TÉCNICO LISBOA

A new LDPC-based McEliece cryptosystem

Pedro de Melo Branco

Thesis to obtain the Master of Science Degree in

Matemática e Aplicações

Supervisor: Prof. Paulo Alexandre Carreira Mateus

Examination Committee

Chairperson: Prof. Maria Cristina De Sales Viana Serôdio Sernadas
Supervisor: Prof. Paulo Alexandre Carreira Mateus
Member of the Committee: Prof. André Souto

October 2017

Acknowledgments

It was a pleasure to work with Prof. Paulo Mateus. I would like to thank him for supervising this thesis.

Resumo

O sistema criptográfico McEliece foi apresentado por Robert McEliece e é um dos mais antigos criptosistemas de chave-pública que ainda estão por quebrar. A sua simplicidade e a sua eficiência tornam-no num candidato interessante para a era pós-quântica dado que se conjectura que é seguro contra ataques quânticos.

Nesta dissertação, analisamos o sistema criptográfico McEliece. Vamos apresentar os seus prós e contras assim como as suas fundações. Vamos também apresentar alguns conceitos básicos de teoria dos códigos de modo a compreender totalmente o sistema criptográfico McEliece.

Propomos um sistema criptográfico eficiente baseado no McEliece para tratar blocos de mensagens grandes e que pode ser facilmente implementado em *hardware*. Para conseguirmos isso, usamos códigos *LDPC* no sistema criptográfico McEliece tirando partido das suas capacidades para tratar blocos de mensagens grandes. Conjectura-se que o sistema criptográfico proposto seja resistente a ataques quânticos visto que a sua segurança está alavancada na segurança do sistema criptográfico McEliece. Além disto, provamos que o sistema criptográfico é tão difícil de quebrar como o McEliece. Fomos capazes de reduzir significantemente o tamanho da chave do sistema criptográfico McEliece, um dos seus maiores problemas e a principal razão pela qual não é usado na práctica. Analisamos também a sua eficiência e propomos uma versão IND-CCA2 segura alavancadas em alguns problemas que se conjecturam difíceis.

Palavras-chave: criptografia, teoria de códigos, encriptação de chave-pública, sistema criptográfico McEliece, códigos LDPC, indistinguibilidade

Abstract

The McEliece cryptosystem was first presented by Robert McEliece and it is one of the oldest public-key cryptosystem that remains unbreakable. Its simplicity and its efficiency makes it a very interesting candidate for the post-quantum era since it is conjectured to be secure against a quantum computer.

In this thesis, we analyze the McEliece cryptosystem. We will go throughout its pros and cons and its foundations. Also, we present some basic concepts of coding theory in order to fully understand the McEliece cryptosystem.

Also, we propose an efficient McEliece-based cryptosystem to handle large messages and that can be easily implemented in hardware. To achieve that, we will use LDPC codes in the McEliece cryptosystem taking advantage of their capacity to handle large blocks of messages. The cryptosystem proposed is conjectured to be robust to quantum attacks since it relies its security of the McEliece cryptosystem. Moreover, we prove that this cryptosystem is at least as hard to break as the McEliece. We were capable of reducing significantly the key size of the cryptosystem, one of its major problems and the principal reason why it is not used in the real world. We also analyze its efficiency and propose an IND-CCA2 secure variant under some hard assumptions.

Keywords: cryptography, coding theory, pubic-key encryption, McEliece cryptosystem, LDPC codes, indistinguishability

Contents

Chapter 1

Introduction

Cryptography is concerned with providing ways of secure communication between two parties and in the presence of a malicious third party. Until the 20^{th} century, cryptography was mainly concerned on the design of encryption schemes using codes or similar techniques. But with the advent of the internet, cryptography has become a crucial subject in our everyday life. Researchers started to use mathematics for the design of cryptographic protocols, and right now, cryptography is considered a subfield of mathematics.

