fault model is necessary to choose one or more fault techniques. The fault measurement consists basically in executing the attack using the chosen techniques.

6.1.1 Fault Model

A fault model represents mathematically the properties of a fault injection. Thus, it is possible to study a real-world fault injection from a theoretical point of view. The modeled properties of a fault injection are the location of the fault within the circuit, the number of bits affected by the fault, and the fault effect on the bits (stuck-at, bit-flip, random, set-reset). Table 6.1 shows the possible fault models assumed by an attacker.

Fault model	Fault location	Number of bits
Chosen bit fault	Precise	1
Single bit fault	Loose	1
Byte fault	Loose	8
Random fault	Loose	Any

Table 6.1: Fault models assumed by an adversary

6.1.1.1 Fault Injection Techniques

There are several fault injection tools and techniques that aim to construct fault models. The following are six possible techniques of fault injection.

- **Clock Glitches.** A clock glitch is the irregular behavior during a short time interval of the clock frequency of a circuit. One of the most desired effects with this type of technique is to interrupt the execution of an instruction. Clock glitches are easy to obtain and were the first to be used in order to induce a faulty behavior in devices. Thus, it is a threat that needs to be covered by circuits manufacturers.

- **Voltage Starving.** It is a technique that consists in keeping the voltage level supplied to a processor lower than its nominal one. The induced faults resulting of this technique are known as *data corruptions* and *instructions swap* (Barenghi, Bertoni, Parrinello, and Pelosi, 2009).

- **Heating Up.** It is a technique that consists in exploiting the leakage of processed data by passively measuring the dissipated heat of the device. (Hutter and Schmidt, 2014).

- **Voltage Spikes.** Voltage spikes (Hutter, Schmidt, and Plos, 2008, 2009) are the sudden peak values in a voltage versus time graph due to abnormal conditions like transient or voltage surges. Like clock glitches, voltage spikes are easy to obtain and needs to be covered by circuits manufacturers.

- **Electromagnetic Pulses.** An electromagnetic pulse (EMP), also sometimes called a transient electromagnetic disturbance, is a short burst of electromagnetic energy. Such a pulse's origination may be a natural occurrence or man-made and can occur as a radiated, electric, or magnetic field or a conducted electric current, depending on the source (Shurenkov and Pershenkov, 2016). An example of this technique against AES can be found in (Dehbaoui, Dutertre, Robisson, and Tria, 2012).

- **Laser and Light Pulses.** This technique causes transistors on a chip to switch with photo-electric effects (van Woudenberg, Witteman, and Menarini, 2011).

- **Hardware Trojans.** This technique requires that the adversary has access to the circuit design, or fabrication. Once this is achieved, the adversary injects malicious modifications in the hardware causing the circuit to have altered functional behavior.

Each one of these techniques has characteristics that change a chosen bit, a single bit, a byte or an random number of bits. These characteristics are classified

according to the intensity, location, and duration of the fault injection. Also, depending on the type of access that an attacker has to the circuit under attack the techniques can be categorized into three types: 1) non-invasive, 2) semi-invasive, and 3) invasive. Non-invasive techniques do not require modifications to the device under attack, i.e., only the chip's external interfaces are used. Invasive attacks assume that the attacker has complete access to the internal structures of the circuit. Thus, the attacker can directly eavesdrop upon buses, set or clear signals and even modify the physical properties of the circuit. Generally, this category requires a high expertise depending on the equipment, and the attacks have a high cost. Semi-invasive attacks stand between noninvasive and invasive attacks. They represent a big threat to the hardware security because they are almost as effective as invasive attacks but can be low-cost like non-invasive attacks (Skorobogatov, 2005).

Table 6.2 shows the techniques described above together with its characteristics, the models that they can generate, and its category (invasive, semi-invase, or noninvasive).

6.2 Fault Attack Procedure

The fault attack procedure consists of two phases: the fault measurement and the fault analysis. The fault measurement phase is the process to get faulty data and the fault analyses techniques to process faulty and unaltered information. In next section, we will see that the fault measurement phase consists of four steps: fault injection access, fault injection, fault effect and fault observation. Examples of fault analysis techniques are (differential fault analysis) DFA (Biham and Shamir, 1997), collision fault analysis (CFA) (Hemme, 2004), ineffective fault analysis (IFA) (Blömer and Seifert, 2003) among others. In Section 6.2.2, we review DFA, which was used by us to attack the Simon family Beaulieu et al. (2013).

Table 6.2: Fault injection techniques

Injection	Fault characteristic			Fault Model				Invasiveness	Equipment
	Intensity	Timing	Duration	Chosen bit	Single bit	Byte	Random		
Glitches	Variable	Precise	Transient			•	•	Noninvasive	Low
Starving	Variable	Loose	Transient			•	•	Noninvasive	Low
Spikes	Fixed	Precise	Transient				•	Noninvasive	Low
EM Pulse	Variable	Precise	Transient		•	•	•	Noninvasive	Low/Moderate
Laser Pulse	Fixed	Precise	Transient	•	•	•	•	Semi-invasive	High
Trojans	Fixed	Precise	Transient	•	•	•		Invasive	Moderate

6.2.1 Fault Measurement

The fault measurement phase consists of five steps (Ghalaty, 2016): Fault injection access, fault injection, fault effect, fault observation, and fault exploitation. Following, we give comprehensive definitions of these steps:

1. Fault Injection Access. This is the first and most important step. In this step the attacker obtains physical access to the device under attack. For example, to introduce Hardware Trojans the attacker needs to have access to the circuit design to inject malicious modifications. The same happens with laser and electromagnetic pulse-based fault attacks, i.e., the attacker needs to have access to the chip surface. Also, the attacker needs to know the data inputs and outputs of the device under attack. This access is necessary to build later a model for the attack.

2. Actual Fault Injection. The second step of the fault measurement is to disturb the operation of device under attack by applying a physical stress on it. The applied physical stress pushes the device under attack out of its normal operating conditions and causes faulty operation. Based on the chosen fault injection method, the adversary can control the timing, location, and intensity of the applied physical stress. Each value of these three parameters affects the device under attack differently and causes different faults in it. Therefore, the adversary needs to carefully set these parameters to create an exploitable fault in the device under attack (Ghalaty, 2016).

3. Fault Effect. As its own name says, the fault effect is the effect of the applied physical stress. For example, an applied clock glitch might create 2-bit faults at the fault injection point. The adversary may need to apply several physical stresses with different parameters to create the desired fault effect on the device under attack (Ghalaty, 2016).

4. Fault Observation. To exploit fault injections an attacker needs to observe

their probability of success. In (Yuce, Ghalaty, and Schaumont, 2015), it is shown methods that can compute the probability of success of a fault injection attempt in this phase using observability analysis.

5. Fault Exploitation. This is the final step, and here the attacker exploits the observable effects and breaks the security.

6.2.2 Differential Fault Analysis

This fault analysis was applied for first time against DES (Biham and Shamir, 1997) and after that it was applied against several other cryptosystems. DFA basically is the analysis of a cryptographic cipher by the means of finding a relationship between the difference in the input data and the output data. Ideally, the slightest difference in input data (cleartext), even a single bit, should produce a completely different cypthertext. However, if the cipher is not well-designed, a correlation between the two resulting ciphertext might be observed. This correlation in turn might be exploited to find out the key. This obviously requires a chosen cleartext attack, which means the attacker needs access to the encryption mechanism, and thus use it to encrypt any number of cleartexts he or she chooses. In the next chapter, we present a study of differential fault analysis against Simon (Beaulieu et al., 2013). We also present one of the main contributions of this thesis, that is a DFA against Simon.

Chapter 7

DFA against an NSA's proposal

7.1 The Feistel Cipher

A Feistel cipher is a symmetric structure used in the construction of block ciphers. The name comes from the German-born physicist and cryptographer Horst Feistel who did pioneering research while working for IBM (USA); the cipher is also commonly known as a Feistel network. Several block ciphers use this symmetric structure, such as DES, Simon, SPECK, Blowfish, among others. The Feistel structure has the advantage that the encryption and decryption processes are very similar, even identical in some cases, requiring only a reversal of the key schedule. Therefore, the size of the code or circuitry required to implement such a cipher is nearly halved. In Figure 7.1, we depict the structure of a Feistel cipher, and below we define it formally.

Definition 7.1 (251, Menezes, Vanstone, and Oorschot (1996))**.** *A Feistel cipher is an iterated cipher mapping a $2t$-bit plaintext (L_0, R_0), for t-bit blocks L_0 and R_0, to a ciphertext (R_r, L_r), through an r-round process where $r \geq 1$. For $1 \leq i \leq r$, round i maps $(L_{i-1}, R_{i-1}) \xrightarrow{K_i} (L_i, R_i)$ as follows: $L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, where each subkey K_i is derived from a cipher key K.*

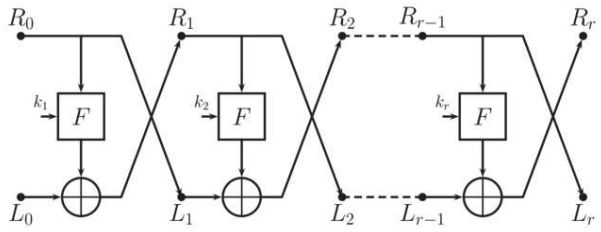

Figure 7.1: Feistel structure

A Feistel cipher was proved secure against chosen message attacks when the number of iterations is three. And it was proved secure both against chosen message attacks and chosen ciphertext attacks, if the number of iterations is four. Both declarations were formalized into theorems and proved by Luby and Rackoff in the paper entitled "How to Construct Pseudorandom Permutations from Pseudorandom Functions" Luby and Rackoff. To prove the first theorem, Luby and Rackoff showed that it is possible to construct an invertible pseudo-random permutation generator with a three round Feistel cipher, and that it is possible to construct a super pseudo-random permutation generator with a four round Feistel cipher.

7.2 The Simon Family Cipher

In 2013, the NSA's members, Beaulieu et al., proposed a family of lightweight block ciphers based upon Feistel structure. This family was designed with the aim of having a better performance for both hardware and software in comparison to other symmetric ciphers currently available in the field. Thus, its design provides optimal performance on resource-constrained devices and provides implementation on a wide range of devices. The Simon family is composed by five members corresponding to 5 block sizes: 32, 48, 64, 96 and 128 bits. Also, each member supports up to 3 key sizes. Hereafter, we will use the term Simon to refer to any of these members.

The design of Simon is a classical Feistel scheme where each round of it operates on two n-bit halves, the left half input and the right half input. Notice, the general round block size is $2n$. In the remainder of this section, we use n to

refer to half of the block size. Also, as we will see in Section 7.2.1, each round uses round keys, which are calculated from a master key. Each member of Simon uses a master key from among three possibilities. The algorithm to calculate the round keys differs depending on the master key sizes.

Each i-th round of Simon applies a non-linear, non-bijective, and non-invertible function

$$\begin{aligned} \mathcal{F} : \mathbb{F}_2^n &\to \mathbb{F}_2^n \\ x &\mapsto ((x <<< 8) \odot (x <<< 1)) \oplus (x <<< 2) \end{aligned} \tag{7.1}$$

to the left half input L^{i-1}. After that, L^i is computed in the following way: 1) $\mathcal{F}(L^{i-1})$ is XORed to the right half input, R^{i-1}, and 2) a round key K^i is XORed to the last output. To complete the operations of the i-th round, the i-th right half input is assigned to the $(i-1)$-th left half input, i.e. $R^i = L^{i-1}$. The number of rounds in Simon relies on the parameters presented in Table 7.1. After T, rounds the ciphertext is obtained, i.e. the output of the last round is the ciphertext C.

As we have said, there are members of the Simon family for different key sizes, block sizes, and number of rounds. The names of each member of Simon with its parameters are presented in Table 7.1.

Table 7.1: Members of the Simon family with their parameters.

Cipher	Block size $2n$	Key words m	Key size mn	Rounds T	Index to z j
Simon 32/64	32	4	64	32	0
Simon 48/72	48	3	72	36	0
Simon 48/96	48	4	96	36	1
Simon 64/96	64	3	96	42	2
Simon 64/128	64	4	128	44	3
Simon 96/92	96	2	92	52	2
Simon 96/144	96	3	144	54	3
Simon 128/128	128	2	128	68	2
Simon 128/192	128	3	192	69	3
Simon 128/256	128	4	256	72	4

In Table 7.1, z_j, $0 \leq j \leq 4$, is a constant sequence.

7.2.1 Key Schedule

The key schedule of Simon is described as a recursive function that uses two, three or four n-bit registers. We denote by m the number of registers used. Depending on m, the recursive function can be:

$$
\begin{aligned}
m = 2\colon\ & K^i = K^{i-2} \oplus \left(K^{i-1} >>> 3\right) \oplus \left(K^{i-1} >>> 4\right) \oplus c \oplus (z_j)_{i-m} \\
m = 3\colon\ & K^i = K^{i-3} \oplus \left(K^{i-1} >>> 3\right) \oplus \left(K^{i-1} >>> 4\right) \oplus c \oplus (z_j)_{i-m} \\
m = 4\colon\ & K^i = \left(K^{i-4} \oplus K^{i-3}\right) \oplus \left(K^{i-1} >>> 3\right) \\
& \quad \oplus \left(\left(K^{i-3} \oplus \left(K^{i-1} >>> 3\right)\right) >>> 1\right) \\
& \quad \oplus c \oplus (z_j)_{i-m}
\end{aligned}
\tag{7.2}
$$

where $c = (2^n - 1) \oplus 3$ is a constant value, $(z_j)_{i-m}$ denotes the $(i-m)$-th bit of z_j, and $i - m$ is taken modulo 62. The values of z_j, $0 \leq j \leq 4$, are depicted in Table 7.2. The value of j depends on the selected Simon member. For example, the value of j according Table 7.1 is Simon 32/64.

Table 7.2: The z_j sequences used in the Simon key schedule.

j	z_j
0	11111010001001010110000111001101111101000100101011000011100110
1	10001110111110010011000010110101000111011111001001100001011010
2	10101111011100000011010010011000101000010001111110010110110011
3	11011011101011000110010111100000010010001010011100110100001111
4	11010001111001101011011000100000010111000011001010010011101111

7.3 DFA against Simon

Now, we study three DFA against Simon, including the one that was proposed here. First, we present a DFA against Simon due to (Tupsamudre et al., 2014). In that work, four models are studied: a chosen bit, a chosen byte, a random bit, and a random byte. After that, a more efficient DFA due to Takahashi and Fukunaga

(2015) is presented. In that work, the authors used a n-bit fault model to perform the attack. At the end, we present our proposal.

7.3.1 Tupsamudre et al. Attack

In 2014, Tupsamudre et al. noted that there is information leakage due to the AND operation used in function $\mathcal{F}$. With this leak information and under certain conditions, they constructed an attack to obtain the last-round key using DFA. Specifically, they proposed four models: (1) a one-bit-flip model; (2) a random one-byte model; (3) a chosen bit model, and (4) a chosen byte model. In this section, we present these models, but firstly, we need to explain some facts about the use of $\mathcal{F}$ and we need to explain what happens when some faults are introduced in the latest rounds.

It can be seen from the application of $\mathcal{F}$ on L^{i-1} that the j-th bit of L^{i-1} affects possibly 3 distinct bits $(j+1)\,\%n$, $(j+2)\,\%n$, and $(j+8)\,\%n$ of $\mathcal{F}\left(L^{i-1}\right)$, i.e. the bits:

$$\begin{aligned}
\mathcal{F}\left(L^{i-1}\right)_{(j+1)\%n} &= \left(L^{i-1}_{j} \odot L^{i-1}_{(j-7)\%n}\right) \oplus L^{i-1}_{(j-1)\%n} \\
\mathcal{F}\left(L^{i-1}\right)_{(j+2)\%n} &= \left(L^{i-1}_{(j+1)\%n} \odot L^{i-1}_{(j-6)\%n}\right) \oplus L^{i-1}_{j\%n} \\
\mathcal{F}\left(L^{i-1}\right)_{(j+8)\%n} &= \left(L^{i-1}_{(j+7)\%n} \odot L^{i-1}_{j\%n}\right) \oplus L^{i-1}_{(j+6)\%n}
\end{aligned} \tag{7.3}$$

where $j \in \{0, \cdots, n-1\}$. Since $L^i = R^{i-1} \oplus \mathcal{F}\left(L^{i-1}\right) \oplus K^{i-1}$, the same bit positions of L^i are also affected by the j-th bit of L^{i-1}.

Another fact is that K^{T-1} and L^{T-2} are related through the next expression

$$K^{T-1} = L^{T-2} \oplus \mathcal{F}\left(R^T\right) \oplus L^T. \tag{7.4}$$

This fact is important because the success of Tupsamudre et al.'s attack is in retrieving K^{T-1} using L^{T-2}.

A third fact is the following: suppose a fault e is induced in the intermediate result L^{T-2}. Let $\left(L^{T^*}, R^{T^*}\right)$ be the resulting faulty ciphertext. Then, the fault e

can be found using the next formula:

$$e = L^T \oplus L^{T^*} \oplus \mathcal{F}\left(R^T\right) \oplus \mathcal{F}\left(R^{T^*}\right). \tag{7.5}$$

Since the output of correct and faulty computation is known, it is possible to deduce the value and location of the fault e injected in L^{T-2} and, hence, it is possible to determine the bits that are flipped in L^{T-2}.

7.3.2 Bit-Flip Fault Attack on Simon at round $T-2$

In the computation of $\mathcal{F}(L^{T-2})$, Tupsamudre et al. observed that if one of the input bits of the AND operation is 0, then flipping the other input bit does not affect the output bit of R^T. Therefore, it is possible to deduce the bit L^{T-2} and consequently to retrieve the bit of K^{T-1} (see equation (7.4)). Let us explore this point in more details.