Post-quantum cryptography is a subfield of cryptography that deals with classical cryptographic algorithms that are conjectured to be robust against quantum attacks. It is a growing area since Shor proved that quantum computer can break the cryptographic protocols that are most used nowadays [51], such as RSA [47] and El Gamal cryptosystems [17]. For this reason, many believe post-quantum protocols will be widely used in a near future. Post-quantum cryptography is based on hard problems that are believed to be unsolvable by quantum computers such as problems based on lattices, codes or multivariate polynomials [10].

In this thesis we will focus our attention mainly in a code-based encryption protocol: the McEliece cryptosystem [39]. This one of the oldest cryptosystems that remains unbreakable. It was proposed by Robert McEliece in 1978, shortly after RSA cryptosystem was presented. The security cryptosystem is based on problems that are conjectured to be unsolvable by quantum computers and thus the cryptosystem is conjectured to be robust against quantum polynomial-time attacks. But the McEliece cryptosystem has still some problems namely its key size, which is probably its main problem [18], and its decryption time [38]. So, this cryptosystem is being deeply studied by cryptographers, not just to reduce its key size, but also in order to enhance its security against known attacks and to enhance its encryption and decryption times.

Goppa codes [5] is the family of codes used in the original McEliece. There have been attempts to use other families of codes in the cryptosystem but they all failed (see, for example, [41, 52]. The problem with Goppa codes is that their decoding algorithm does not scale well turning the decryption algorithm of the McEliece cryptosystem too slow for large messages, specially in hardware implementations [38].

We propose a new McEliece-based cryptosystem that uses Goppa codes, the family used in the

original McEliece, and LDPC codes, a family of codes based on graphs and that allows fast decoding on hardware. This new construction achieves fast encryption and decryption both in software and in hardware and scales very well for large messages, solving the problem presented above. Also, with this construction we are able to reduce the key size up to ten times compared to the original McEliece. However, this construction does not meet some important security notions such as indistinguishability or non-malleability. So we present a variant of the construction that does meet these security notions and that can be used in practice with very little extra cost.

Thesis outline

In Chapter 2 of this thesis, we start by introducing some basic notions on coding theory, in order to introduce latter in the chapter two families of codes: the family of Goppa codes, which is a very important family of codes for cryptography, and the family of LDPC codes, where the recent research in coding theory has been focused. The introduction of these two families of codes is essential for the understanding of the next chapters. In Chapter 3, we continue to introduce basic concepts, but now on public-key cryptography, a crucial idea in cryptography. Also, we define some security notions that we will need for our study.

After the reader is familiarized with some basic concepts of coding theory, with these classes of codes and with cryptography, in Chapter 4 we explain how codes can be used in cryptography, namely by introducing a couple of code-based cryptosystems: the McEliece and the Niederreiter cryptosystems. We will mainly focus our attention on the security and efficiency of the McEliece cryptosystem. At the same time, we will give some important security definitions.

The main results of this thesis are presented in Chapter 5, where we propose a new code-based cryptosystem using Goppa and LDPC codes. We also present some variations that achieve some important security notions (namely IND-CPA and IND-CCA2) and the corresponding security proofs.

Finally, in Chapter 6 we exposed some thoughts to conclude this thesis along with some directions of future work.

Chapter 2

Coding Theory

In the real world, when we want to send a message to someone else, usually this message is sent through a noisy channel. This means that the receiver will probably received the message with errors, tipically flipped bits (where is supposed to be a 1, it appears 0 in the received message for example) or erased bits. We can solve this problem by using codes that detect these errors and, ideally, correct them. Coding theory is the study of codes and their properties. It appeared in the 40's, mainly due to the work developed by Claude Shannon [49] and, since then, it is a growing area. Since it has many applications in the real world, coding theory is a very interesting subject for mathematicians, information scientist, electrical engineers, computer scientists, etc. The studies on coding theory tipically involve trying to make efficient construction of codes and finding efficient algorithms of detection and correction of errors in those codes.