Suppose a fault flips the j-th bit of the intermediate result L^{T-2} resulting in a faulty ciphertext $\left(L^{T^*}, R^{T^*}\right)$. Then, $R^{T^*} = L^{T-1^*} = R^{T-2} \oplus \mathcal{F}\left(L^{(T-2)^*}\right) \oplus K^{T-2}$. The XORed of the correct and the faulty right halve ciphertexts is written as

$$R^T \oplus R^{T^*} = \mathcal{F}\left(L^{T-2}\right) \oplus \mathcal{F}\left(L^{(T-2)^*}\right).$$

Because the j-th bit of L^{T-2} affects possibly 3 distinct bits of $\mathcal{F}\left(L^{T-2}\right)$, the correct computation of R^T differs from its faulty computation in at most 3 distinct positions:

$$\begin{aligned}
\left(R^T \oplus R^{T^*}\right)_{(j+1)\%n} &= \left(L_j^{T-2} \odot L_{(j-7)\%n}^{T-2}\right) \oplus \left(\left(L_j^{T-2} \oplus 1\right) \odot L_{(j-7)\%n}^{T-2}\right) \\
\left(R^T \oplus R^{T^*}\right)_{(j+8)\%n} &= \left(L_{j+7}^{T-2} \odot L_j^{T-2}\right) \oplus \left(L_{j+7}^{T-2} \odot \left(L_j^{T-2} \oplus 1\right)\right) \\
\left(R^T \oplus R^{T^*}\right)_{(j+2)\%n} &= 1
\end{aligned} \tag{7.6}$$

Both values $L_{(j-7)\%n}^{T-2}$ and $L_{(j+7)\%n}^{T-2}$ can be deduced easily using their truth tables (see Table 7.3 and Table 7.4). For example, looking at Table 7.3, we can deduce

$L^{T-2}_{(j-7)\%n}$. In fact, if $\left(R^T \oplus R^{T^*}\right)_{(j+1)\%n}$ is 0, then irrespective of the bit value L^{T-2}_j, $L^{T-2}_{(j-7)\%n}$ is 0, otherwise it is 1. In a similar way, by looking the Table 7.4, we can deduce $L^{T-2}_{(j+7)\%n}$. In fact, if $\left(R^T \oplus R^{T^*}\right)_{(j+8)\%n}$ is 0, then irrespective of the bit value L^{T-2}_j, $L^{T-2}_{(j+7)\%n}$ is also 0, otherwise it is 1.

Table 7.3: Truth table for $\left(R^T \oplus R^{T^*}\right)_{(j+1)\%n}$.

L^{T-2}_j	$L^{T-2}_{(j-7)\%n}$	$\left(R^T \oplus R^{T^*}\right)_{(j+1)\%n}$
0	0	0
1	0	0
0	1	1
1	1	1

Table 7.4: Truth table for $\left(R^T \oplus R^{T^*}\right)_{(j+8)\%n}$.

L^{T-2}_j	$L^{T-2}_{(j+7)\%n}$	$\left(R^T \oplus R^{T^*}\right)_{(j+8)\%n}$
0	0	0
1	0	0
0	1	1
1	1	1

Since the values of $L^{T-2}_{(j-7)\%n}$ and $L^{T-2}_{(j+7)\%n}$ are known, it is possible to retrieve the corresponding bits of K^{T-1} using Equation (7.4), i.e.

$$\begin{aligned} K^{T-2}_{(j-7)\%n} &= L^{T-2}_{(j-7)\%n} \oplus \mathcal{F}\left(R^T\right)_{(j-7)\%n} \oplus L^T_{(j-7)\%n} \\ K^{T-2}_{(j+7)\%n} &= L^{T-2}_{(j+7)\%n} \oplus \mathcal{F}\left(R^T\right)_{(j+7)\%n} \oplus L^T_{(j+7)\%n}. \end{aligned} \tag{7.7}$$

Tupsamudre et al. showed that to retrieve the n-bit round key is necessary $n/2$ faulty ciphertexts if there is a control over the location where the faults were added. If there is no control, they calculated the average of faulty encryptions in an experimental way, see Table 7.5.

Table 7.5: Bit-Flip Fault Attack on Simon assuming no control over the fault position.

n bits	Avg. No. of Faulty Encryptions
16	25
24	43
32	62
48	104
64	150

7.3.3 Random Byte Fault Attack on Simon at round $T-2$

A most practical attack might affect a byte of L^{T-2}. In this case, Tupsamudre et al. showed that it is possible to use the same working principle explained in the last section to retrieve K^{T-1}. They experimentally found the average of random fault injections that are required to retrieve K^{T-1}. This average is shown in Table 7.6. They also showed that when there is a control over the location for fault injections, the number of fault injections to retrieve K^{T-1} is $n/8$.

Generally with a single bit of an byte fault, we can retrieve two values of L^{T-2}. However, Tupsamudre et al. mentioned two exceptions: (1) the least and most significant bits of the induced byte fault are one, and (2) a byte fault flips two adjacent bits. In the first exception the adversary can retrieve only one bit of the last-round, and in the second exception the adversary can retrieve four bits of the last round but modifying slightly the formulas to deduce these bits.

Table 7.6: Average of faulty encryptions for the Random Byte Fault Attack on Simon at round $T-2$.

n bits	Avg. No. of Faulty Encryptions
16	6
24	9
32	13
48	21
64	30

7.3.4 n-bit Fault Attack on Simon

In a way similar to the last two attacks, Takahashi and Fukunaga analyzed the input and output differences in the AND operation when a random fault injection is added on n bits in an intermediate position of Simon. They have precisely calculated the average number of fault injections to obtain a round key by examining the relationships between the bits obtained through multiple fault injections. Their analysis reduces significantly the average number of fault injections to retrieve L^{T-2}. For instance, in Simon 128/128 the n-bit model uses 3.91 faults instead of 8 used by the one-byte model. Also, they showed specifically what fault positions are necessary to obtain m round keys. With these m round keys and using the key-schedule algorithm, they calculated the entire secret key of Simon.

7.4 One-Bit-Flip Fault Attack on Simon at round $T-3$

In this section, we present the main idea of our attack, that is, a one-bit-flip fault attack on Simon at round $T-3$. This contribution was presented in the Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2015 (Grados et al., 2015)

We observed that if an adversary is able to flip a single round, then we can retrieve information of two rounds. This is new compared with the one-bit-flip, one-byte and n-bit models presented in Section 7.3.1, i.e., those which retrieve information of only one round. Specifically, we will show that is possible to obtain all bits of L^{T-2} and L^{T-3} injecting faults in L^{T-3}. Also, we will show how to retrieve m round keys using our modification.

Before explaining how to retrieve information of L^{T-2} and L^{T-3}, using the single round $T-3$, we need to know what are the probabilities that an error occurred in L^{T-3} affects 1, 2 or 3 bits of $F\left(L^{T-3}\right)$. To achieve this, remember that we can know the number of possible affected bits (1, 2 or 3 bits) due to equation (7.3).

Without loss of generality, let L be a n-bit string. Let F be a non-linear

function as defined in (7.1). If $F(L^*)$ is the application of F on L when its j^{th} bit is flipped, then the probability of flipping 1, 2, or 3 bits of $F(L^*)$ is 1/4, 1/2 and 1/4 respectively.

To prove this statement, we have to look at the 3 distinct bits of $F(L^*)$ affected by the j^{th} flipped bit of L^*:

$$F(L^*)_{j+1} = \big((L_j \oplus 1) \odot L_{(j-7)\%n}\big) \oplus L_{j-1}$$
$$F(L^*)_{j+2} = (L_{j+6} \odot L_{j+1}) \oplus L_j \oplus 1$$
$$F(L^*)_{j+8} = (L_{j+7} \odot (L_j \oplus 1)) \oplus L_{j+6}$$

First, we calculate the individual probabilities of $F(L^*)_{j+1}$, $F(L^*)_{j+2}$ and $F(L^*)_{j+8}$ be affected by the j^{th} flipped bit in L. Because $F(L^*)_{j+2} \oplus F(L)_{j+2}$ is always 1, the probability of the $(j+2)^{th}$ bit of $F(L^*)$ be flipped is 1. The $(j+1)^{th}$ bit of $F(L^*)$ flips only if $L_j = 1$ and $L_{(j-7)\%n} = 1$, or if $L_j = 0$ and $L_{(j-7)\%n} = 1$. In other words, when half of the possibilities happens. In similar way, the $(j+8)^{th}$ bit of $F(L^*)$ flips only if $L_j = 1$ and $L_{(j+7)\%n} = 1$, or if $L_j = 0$ and $L_{(j+7)\%n} = 1$. Again, when half of the possibilities happens. Then the probability that no error occurs in the $(j+1)^{th}$ and $(j+8)^{th}$ bits of $F(L^*)$ is equal to 1/2 for both cases. Thus, the probability P_1 of only one flipped bit occur in $F(L^*)$ is $\mathrm{P}_1 = 1 \cdot 1/2 \cdot 1/2 = 1/4$. The one-bit-flip and one-byte models cannot be performed without knowing the positions of the flipped bits in the left half input L^{T-2}. To retrieve these positions, equation (7.5) is necessary. Similarly, for our modification we cannot perform our attack if we do not know the position of the flipped bit in the left half input L^{T-3}, and the positions of flipped bits in L^{T-2} affected by $F\left(L^{(T-3)^*}\right)$. In the following discussion, we show how to retrieve the location of the j^{th} flipped bit in L^{T-3}, and then how to retrieve the locations of the flipped bits in L^{T-2}.

7.4.1 Deducing j

Since the j^{th} flipped bit in $L^{(T-3)^*}$ affects the 3 possible positions $a = (j + 1)\%n$, $b = (j + 2)\%n$ and $c = (j + 8)\%n$, the idea for deducing j arises from the fact that these locations are fixed. In fact, if 3 flipped bits appear in $F\left(L^{(T-3)^*}\right)$, then to deduce j it is only necessary to calculate a, and then j must be $a - 1$. But because a, b, and c are unknown we need to find them. In the next section, we explain how to retrieve those affected bits. For now, suppose that these bits are represented with ones in the a^{th}, b^{th} and c^{th} positions of a n-bit string e'. For example, suppose the bit at the location $j = 2$ was flipped resulting in $L^{(T-3)^*}$. Also, suppose this flipped bit affects 3 bits of $F\left(L^{(T-3)^*}\right)$. Then,

$$e' = 0\underset{j}{0}01100000100\cdots 0.$$

The case with 3-bit flips in e' does not always happen. In fact, we have shown that the probability of flipping 1, 2 or 3 bits is 1/4, 1/2, and 1/4 respectively. To capture all those cases, we have developed Algorithm 2. Here, LSB$(\cdot)$ and MSB$(\cdot)$ are functions to obtain the least and the most significant bits respectively. The function wt$(\cdot)$ calculates the Hamming weight, and abs$(\cdot)$ returns the absolute value.

How to deduce j when e' has Hamming weight 2 or 3 is shown in steps 4 and 11, respectively. The idea to retrieve j for the case wt$(e') = 3$ is to find the two adjacent flipped bits, i.e., $(j + 1)\,\%n$ and $(j + 2)\,\%n$, in e' and then $j = \text{LSB}\,(e')-1$. The idea in the case wt$(e') = 2$ is to know whether e' has two adjacent bits, which is stablished by functions LSB$(\cdot)$ and MSB$(\cdot)$ in the steps 29 and 13, respectively. Note that the two-adjacent bit case is the same in step 4 except when the difference between the least and the most significant bits is equal to $n - 1$. This case is considered as two separate bits and the expression to calculate j is showed in step 27. The case of two separate bits has four sub-cases. The first two happen when the $(j + 1)\,\%n < (j + 8)\,\%n$, or $(j + 2)\,\%n < (j + 8)\,\%n$. In those subcases

the expressions to retrieve j are shown in steps 15 and 18, respectively. The last two subcases occur when $(j+1)\,\%n > (j+8)\,\%n$, or $(j+2)\,\%n > (j+8)\,\%n$. In those subcases the expressions to retrieve j are shown in steps 21 and 24, respectively.

It is not possible to deduce j when e' has Hamming weight 1. In the next section, we will show that this case is not necessary in our attack.

7.4.2 Retrieving L^{T-2} and K^{T-1}

Suppose that a fault flips the j^{th} bit of the intermediate result L^{T-3} resulting in a faulty ciphertext $\left(L^{T^*}, R^{T^*}\right)$. As we can see in Section 7.4.1, one flipped bit at the j^{th} location of L^{T-3} possibly affects 3 distinct bits of $F\left(L^{(T-3)^*}\right)$. Those bits are $F\left(L^{T-3}\right)_{(j+1)}$, $F\left(L^{T-3}\right)_{(j+2)}$, and $F\left(L^{T-3}\right)_{(j+8)}$, and again they may affect 3 distinct bits of L^{T-2} because $L^{T-2} = F\left(L^{T-3}\right) \oplus R^{T-3} \oplus K^{T-3}$.

It is possible that the j^{th} flipped bit in L^{T-3} flips only one bit of L^{T-2}, which flips only one bit of $F\left(L^{(T-2)^*}\right)$. Since $L^{T-3} = R^{T-2}$, one might think that the i^{th} affected bit in $F\left(L^{(T-2)^*}\right)$ might coincide with the j^{th} flipped bit in R^{T-2}, and then no error would occur in the output of the last round of Simon. However, from the discussion at the beginning of Section 7.4, this case does not happen because we know that if only one bit is affected in $F\left(L^{(T-2)^*}\right)$, then the bit location is $i = (j+2)\,\%n$ and for all cases of Simon $j \neq i \neq (j+2)\,\%n$. If one flipped bit occurs in L^{T-3}, then the left half input L^T will always have at least one flipped bit.

By (7.5), we know how to find the fault e that flipped the bits in L^{T-2}. Using Algorithm 2 with input e, it is possible to deduce the j^{th} bit flipped in L^{T-3} and consequently in R^{T-2}. Since the bits L^{T-2}_{j+1}, L^{T-2}_{j+2} and L^{T-2}_{j+8} are inside a byte then it is possible to apply the random one-byte model presented in Section 7.3.3 to retrieve information about L^{T-2} and K^{T-1}. But the formulas of that section have to be re-written because the j^{th} fault location of R^{T-2} may coincide with some fault locations of each 3 possible distinct bits affected by L^{T-2}_{j+1}, L^{T-2}_{j+2} or L^{T-2}_{j+8}.

Algorithm 2 Deducing j

Input: bit string e' of size n
Output: deducing j

1: $lsb \leftarrow \mathrm{LSB}(e')$
2: $msb \leftarrow \mathrm{MSB}(e')$
3: $j \leftarrow -1$
4: **if** $\mathrm{wt}(e') = 3$ **then**
5: **for** $i = 0$ **to** $n-1$ **do**
6: **if** $e'[i\%n] = 1$ and $e'[(i+1)\%n] = 1$ **then**
7: $j \leftarrow i - 1$
8: **end if**
9: **end for**
10: **end if**
11: **if** $\mathrm{wt}(e') = 2$ **then**
12: $d \leftarrow \mathrm{abs}(lsb - msb)$
13: **if** $d > 1$ **then**
14: **if** $d = 7$ **then**
15: $j \leftarrow (lsb - 1)\%n$
16: **end if**
17: **if** $d = 6$ **then**
18: $j \leftarrow (lsb - 2)\%n$
19: **end if**
20: **if** $d = n - 7 + 1$ **then**
21: $j \leftarrow (msb - 2)\%n$
22: **end if**
23: **if** $d = n - 7$ **then**
24: $j \leftarrow (msb - 1)\%n$
25: **end if**
26: **if** $d = n - 1$ **then**
27: $j \leftarrow n - 2$
28: **end if**
29: **else**
30: **for** $i = 0$ **to** $n-1$ **do**
31: **if** $e'[i\%n] = 1$ and $e'[(i+1)\%n] = 1$ **then**
32: $j \leftarrow i - 1$
33: **end if**
34: **end for**
35: **end if**
36: **end if**
37: **return** $j\%n$

These new formulas are in Appendix 8.

Because the output of F on L^{T-2} is XORed with R^{T-2}, in these new formulas we need to add the difference of $\tilde{R}_x^{T-2}$ between R_x^{T-2} and $R_x^{(T-2)^*}$ where x corresponds to the fault location of each bit affected by the $(j+1)^{th}$, $(j+2)^{th}$ and $(j+8)^{th}$ of L^{T-2}. If only one bit is flipped in the output of $F\left(L^{(T-2)^*}\right)$, it is at the location $((j+2)\,\%n)^{th}$. This flipped bit affects 3 bits of $\left(R^T \oplus R^T\right)$, namely $\left(R^T \oplus R^T\right)_{(j+3)\%n}$, $\left(R^T \oplus R^T\right)_{(j+4)\%n}$ and $\left(R^T \oplus R^T\right)_{(j+10)\%n}$. In this case, note that the terms $\tilde{R}_{j+3}^{T-2}$,$\tilde{R}_{j+4}^{T-2}$ and $\tilde{R}_{j+10}^{T-2}$ for these formulas, are always zero. This explains why is not necessary to deduce j when L^{T-2} has Hamming weight 1.

Table 7.7 shows the bits of L^{T-2} that are retrievable using those formulas when L_{j+1}^{T-2}, L_{j+2}^{T-2} or L_{j+8}^{T-2} were affected. The first column shows the affected bits by L_{j+1}^{T-2}, L_{j+2}^{T-2} or L_{j+8}^{T-2}. The second column shows the conditions, because multiple bits are flipped. The third column shows the deduced values using the truth tables of the formulas in Appendix 8.

For example, suppose the j^{th} bit in L^{T-3} was flipped and this bit affects the $(j+2)^{th}$ bit in L^{T-2}. Also, suppose that the $(j+2)^{th}$ bit flipped in L^{T-2} affects the $(j+3)^{th}$ and $(j+4)$-th bits in L^{T-1}. Using the $((j+3)\,\%n)^{th}$ bit of $\left(R^T \oplus R^{T^*}\right)$, and supposing that the $((j+1)\,\%n)^{th}$ bit of L^{T-2} was not flipped, we can get the following formula:

$$\begin{aligned}\left(R^T \oplus R^{T^*}\right)_{(j+3)\%n} = &\left(L_{(j+2)\%n}^{T-2} \odot L_{(j-5)\%n}^{T-2}\right) \\ &\oplus \left(\left(L_{(j+2)\%n}^{T-2} \oplus 1\right) \odot L_{(j-5)\%n}^{T-2}\right) \\ &\oplus \tilde{R}_{(j+3)\%n}^{T-2}.\end{aligned} \tag{7.8}$$

Here, $\tilde{R}_{(j+3)\%n}^{T-2} = \left(R^{T-2} \oplus R^{(T-2)^*}\right)_{(j+3)\%n}$. After constructing the truth table for that formula, we can deduce the $((j-5)\,\%n)^{th}$ bit of L^{T-2}. This example is highlighted in gray in Table 7.7.