We will begin this chapter with a brief introduction to coding theory, presenting some notation and basic results in Section 2.1. Then, in Section 2.2 we will focus on Goppa codes, a family of codes which is very useful for cryptography. Finally, in Section 2.3, we will introduce LDPC codes, a family of codes that was recently presented; we will begin with some background on graph theory and then we will present some constructions of these codes.

2.1 Basic Notions

We begin this section with some notation: if S is a set, $|S|$ denotes the number of elements in S. We will denote a *field* by $\mathbb{F}$. A *finite field* with q elements will be denoted by $\mathbb{F}_q$, where q is a prime number or a power of a prime number. By a *vector space* $\mathbb{F}_q^n$, we mean a vector space where the vectors have coordinates in $\mathbb{F}_q$.[1] We define the *Hamming distance* $d(v,u)$ of two vectors v,u in a vector space F_q^n by the number of coordinates of these vectors that are different from each other, i.e,

$$d(v,u) = \#\{i : v_i \neq u_i\}.$$

[1] Throughout this thesis, we will often refer to the field with two elements, $\mathbb{F}_2$, by $\mathbb{Z}_2 = \mathbb{Z}/(2) = \{0,1\}$.

The *Hamming weight* of a vector $v \in V$ is the number of coordinates that are non-null. We will denote it by $wt(v)$.

A *linear code* $\mathcal{C}$ of length n over some field $\mathbb{F}_q$ is a subspace of $\mathbb{F}_q^n$. The minimum Hamming distance, or minimum distance, of a code $\mathcal{C}$ is the minimum distance between any two words[2] of the code. By a $[n, k, d]_q$ code we mean a code $\mathcal{C}$ whose codewords have symbols in $\mathbb{F}_q$, with length n, dimension k and minimum Hamming distance d between any two codewords. Throughout this thesis, when we refer a code $\mathcal{C}$, we mean a $[n, k, d]_q$ code, unless stated otherwise. Also, when we omit q, it means that $q = 2$.

Finding the minimum distance of a given code is not a trivial problem, in fact it is a NP-complete problem [59]; but there are some classes of codes for which this value is known or, at least, some bounds for it [6, 25].

A linear code $\mathcal{C}$ has a *generating matrix* G, whose lines form a basis of $\mathcal{C}$. It also has a *parity-check matrix* H, whose null space is $\mathcal{C}$ and such that $GH^T = 0$. We say that G is written in the *systematic form*, or *standard form*, if

$$G = \left(I_k | G'\right)$$

where I_k is the identity matrix of size k and G' is a matrix with k rows and $n - k$ columns. If $G = \left(I_k | G'\right)$ is written in the systematic form, we can derive the parity-check matrix H very easily:

$$H = \left(-G'^T | I_{n-k}\right).$$

Obviously, if G and H are associated with a binary code (which is defined in $\mathbb{Z}_2$), we have $-G' = G'$.

A code with parameters $[n, k, d]$ can correct at most t errors, where $t \leq \frac{d-1}{2}$. Its redundancy is $r = n - k$ (i.e. it is the number of extra bits we get by encoding a message). By rate R of a code $\mathcal{C}$ we mean the fraction of bits of a codeword that is not redundant, i.e, $R = k/n$.

Example 1. *[Hamming code] These class of linear codes were invented by Richard Hamming in the early 50's and it is a classical example of a linear code. The columns of the parity-check matrix H of a Hamming code are all the non-null vectors of $\mathbb{F}_2^r$, where r is the redundancy of the code. This codes are denoted by $Ham(r, 2)$.*

The matrix

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

is a parity-check matrix of $Ham(3, 2)$. This code has parameters $[7, 4, 3]$ so it can correct up to $t \leq \frac{3-1}{2} = 1$ error.