Table 7.7: Deducing bits of L^{T-2}.

affected bits by L_{j+1}^{T-2}	conditions	deduced value
$\left(R^T \oplus R^{T^*}\right)_{(j+2)\%n}$		$L_{(j-6)\%n}^{T-2}$
$\left(R^T \oplus R^{T^*}\right)_{(j+3)\%n}$	$\tilde{L}_{(j+2)\%n}^{T-2} = 1$	$L_{(j-5)\%n}^{T-2}$
	$\tilde{L}_{(j+2)\%n}^{T-2} = 0$	
$\left(R^T \oplus R^{T^*}\right)_{(j+9)\%n}$	$\tilde{L}_{(j+8)\%n}^{T-2} = 1$	
	$\tilde{L}_{(j+8)\%n}^{T-2} = 0$	$L_{(j+8)\%n}^{T-2}$
affected bits by L_{j+2}^{T-2}	conditions	deduced value
$\left(R^T \oplus R^{T^*}\right)_{(j+3)\%n}$	$\tilde{L}_{(j+1)\%n}^{T-2} = 1$	$L_{(j-5)\%n}^{T-2}$
	$\tilde{L}_{(j+1)\%n}^{T-2} = 0$	$L_{(j-5)\%n}^{T-2}$
$\left(R^T \oplus R^{T^*}\right)_{(j+4)\%n}$		
$\left(R^T \oplus R^{T^*}\right)_{(j+10)\%n}$	$\tilde{L}_{(j+8)\%n}^{T-2} = 1$	$L_{(j+9)\%n}^{T-2}$
	$\tilde{L}_{(j+8)\%n}^{T-2} = 0$	$L_{(j+9)\%n}^{T-2}$
affected bits by L_{j+8}^{T-2}	conditions	deduced value
$\left(R^T \oplus R^{T^*}\right)_{(j+9)\%n}$	$\tilde{L}_{(j+1)\%n}^{T-2} = 1$	
	$\tilde{L}_{(j+1)\%n}^{T-2} = 0$	$L_{(j+1)\%n}^{T-2}$
$\left(R^T \oplus R^{T^*}\right)_{(j+10)\%n}$	$\tilde{L}_{(j+2)\%n}^{T-2} = 1$	$L_{(j+9)\%n}^{T-2}$
	$\tilde{L}_{(j+2)\%n}^{T-2} = 0$	
$\left(R^T \oplus R^{T^*}\right)_{(j+16)\%n}$		$L_{(j+15)\%n}^{T-2}$

7.4.3 Retrieving L^{T-3} and K^{T-2}

To retrieve L^{T-3} and K^{T-2} we use the one-bit-flip model attack described in Section 7.3.2 only on the $T-1$ rounds as indicated by the dashed rectangle in Figure 7.2. This is possible now because we know the output $\left(L^{T-1}, R^{T-1}\right)$ of the round $T-2$. We will be calling this output by "correct intermediate text", and since K^{T-1} is known, the elements of this output are $L^{T-1} = R^T$ and $R^{T-1} = F\left(Y^T\right) \oplus L^T \oplus K^{T-1}$. When an intermediate error occurs in L^{T-3}, we will be calling the output of the round $T-2$ by "faulty intermediate text". We could

construct the "faulty intermediate text" by adding new errors, but instead we reuse the errors generated for each faulty error injected in L^{T-3} explained in the last section, and then we use the idea of the one-bit model at round $T-2$ to retrieve L^{T-3}. That is, we repeat the random fault injections in L^{T-3} but using the "faulty intermediate text", which can be calculated in the following way: $Y^{T^*} = L^{(T-1)^*}$ and $R^{(T-1)^*} = R^{T-1} \oplus e$.

Following the example of the Section 7.4.2, we can retrieve the $((j-7)\,\%n)^{th}$ and $((j+7)\,\%n)^{th}$ bits of L^{T-3}. The flipped bits of this example are highlighted in red, and the retrieved bits are highlighted in green in Figure 7.2.

7.4.4 Retrieving the entire key of Simon 96/96, Simon 128/128 and other members

The idea of our modification is to use less fault positions compared with Tupsamudre et al. (2014); Takahashi and Fukunaga (2015). In this context, to retrieve the entire secret key of Simon 96/96 and Simon 128/128, we need to obtain 2 round keys based on the key word size $m = 2$. Then, we use the previous analysis of sections 7.4.2 and 7.4.3. From the key expansion of Simon, we can calculate the round keys one after another from the deduced $m = 2$ round keys. That is, for $k = i - 2$ and $m = 2$ of (7.2) we can obtain

$$K^k = K^{k+2} \oplus \left(K^{k+1} >>> 3\right) \oplus \left(K^{k+1} >>> 4\right) \oplus c \oplus \left(z_j\right)_{j-m+2}. \tag{7.9}$$

Since K^{T-1} and K^{T-2} can be found using the method presented in Section 7.4.3, we can feed these values into (7.9) and after $T-2$ rounds we can find the entire key of the Simon 96/96 or the Simon 128/128.

For the $m = 3$ and $m = 4$, the above explanation can be applied if m adjacent round-keys are known and if we chose more rounds for fault injections. That is, the adversary needs to select one more fault position. For the case $m = 3$, the adversary has two options. The first one is injecting faults to the left half input L^{T-3} and to retrieve the round-keys K^{T-1} and K^{T-2} and the output $\left(L^{T-2}, R^{T-2}\right)$ of round

$T-3$. This output can be retrieved with the method presented in Section 7.4.3. To retrieve K^{T-3}, the adversary must inject faults to the left half input L^{T-5} and to retrieve K^{T-3} and K^{T-4}, the adversary must use the same method and the output $\left(L^{T-2}, R^{T-2}\right)$, which is known. The second option is: after retrieving $\left(L^{T-2}, R^{T-2}\right)$ the adversary injects faults at round $T-4$ using one of the methods presented in Section 7.3.3. For the case $m = 4$, the adversary must follow the first option of the case $m = 3$.

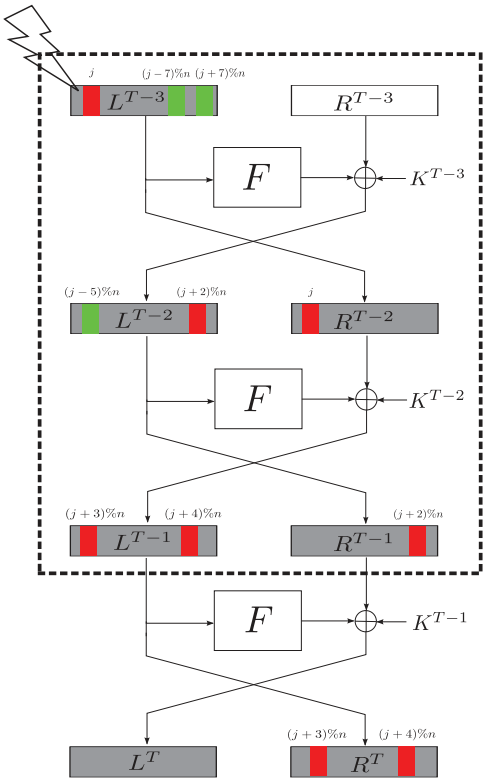

Figure 7.2: Fault propagation when L^{T-3} is randomly corrupted in the j^{th} bit. Gray denotes the faulty intermediate states and the lines the rounds necessary for retrieve K^{T-2}.

7.4.5 Average Number of Fault Injections

The expected number of K^{T-1} bits recovered by a single bit-flip in L^{T-3} is

$$2\text{P}(1 \text{ flip in } L^{T-2}) + 4\text{P}(2 \text{ flips in } L^{T-2}) + 4\text{P}(3 \text{ flips in } L^{T-2}) = 2\frac{1}{4} + 4\frac{1}{2} + 4\frac{1}{4} = 3.5 \text{ bits.}$$

In addition, using the same pair of correct and faulty ciphertexts, two bits of K^{T-2} can be recovered. Thus, one-bit-flip in $L^{(T-3)}$ recovers 3.5 bits of K^{T-1} and 2 bits of K^{T-2} on average. The current bit-flip attack is therefore more efficient than the bit-flip attack of Tupsamudre et al. (2014), which recovers only 2 bits of K^{T-1}.

The random process to recover K^{T-1} and K^{T-2} can be modeled using the cover time of a random walk. According to Lovász (1993), the average number of faulty injections needed to recover K^{T-1} and K^{T-2} using our one-bit-flip model is

$$R = \frac{(n-1)}{4}\left(\Psi\left(\frac{n+1}{2}\right) + 2\left(\gamma + \ln(2)\right)\right) + \frac{(2\gamma - 1) + n\Psi\left(\frac{1}{2}\right)}{4} \tag{7.10}$$

where Ψ is a digamma function and γ is the Euler number.

Table 7.8 shows the experimental average number of fault injections to obtain the two round keys K^{T-1} and K^{T-2}. Specifically, the second column shows the average of fault injections to obtain K^{T-1} and K^{T-2}. These values confirm the theoretical average number of fault injections R for each member of Simon. For instance, when $n = 16$, $R = 25.43$. Also, these values confirm the previous study of the average number of fault injection presented in Table 7.5, but in our modification, we retrieve two round keys and not only one. The third column of the Table 7.8 shows the average number of reused fault injections to retrieve K^{T-1}. For example, for Simon 32/64 this is approximately 15. Note that the average number of reused fault injections is always limited by the average number of fault injection used in the one-bit-flip model. This also confirms that it is possible to retrieve L^{T-2} and L^{T-3} using our modification.

Table 7.8: Average Number of Fault Encryptions to obtain L^{T-3} and L^{T-2}.

n	Avg. No. Fault Injections L^{T-3} and L^{T-2}	Avg. No. Fault Injections reused L^{T-2}	Time (s)
16	24.81	15.26	10.17
24	42.74	29.70	15.49
32	61.75	44.19	36.84
48	103.42	77.02	57.85
64	147.57	110.81	151.82

7.5 Simulations

To verify the proposed attack and to evaluate the average number of fault injections, we implement the attack in Python and execute the code on an Intel Core i5 2.6 GHz processor using Ubuntu 12.04 (64 bits) OS. In the simulation, we assume that L^{T-3} is randomly corrupted with one-bit fault. The plaintexts and the secret keys are randomly chosen.

As mentioned in Section 7.4.5, the first column of Table 7.8 shows the average number of fault injections to obtain all bits in L^{T-3} and L^{T-2} when $n = 16, 24, 32, 48, 64$. The third column of Table 7.8 shows the average number of fault injections that will be reused to obtain only L^{T-2}. The number of samples is 100,000 in Table 7.8. From the results, the average number of fault injections to obtain all 128 bits of L^{T-2} and L^{T-3} is 150.

7.6 Comparison with Related Work

Table 7.9 gives a summary of our method applied on Simon family. We also show a comparison between this study and previous work Tupsamudre et al. (2014); Takahashi and Fukunaga (2015). The number of samples is 100,000 in Table 7.9. The 5^{th}, 6^{th},7^{th} and 9^{th} columns of Table 7.9 shows the average number of fault injections required to retrieve the entire secret key for all members of Simon. The

5^{th} column shows the average number of fault injections using the one-byte model. The 6^{th} column shows the average number of fault injections using the one-bit-flip model presented in Tupsamudre et al. (2014). The 7^{th} column shows the average number of fault injections using the n-bit model. And, the 9^{th} column shows the average number of fault injections using our one-bit model.

Note that when our modification is compared with the one-bit-flip model in the cases $m = 2$ and $m = 4$, our modification needs in average half of fault injections and locations to retrieve the entire key. When our modification is compared with both the one-byte model and n-bit model, our modification needs half of fault positions. Obviously, for the last two comparisons the average of number of fault injections is greater.

Different from Tupsamudre et al. (2014), and similar to Takahashi and Fukunaga (2015), our attack method achieves the entire key and not only a round key. Besides, our modification works with all original rounds of each member of the Simon in contrast with the linear cryptanalysis proposed in Alizadeh, Alkhzaimi, Aref, Bagheri, and Lauridsen (2014); Alizadeh, Bagheri, Gauravaram, Kumar, and Sanadhya (2013); Wang, Liu, Varici, Sasaki, Rijmen, and Todo (2014), which works with the reduced round version of Simon.

Table 7.9: Comparison of results of DFA on Simon family.

Block Size	Key Size	Key Words(m)	Fault Location	Avg.Tupsamudre et al. (2014) One-byte	Avg.Tupsamudre et al. (2014) One-bit-flip	Avg.Takahashi and Fukunaga (2015) n-bit	Fault Location	Avg. One-bit-flip
32	64	4	$L^{27}, L^{28}, L^{29}, L^{30}$	24	101.72	12.20	L^{27}, L^{29}	50.85
48	72	3	L^{32}, L^{33}, L^{34}	27	130.78	9.91	L^{32}, L^{33}	87.19
48	96	4	$L^{31}, L^{32}, L^{33}, L^{34}$	36	174.37	13.22	L^{31}, L^{33}	87.19
64	96	3	L^{38}, L^{39}, L^{40}	39	189.44	10.45	L^{38}, L^{39}	126.29
64	128	4	$L^{39}, L^{40}, L^{41}, L^{42}$	52	252.58	13.93	L^{39}, L^{41}	126.29
96	96	2	L^{49}, L^{50}	42	210.24	7.46	L^{49}	105.12
96	144	3	L^{50}, L^{51}, L^{52}	63	315.36	11.19	L^{50}, L^{51}	210.24
128	128	2	L^{65}, L^{66}	60	299.68	7.82	L^{65}	149.84
128	192	3	L^{65}, L^{66}, L^{67}	90	449.52	11.73	L^{65}, L^{66}	299.68
128	256	4	$L^{67}, L^{68}, L^{69}, L^{70}$	120	599.36	15.64	L^{67}, L^{69}	299.68

Chapter 8

Conclusions

This thesis is divided into three parts. In the first part, we have reviewed the fundamentals of coding theory and three code-based cryptosystems: McEliece, Niederreiter and the CFS signature. In the second part, we have studied the following one-time signatures: Lamport, Winternitz, W-OTS$^+$, Merkle, KKS, and BMS-OTS. In the third part, we have analyzed fundamentals of differential fault analysis and the Simon family.

Our contribution in the second part is a new code-based one-time signature inspired in the Lamport scheme. Our proposal is better in certain aspects than the code-based one-time signatures reviewed in this thesis. In fact, we obtain a significant gain on the verification key size when we compare our proposal with KKS and BMS-OTS with random codes. However, if we implement KKS and BMS-OTS with double-circulant codes, our proposal has larger key and signature sizes. When we compare our proposal with CFS, ours has smaller key size but larger signature size. When we compare with Stern, our proposal has smaller signature size but larger key size. Then, our proposal is suitable depending on the implementation demand, which can be to optimize the size of the verification key or the size of the signature.

The one-time signatures reviewed in this thesis use hash functions; our proposal does not. As a future work, we intend check whether it is possible to create a proof in the standard model for our proposal. Besides, we intend to analyze the

possibility of reducing the signature key by using seeds.

Our contribution in the third part is an efficient one-bit model for differential fault analysis on the Simon family. The idea behind this analysis is to insert the fault in an iteration prior to the one used in Tupsamudre et al. (2014). The strategy is to take advantage of both the one-bit and the one-byte models presented in Tupsamudre et al. (2014). We have shown how to extract the secret key using a random one-bit fault model for all parameters of the Simon family. By taking Simon 128/128 as an example, we extract the entire secret key using 149.84 fault injections on average. The average number of fault injections needed by our proposal in order to extract the secret key of Simon is greater than in the random one-byte and n-bit models discussed in this thesis. However, the number of fault positions needed by our proposal is smaller than the random one-byte and n-bit models. Our proposal is useful when the adversary pays a high price to inject the faults in more than one round.

As a future work, we intend to investigate whether it is possible to extend our ideas to attack other symmetric ciphers using differential fault analysis.

Bibliography

Javad Alizadeh, Hoda A. Alkhzaimi, Mohammad Reza Aref, Nasour Bagheri, and Martin M. Lauridsen. Improved Linear Cryptanalysis of Round Reduced SIMON, 2014.

Javad Alizadeh, Nasour Bagheri, Praveen Gauravaram, Abhishek Kumar, and Somitra Kumar Sanadhya. Linear Cryptanalysis of Round Reduced Variants of SIMON, 2013.

M. Barbier and P. S. L. M. Barreto. Key reduction of McEliece's cryptosystem using list decoding. In: **2011 IEEE International Symposium on Information Theory Proceedings**, pages 2681–2685, July 2011.

A. Barenghi, G. Bertoni, E. Parrinello, and G. Pelosi. Low Voltage Fault Attacks on the RSA Cryptosystem. In: **2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)**, pages 23–31, Sept 2009.

Paulo S. L. M. Barreto, Rafael Misoczki, and Marcos A. Simplicio Jr. One-time Signature Scheme from Syndrome Decoding over Generic Error-correcting Codes. **J. Syst. Softw.**, 84(2):198–204, Fevereiro 2011.

Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.

Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding Random Binary Linear Codes in $2^{n/20}$: How $1 + 1 = 0$ Improves Information Set Decoding. In: David Pointcheval and Thomas Johansson (editors), **Advances**

in Cryptology – EUROCRYPT 2012, pages 520–536, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit, and Ayoub Otmani. **Reducing Key Length of the McEliece Cryptosystem**. In: Bart Preneel (editor), **Progress in Cryptology – AFRICACRYPT 2009: Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings**. pages 77–97, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (Corresp.). **IEEE Transactions on Information Theory**, 24(3):384–386, May 1978.

Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. **Post Quantum Cryptography**. Springer Publishing Company, Incorporated, 1st edição, 2008.

Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. **SPHINCS: Practical Stateless Hash-Based Signatures**. In: Elisabeth Oswald and Marc Fischlin (editors), **Advances in Cryptology – EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I**. pages 368–397, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

Eli Biham and Adi Shamir. Differential fault analysisystemsof secret key cryptosystems. In: Burton S. Kaliski (editor), **Advances in Cryptology — CRYPTO '97**, pages 513–525, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

Johannes Blömer and Jean-Pierre Seifert. Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In: Rebecca N. Wright (editor), **Financial**

Cryptography, pages 162–181, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. **On the Security of the Winternitz One-Time Signature Scheme**. In: Abderrahmane Nitaj and David Pointcheval (editors), **Progress in Cryptology – AFRICACRYPT 2011: 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings**. pages 363–378, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. **Merkle Signatures with Virtually Unlimited Signature Capacity**. In: Jonathan Katz and Moti Yung (editors), **Applied Cryptography and Network Security: 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007. Proceedings**. pages 31–45, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

Johannes Buchmann, Luis Carlos Coronado García, Erik Dahmen, Martin Döring, and Elena Klintsevich. CMSS: An Improved Merkle Signature Scheme. In: **Proceedings of the 7th International Conference on Cryptology in India**, INDOCRYPT'06, pages 349–363, Berlin, Heidelberg, 2006a. Springer-Verlag.