By a decoding algorithm $D_{\mathcal{C}}$ of a code $\mathcal{C}$ that correct t errors, we mean an algorithm such that given an input word $w + e$, where $w \in \mathcal{C}$ and e is a vector with weight $wt(e) \leq t$, it outputs w.

Given a code $\mathcal{C}$ (with length n, over $\mathbb{F}_q$) and a parity-check matrix H of $\mathcal{C}$, a syndrome of a vector $w \in \mathbb{F}_q^n$ is the product wH. If the syndrome of a vector w is $wH = 0$ then $w \in \mathcal{C}$. On the other hand, if we

[2] A word is the same as a vector; we will often use this terminology.

compute the syndrome of a vector $y = w + e$, where $w \in \mathcal{C}$ and e is an error vector such that $wt(e) \leq t$, we get

$$yH = (w + e)H = wH + eH = eH.$$

So, the syndrome can be used for decoding, since the decoding problem is equal to finding the error vector e such that $yH = eH$.

Given a code $\mathcal{C}$ and a parity matrix H of $\mathcal{C}$, we define $S_H(s)^{-1}$ as the set of words such that have syndrome s, i.e,

$$S_H(s)^{-1} = y + \mathcal{C} = \{y + w : yH = s, w \in \mathcal{C}\}.$$

Finding a vector $y \in S_H(s)^{-1}$ can be done in polynomial time. But the following problem is known to be computationally hard.

Problem 2 (Computational Syndrome Vector)**.** *Given a code $\mathcal{C} \in \mathbb{Z}_2^n$ of dimension $k \leq n$, a parity-check matrix H of $\mathcal{C}$, a syndrome $s \in \mathbb{Z}_2^r$ and $t \in \mathbb{N}$, find a word $x \in \mathbb{Z}_2^n$ such that $x \in S_H(s)^{-1}$ and x has weight $wt(x) \leq t$.*

This problem was proven to be NP-complete for an arbitrary linear code [7]. If we want to minimize t, then the problem is in NP-hard. Hence, if the parameters are large enough, the problem becomes infeasible to solve. What this problem tells us is that decoding a corrupted codeword of an arbitrary code is a non trivial problem.

One of the main goals of coding theory is to find families of codes such that the Computational Syndrome Vector problem is easy to solve, i.e., is in P (or BPP). If we can do that efficiently for a particular family of codes, we have a family of codes for which decoding is efficient. Also, there is no known quantum algorithm to solve this problem, which makes this problem interesting for post-quantum cryptography as we will see later on.

2.2 Goppa Codes

In this thesis we are interested in families of codes that we can use in cryptography and one of the most important family of those codes is the Goppa codes family. They were first presented in [23], and latter an English version was published in [5]. We will now see what a Goppa code is and what are its properties.

A *Goppa code* is a binary error-correcting code that has some interesting properties for cryptography, namely the number of errors that it can correct and the number of different codes that we can create, for a fixed length and minimum distance.

Let $p \in \mathbb{N}$. A Goppa code is defined by (g, L) where $g \in \mathbb{F}_{2^p}[x]$ is a polynomial of degree t (with coefficients in $\mathbb{F}_{2^p}$ and variable x), and without multiple zeros, and L is a sequence $L_0, ..., L_{n-1} \in \mathbb{F}_{2^p}$ such that

$$L_i \neq L_j \wedge g(L_i) \neq 0 \quad \forall i, j \in \{0, ..., n-1\},$$

i.e., all the elements of L must be different from each other and none of them is a root of g. The code is

the set

$$\mathcal{G}(g,L)=\left\{w\in\mathbb{Z}_2^n:\sum_{i=0}^{n-1}\frac{w_i}{x-L_i}=0 \mod g(x)\right\}.$$

Theorem 3 ([8]). *For a giving pair $(g, L=\{L_0,\ldots,L_{n-1}\})$, the parity-check matrix of the corresponding code is very easy to derive. It can be computed in the following way:*

$$H=\begin{pmatrix}\frac{1}{g(L_0)} & \cdots & \frac{1}{g(L_{n-1})}\\ \frac{L_0}{g(L_0)} & \cdots & \frac{L_{n-1}}{g(L_{n-1})}\\ \vdots & & \vdots\\ \frac{L_0^{t-1}}{g(L_0)} & \cdots & \frac{L_{n-1}^{t-1}}{g(L_{n-1})}\end{pmatrix}.$$

The ease with which we create different Goppa codes comes from the ease with which we create different parity-check matrices.

This code has minimum distance greater or equal than $2t+1$ since it can correct up to t errors (why it can correct t errors will be explained latter). The dimension of this code is $k=n-mt$. Using the Patterson algorithm [45] one can decode a corrupt Goppa codeword easily. The existence of such a decoding algorithm is what makes Goppa codes useful for applications. A simple example of a Goppa code is presented in Example 4.

Example 4. *Let $p=4$ and consider the polynomial $f(x)=x^4+x+1$ which is irreducible in $\mathbb{F}_2$. We have that $\mathbb{F}_{16}=\mathbb{Z}_2[x]/\langle f(x)\rangle$ is a field. Let α be a solution for the equation $x^4+x+1=0$ in $\mathbb{F}_{16}$; α is a primitive element of $\mathbb{F}_{16}$, i.e., it is a generator for the multiplicative group $\mathbb{F}_{16}^*=\mathbb{F}_{16}\setminus\{0\}$.*

Now consider the polynomial $g(x)=x^2+x+\alpha^3$ in $\mathbb{F}_{16}[x]$ and $L=\{\alpha^i:2\leq i\leq 13\}$ a set of elements of $\mathbb{F}_{16}$. The code defined by g and l has length 12, minimum distance $d\geq 2t+1=5$ and dimension 4. So its parameters are $[12,4,d]$ where $d\geq 5$.

$$\begin{aligned}H&=\begin{pmatrix}\frac{1}{g(\alpha^2)} & \frac{1}{g(\alpha^3)} & \frac{1}{g(\alpha^4)} & \frac{1}{g(\alpha^5)} & \frac{1}{g(\alpha^6)} & \frac{1}{g(\alpha^7)} & \frac{1}{g(\alpha^8)} & \frac{1}{g(\alpha^9)} & \frac{1}{g(\alpha^{10})} & \frac{1}{g(\alpha^{11})} & \frac{1}{g(\alpha^{12})} & \frac{1}{g(\alpha^{13})}\\ \frac{\alpha^2}{g(\alpha^2)} & \frac{\alpha^3}{g(\alpha^3)} & \frac{\alpha^4}{g(\alpha^4)} & \frac{\alpha^5}{g(\alpha^5)} & \frac{\alpha^6}{g(\alpha^6)} & \frac{\alpha^7}{g(\alpha^7)} & \frac{\alpha^8}{g(\alpha^8)} & \frac{\alpha^9}{g(\alpha^9)} & \frac{\alpha^{10}}{g(\alpha^{10})} & \frac{\alpha^{11}}{g(\alpha^{11})} & \frac{\alpha^{12}}{g(\alpha^{12})} & \frac{\alpha^{13}}{g(\alpha^{13})}\end{pmatrix}\\ &=\begin{pmatrix}\alpha^3 & \alpha^9 & \alpha^4 & \alpha & \alpha^8 & \alpha^6 & \alpha^3 & \alpha^6 & \alpha & \alpha^2 & \alpha^2 & \alpha^8\\ \alpha^5 & \alpha^{12} & \alpha^8 & \alpha^6 & \alpha^{14} & \alpha^{13} & \alpha^{11} & \alpha^{15} & \alpha^{10} & \alpha^{13} & \alpha^{14} & \alpha^{21}\end{pmatrix}.\end{aligned}$$

Writing the vectors in the basis $\{1,\alpha,\alpha^2,\alpha^3\}$, we get

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

and since $GH^T = 0$, we can compute the generator matrix G, for the code generated by the pair (g, L).