Johannes Buchmann, Luis Carlos Coronado García, Erik Dahmen, Martin Döring, and Elena Klintsevich. CMSS - An Improved Merkle Signature Scheme. In: Rana Barua and Tanja Lange (editors), **Progress in Cryptology - INDOCRYPT 2006**, volume 4329 of **Lecture Notes in Computer Science**, pages 349–363. Springer Berlin Heidelberg, 2006b.

Pierre-Louis Cayrel, Ayoub Otmani, and Damien Vergnaud. **On Kabatianskii-Krouk-Smeets Signatures**. In: Claude Carlet and Berk Sunar (editors), **Arithmetic of Finite Fields: First International Workshop, WAIFI**

2007, Madrid, Spain, June 21-22, 2007. Proceedings. pages 237–251, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

N. Courtois, M. Finiasz, and N Sendrier. How to achieve a McEliece-based Digital Signature Scheme. **Lecture Notes in Computer Science**, pages 157–174, 2001.

Erik Dahmen, Katsuyuki Okeya, Tsuyoshi Takagi, and Camille Vuillaume. **Digital Signatures Out of Second-Preimage Resistant Hash Functions**. In: Johannes Buchmann and Jintai Ding (editors), **Post-Quantum Cryptography: Second International Workshop, PQCrypto 2008 Cincinnati, OH, USA, October 17-19, 2008 Proceedings**. pages 109–123, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

A. Dehbaoui, J. M. Dutertre, B. Robisson, and A. Tria. Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES. In: **2012 Workshop on Fault Diagnosis and Tolerance in Cryptography**, pages 7–15, Sept 2012.

Dennis White Dennis Stanton. **Constructive Combinatorics**. Springer-Verlag New York, U.S.A., 1986.

W. Diffie and M. Hellman. New Directions in Cryptography. **IEEE Trans. Inf. Theor.**, 22(6):644–654, Setembro 1976. ISSN 0018-9448.

J. Ding, C. Wolf, and B.Y. Yang. l-invertible cycles for multivariate quadratic public cryptography. **Lecture Notes in Computer Science**, 4450:266–281, 2007.

I. Dumer. On minimum distance decoding of linear codes. pages 50–52, 1991a.

IIlya Dumer. On minimum distance decoding of linear codes. In: **Proceedings of 5th Joint Soviet-Swedish International Workshop on Information Theory**, pages 50–52, Moscow, 1991b.

T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. **IEEE Transactions on Information Theory**, 31(4):469–472, Jul 1985.

Matthieu Finiasz and Nicolas Sendrier. **Security Bounds for the Design of Code-Based Cryptosystems**. In: Mitsuru Matsui (editor), **Advances in Cryptology – ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings**. pages 88–105, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

P. Gaborit and M. Girault. Lightweight code-based identification and signature. In: **2007 IEEE International Symposium on Information Theory**, pages 191–195, June 2007.

Nahid Farhady Ghalaty. Fault Attacks on Cryptosystems: Novel Threat Models, Countermeasures and Evaluation Metrics, 2016.

Juan Grados, Fábio Borges, Renato Portugal, and Pedro C. S. Lara. An Efficient One-Bit Model for Differential Fault Analysis on Simon Family. In: **2015 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2015, Saint Malo, France, September 13, 2015**, pages 61–70, 2015.

Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In: **Proceedings of the 28th Annual ACM Symposium on Theory of Computing**, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM. ISBN 0-89791-785-5.

A. Hefez and M. L. T. Villela. **Códigos Corretores de Erros**. Instituto de Matemática Pura e Aplicada IMPA, Rio de Janeiro, 2008.

Ludger Hemme. A Differential Fault Attack Against Early Rounds of (Triple-)DES. In: Marc Joye and Jean-Jacques Quisquater (editors), **Cryptographic**

Hardware and Embedded Systems - CHES 2004, pages 254–267, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Alejandro Hevia and Daniele Micciancio. **The Provable Security of Graph-Based One-Time Signatures and Extensions to Algebraic Signature Schemes**. In: Yuliang Zheng (editor), **Advances in Cryptology — ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings**. pages 379–396, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

G. Hoffmann. Implementation of McEliece using quasi-dyadic Goppa codes. Relatório Técnico 2, Technical University of Darmstadt, 2011.

Joshua Holden. **The mathematics of secrets: cryptography from Caesar ciphers to digital encryption**. Princeton University Press, Princeton, 2017. ISBN 0691141754.

Andreas Hülsing. Forward Secure Signatures using Minimal Security Assumptions", 2013a.

Andreas Hülsing. **W-OTS+ – Shorter Signatures for Hash-Based Signature Schemes**. In: Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien (editors), **Progress in Cryptology – AFRICACRYPT 2013: 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings**. pages 173–188, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013b.

M. Hutter, J. M. Schmidt, and T. Plos. Contact-based fault injections and power analysis on RFID tags. In: **2009 European Conference on Circuit Theory and Design**, pages 409–412, Aug 2009.

Michael Hutter and Jörn-Marc Schmidt. The Temperature Side Channel and Heating Fault Attacks. **IACR Cryptology ePrint Archive**, 2014:190, 2014.

Michael Hutter, Jörn-Marc Schmidt, and Thomas Plos. RFID and Its Vulnerability to Faults. In: Elisabeth Oswald and Pankaj Rohatgi (editors), **Cryptographic Hardware and Embedded Systems – CHES 2008**, pages 363–379, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

G. Kabatianskii, E. Krouk, and B. Smeets. **A digital signature scheme based on random error-correcting codes**. In: Michael Darnell (editor), **Crytography and Coding: 6th IMA International Conference Cirencester, UK, December 17–19, 1997 Proceedings**. pages 161–167, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

David Kahn. **The codebreakers: the story of secret writing**. Scribner, New York, 1996. ISBN 0684831309.

Petteri Kaski and Patric R. J. Östergård. **Classification Algorithms for Codes and Designs (Algorithms and Computation in Mathematics)**. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

R. E. Klima, N. Sigmon, and E. Stitzinger. **Applications of Abstract with MAPLE**. CRC Press LLC, U.S.A., 2000.

Philip Lafrance and Alfred Menezes. On the security of the WOTS-PRF signature scheme. Cryptology ePrint Archive, Report 2017/938, 2017.

L. Lamport. Constructing digital signatures from a one-way function. Relatório técnico, SRI International Computer Science Laboratory, Outubro 1979.

J. H. Van Lint. **Introduction to Coding Theory**. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edição, 1998.

László Lovász. Random Walks on Graphs: A Survey, 1993.

Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. **SIAM J. Comput.**, 17(2):373 – 386, Abril 1988.

F. J. MacWilliams and N. J. A. Sloane. **The Theory of Error Correcting Codes**. North-Holland Publishing Company, 1 edição, 1977.

F. J. MacWilliams and N. J. A. Sloane. **The Theory of Error Correcting Codes**, volume 16. North-Holland Publishing Company, U.S.A, 1997.

R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. **The Deep Space Network Progress Report**, pages 114–116, 1978.

Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. **Handbook of Applied Cryptography**. CRC Press, Inc., Boca Raton, FL, USA, 1st edição, 1996.

RalphC. Merkle. A Certified Digital Signature. In: Gilles Brassard (editor), **Advances in Cryptology - CRYPTO' 89 Proceedings**, volume 435 of **Lecture Notes in Computer Science**, pages 218–238. Springer New York, 1990.

R. Misoczki and P. Barreto. Compact McEliece keys from Goppa codes. **Lecture Notes in Computer Science**, 5867:376–392, 2009.

Rafael Misoczki. UMA FAMÍLIA DE CÓDIGOS CORRETORES DE ERRO PARA CRIPTOSSISTEMAS EFICIENTES BASEADOS EM DECODIFICAÇÃO DE SÍNDROMES. Dissertação de Mestrado, 2010.

Robert Niebuhr, Pierre-Louis Cayrel, and Johannes Buchmann. Improving the efficiency of Generalized Birthday Attacks against certain structured cryptosystems. In: **WCC 2011 - Workshop on coding and cryptography**, pages 163–172, Paris, France, Abril 2011.

H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. **Problems of Control and Information Theory**, 15(2):159–166, 1986.

Ayoub Otmani and Jean-Pierre Tillich. **An Efficient Attack on All Concrete KKS Proposals**. In: Bo-Yin Yang (editor), **Post-Quantum Cryptography:**

4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 – December 2, 2011. Proceedings. pages 98–116, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

David Pointcheval and Jacques Stern. **Security Proofs for Signature Schemes**. In: Ueli Maurer (editor), **Advances in Cryptology — EUROCRYPT '96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings**. pages 387–398, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. **Journal of Cryptology**, 13(3):361–396, Jun 2000.

E. Prange. The use of information sets in decoding cyclic codes. **IRE Transactions on Information Theory**, 8(5):5–9, September 1962.

R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. **Commun. ACM**, 21(2):120–126, Fevereiro 1978. ISSN 0001-0782.

C. P. Schnorr. **Efficient Identification and Signatures for Smart Cards**. In: Gilles Brassard (editor), **Advances in Cryptology — CRYPTO' 89 Proceedings**. pages 239–252, Springer New York, New York, NY, 1990.