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

Patterson's algorithm

To decode Goppa codes we can use *Patterson's algorithm* which is decribed in Algorithm 1 in page 9. This procedure, presented in [45], can correct up to t errors.

Before we present the algorithm, note that we can view a codeword of length n as a polynomial with degree $n-1$ in a variable x. So, the syndrome of a received word y is given by

$$s_y(x) = \sum_{i=0}^{n-1} \frac{y_i}{x - L_i} \mod g(x)$$

by the definition of Goppa code. So, if the received word y is $y = w + e$, where w is a codeword and e an error vector such that $wt(e) \leq t$, we have

$$s_y(x) = \sum_{i=0}^{n-1} \frac{y_i}{x - L_i} = \sum_{i=0}^{n-1} \frac{w_i}{x - L_i} + \frac{e_i}{x - L_i} = \sum_{i=0}^{n-1} \frac{e_i}{x - L_i} = s_e(x)$$

and so, the syndrome of y is the same as the error vector e (as expected).

We define the *error locator polynomial* of $y = w + e$ where $w \in \mathcal{C}$ and e is such that $wt(e) \leq t$.

$$\sigma_y(x) = \prod_{i:e_i \neq 0} (x - L_i)$$

whose roots give the position of the errors and, obviously, $\deg \sigma_y$ gives the number of errors. If we find the error locator polynomial associated with a corrupt word, we can correct the error. In our case, we will assume $\deg \sigma_y \leq t$, otherwise we are not able to correct the errors. Let $\sigma_y'(x)$ be the derivative of $\sigma_y(x)$.

We have that:

$$\sigma'_y(x) = \sum_{i:e_i \neq 0} \prod_{\substack{j:e_j \neq 0 \\ j \neq i}} (x - L_j) = \sum_{i:e_i \neq 0} \frac{1}{x - L_i} \prod_{j:e_j \neq 0} (x - L_j) = s_e(x)\sigma_y(x).$$

So, we aim to find $\sigma_y(x)$ and $\sigma'_y(x)$ that satisfy the equation $s_e(x)\sigma_y(x) = \sigma'_y(x) \mod g(x)$ with $\deg \sigma_y(x) \leq t$ and $\deg \sigma'_y(x) \leq t - 1$.

Recall that $(a + b)^2 = a^2 + b^2$ in a field with characteristic 2. Since σ_y is a polynomial defined over a field with characteristic 2, it can be decomposed in $\alpha^2(x) + x\beta^2(x)$, with $\deg(\alpha) \leq \lfloor t/2 \rfloor$ and $\deg \beta \leq \lfloor \frac{t-1}{2} \rfloor$. Hence, $\sigma'_y(x) = \beta^2(x)$. We have

$$s_e(x)\sigma_y(x) = \sigma'_y(x) \Leftrightarrow s_e(x)(\alpha^2(x) + x\beta^2(x)) = \beta^2(x) \Leftrightarrow \alpha(x) = \beta(x)d(x) \mod g(x)$$

where $d(x)^2 = x + s_e(x)^{-1}$. To efficiently compute $d(x)$, we have the following lemma.

Lemma 5 ([48]). *Let $x + s_e^{-1}(x) = t_0^2(x) + xt_1^2(x) \in \mathbb{F}_{2^p}[x]/\langle g(x) \rangle$ and $g(x) = h_0^2(x) + xh_1^2(x)$. Let $d(x) = t_0(x) + h_0(x)h_1^{-1}(x)t_1(x)$. Then $d(x) = \sqrt{x + s_e^{-1}(x)}$.*

There is an algorithm to find such α and β (a variant of Extended Euclidean algorithm [45], noting that $\alpha(x) = \beta(x)d(x) + g(x)k(x)$ for some polynomial $k(x)$), and then we can compute $\sigma_y(x)$ and find its zeros (by checking for which of the L_i we have $\sigma_y(L_i) = 0$). By decomposing

$$\sigma_y(x) = \prod_{i:e_i \neq 0} (x - L_i)$$

we can get the non-null coordinates of the error vector and thus correct the received corrupt word.