C. E. Shannon. A mathematical theory of communication. **The Bell System Technical Journal**, 27(3):379–423, July 1948.

Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. **SIAM J. Comput.**, 26(5):1484–1509, Outubro 1997. ISSN 0097-5397.

Vladimir Vasilevich Shurenkov and Vyacheslav Sergeevich Pershenkov. ELECTROMAGNETIC PULSE EFFECTS AND DAMAGE MECHANISM ON THE SEMICONDUCTOR ELECTRONICS. In: **Facta Universitatis, Series: Electronics and Energetics**, volume 29, pages 621–629, 2016.

Sergei P. Skorobogatov. Semi-invasive attacks - A new approach to hardware security analysis, 2005.

J. Stern. A new identification scheme based on syndrome decoding. **Lecture Notes in Computer Science**, (973):13–21, 1993.

J. Stern. Can one design a signature scheme based on error-correcting codes? **Lecture Notes in Computer Science**, (917):424–426, 1994.

Madhu Sudan. Coding Theory: Tutorial and Survey, 2001.

Michael Szydlo. Merkle Tree Traversal in Log Space and Time. In: Christian Cachin and JanL. Camenisch (editors), **Advances in Cryptology - EUROCRYPT 2004**, volume 3027 of **Lecture Notes in Computer Science**, pages 541–554. Springer Berlin Heidelberg, 2004.

Junko Takahashi and Toshinori Fukunaga. Fault Analysis on SIMON Family of Lightweight Block Ciphers. In: Jooyoung Lee and Jongsung Kim (editors), **Information Security and Cryptology - ICISC 2014**, volume 8949 of **Lecture Notes in Computer Science**, pages 175–189. Springer International Publishing, 2015.

H. Tupsamudre, S. Bisht, and D. Mukhopadhyay. Differential Fault Analysis on the Families of SIMON and SPECK Ciphers. In: **Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on**, pages 40–48, Sept 2014.

J. G. J. van Woudenberg, M. F. Witteman, and F. Menarini. Practical Optical Fault Injection on Secure Microcontrollers. In: **2011 Workshop on Fault Diagnosis and Tolerance in Cryptography**, pages 91–99, Sept 2011.

V. I. Villányi. Promising one-time signature schemes (survey). In: **Computational Intelligence and Informatics (CINTI), 2010 11th International Symposium on**, pages 187–192, Nov 2010.

David Wagner. A Generalized Birthday Problem. In: Moti Yung (editor), **Advances in Cryptology — CRYPTO 2002**, pages 288–304, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

Qingju Wang, Zhiqiang Liu, Kerem Varici, Yu Sasaki, Vincent Rijmen, and Yosuke Todo. Cryptanalysis of Reduced-Round SIMON32 and SIMON48. In: Willi Meier and Debdeep Mukhopadhyay (editors), **Progress in Cryptology – INDOCRYPT 2014**, Lecture Notes in Computer Science, pages 143–160. Springer International Publishing, 2014.

B. Yuce, N. F. Ghalaty, and P. Schaumont. TVVF: Estimating the vulnerability of hardware cryptosystems against timing violation attacks. In: **2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)**, pages 72–77, May 2015.

Appendix A

Equations of the last two rounds

A flip in $L^{T-2}_{(j+1)\%n}$ affects 3 bits:

$$\left(R^T \oplus R^{T^*}\right)_{(j+2)\%n} = \left(L^{T-2}_{(j+1)\%n} \odot L^{T-2}_{(j-6)\%n}\right) \oplus \left(\left(L^{T-2}_{(j+1)\%n} \oplus 1\right) \odot L^{T-2}_{(j-6)\%n}\right) \oplus \tilde{R}^{T-2}_{(j+2)\%n}$$

$$\left(R^T \oplus R^{T^*}\right)_{(j+3)\%n} = \begin{cases} \left(L^{T-2}_{(j+2)\%n} \odot L^{T-2}_{(j-5)\%n}\right) \oplus \left(\left(L^{T-2}_{(j+2)\%n} \oplus 1\right) \odot L^{T-2}_{(j-5)\%n}\right) \oplus 1 \oplus \tilde{R}^{T-2}_{(j+3)\%n} \\ \text{if } L_{(j+2)\%n} \text{ was affected} \\ 1 \oplus \tilde{R}^{T-2}_{(j+3)\%n} \\ \text{if otherwise} \end{cases}$$

$$\left(R^T \oplus R^{T^*}\right)_{(j+9)\%n} = \begin{cases} \left(L^{T-2}_{(j+8)\%n} \odot L^{T-2}_{(j+1)\%n}\right) \oplus \left(\left(L^{T-2}_{(j+8)\%n} \oplus 1\right) \odot \left(L^{T-2}_{(j+1)\%n} \oplus 1\right)\right) \oplus \tilde{R}^{T-2}_{(j+9)\%n} \\ \text{if } L_{(j+8)\%n} \text{ was affected} \\ \left(L^{T-2}_{(j+8)\%n} \odot L^{T-2}_{(j+1)\%n}\right) \oplus \left(\left(L^{T-2}_{(j+8)\%n}\right) \odot \left(L^{T-2}_{(j+1)\%n} \oplus 1\right)\right) \oplus \tilde{R}^{T-2}_{(j+9)\%n} \\ \text{if otherwise} \end{cases}$$

A flip in $L^{T-2}_{(j+2)\%n}$ affects 3 bits:

$$\left(R^T \oplus R^{T^*}\right)_{(j+3)\%n} = \begin{cases} \left(L^{T-2}_{(j+2)\%n} \odot L^{T-2}_{(j-5)\%n}\right) \oplus \left(\left(L^{T-2}_{(j+2)\%n} \oplus 1\right) \odot L^{T-2}_{(j-5)\%n}\right) \oplus 1 \oplus \tilde{R}^{T-2}_{(j+3)\%n} \\ \text{if } L_{(j+1)\%n} \text{ was affected} \\ \left(L^{T-2}_{(j+2)\%n} \odot L^{T-2}_{(j-5)\%n}\right) \oplus \left(\left(L^{T-2}_{(j+2)\%n} \oplus 1\right) \odot L^{T-2}_{(j-5)\%n}\right) \oplus \tilde{R}^{T-2}_{(j+3)\%n} \\ \text{if otherwise} \end{cases}$$

$$\left(R^T \oplus R^{T^*}\right)_{(j+4)\%n} = L^{T-2}_{(j+2)\%n} \oplus (L^{T-2}_{(j+2)\%n} \oplus 1) \oplus \tilde{R}^{T-2}_{(j+4)\%n} = 1 \oplus \tilde{E}^{T-2}_{(j+4)\%n}$$

$$\left(R^T \oplus R^{T^*}\right)_{(j+10)\%n} = \begin{cases} \left(L^{T-2}_{(j+9)\%n} \odot L^{T-2}_{(j+2)\%n}\right) \oplus \left(L^{T-2}_{(j+9)\%n} \odot \left(L^{T-2}_{(j+2)\%n} \oplus 1\right)\right) \oplus 1 \oplus \tilde{R}^{T-2}_{(j+10)\%n} \\ \text{if } L_{(j+8)\%n} \text{ was affected} \\ \left(L^{T-2}_{(j+9)\%n} \odot L^{T-2}_{(j+2)\%n}\right) \oplus \left(L^{T-2}_{(j+9)\%n} \odot \left(L^{T-2}_{(j+2)\%n} \oplus 1\right)\right) \oplus \tilde{R}^{T-2}_{(j+10)\%n} \\ \text{if otherwise} \end{cases}$$

A flip in $L^{T-2}_{(j+8)\%n}$ affects 3 bits:

$$\left(R^T \oplus R^{T^*}\right)_{(j+9)\%n} = \begin{cases} \neg \left(L^{T-2}_{(j+8)\%n} \oplus L^{T-2}_{(j+1)\%n}\right) \oplus \tilde{R}^{T-2}_{(j+9)\%n} \\ \text{if } L_{(j+1)\%n} \text{ was affected} \\ \left(L^{T-2}_{(j+8)\%n} \odot L^{T-2}_{(j+1)\%n}\right) \oplus \left(L^{T-2}_{(j+1)\%n} \odot \left(L^{T-2}_{(j+8)\%n} \oplus 1\right)\right) \oplus \tilde{R}^{T-2}_{(j+9)\%n} \\ \text{if otherwise} \end{cases}$$

$$\left(R^T \oplus R^{T^*}\right)_{(j+10)\%n} = \begin{cases} \left(L^{T-2}_{(j+9)\%n} \odot L^{T-2}_{(j+2)\%n}\right) \oplus \left(L^{T-2}_{(j+9)\%n} \odot \left(L^{T-2}_{(j+2)\%n} \oplus 1\right)\right) \oplus \tilde{R}^{T-2}_{(j+10)\%n} \\ \text{if } L_{(j+2)\%n} \text{ was affected} \\ 1 \\ \text{if otherwise} \end{cases}$$

$$\left(R^T \oplus R^{T^*}\right)_{(j+16)\%n} = \left(L^{T-2}_{(j+15)\%n} \odot L^{T-2}_{(j+8)\%n}\right) \oplus \left(L^{T-2}_{(j+15)\%n} \odot \left(L^{T-2}_{(j+8)\%n} \oplus 1\right)\right) \oplus \tilde{R}^{T-2}_{(j+16)\%n}$$