If the code is defined by a polynomial g with coefficients over $\mathbb{F}_{2^p}$ such that $\deg g = t$ and by a sequence of numbers L with size n, then this algorithm runs in time $\mathcal{O}(n.t.p^2)$ [10, 18].

2.3 LDPC Codes

Low-Density Parity-Check (LDPC) codes form a class of linear codes that are obtained from sparse bipartite graphs. Let X be a bipartite graph; the left nodes of X are called the *message nodes* (or *variable nodes*) and the right nodes are called the *check nodes* (or *constraint nodes*). We contruct an LDPC from X in the following way: given a word w, we associate each left vertex of the graph with each bit of w; w is a codeword if, for all check nodes, the sum of the neighbor bits (the bits associated with neighbor nodes) is zero.

In matrix terms, the parity-check matrix H is the adjacency matrix of X. Since X is a sparse graph, H is a sparse matrix. As usual, the code is defined by the set of words w such that $wH = 0$.

Example 6. *Consider the graph presented in Figure 2.1.*

The vertices on the left side are called variables, denoted by $v_1, \ldots, v_{10}$. The vertices on the right side are called constraints, denoted by $C_1, \ldots, C_5$. We associate each variable with a bit of a word.

Algorithm 1 Patterson's decoding algorithm

input: Corrupt word $y = w + e$, where w is a codeword and $wt(e) \leq t$;

output: Codeword w.

$w := y$;

Compute the syndrome of y, $s(x)$. Compute $1/s(x) = s(x)^{-1}$;

if $s(x)^{-1} = x$ **then**

$\sigma_y(x) = x$ and terminate;

else

$d(x) = \sqrt{x + s_e(x)^{-1}}$;

Find $\alpha(x)$ and $\beta(x)$ such that $\alpha(x) = \beta(x)d(x) \mod g(x)$;

$\sigma_y(x) = \alpha^2(x) + x\beta^2(x)$ and compute its roots. Let $S = \{i : \sigma(L_i) = 0\}$;

for each i in S **do**

flip the i^{th} bit of w;

end for

return w

end if

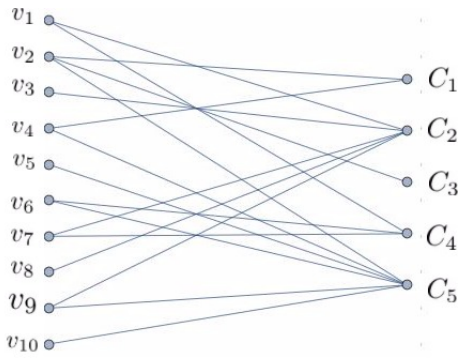

Figure 2.1: LPDC code.

The LDPC code defined by this graph is the set of words with length 10 such that, for each constraint $C_1, \ldots, C_5$, the sum of the adjacent variables is equal to 0. In other words, the code is the set of words $v = v_1 ... v_{10}$ that satisfy the following equations:

$$\begin{aligned} v_2 + v_4 &= 0 \\ v_1 + v_3 + v_7 + v_8 + v_9 &= 0 \\ v_2 &= 0 \\ v_1 + v_6 + v_7 &= 0 \\ v_2 + v_4 + v_5 + v_6 + v_9 + v_{10} &= 0. \end{aligned}$$

The parity-check matrix is

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

i.e., it is the adjacency matrix of the graph. Equivalently, we can define the code as the set of words w *such that* $wH = 0$.

The great advantage of LDPC codes is their decoding algorithm. These codes can be decoded using the so-called belief propagation techniques which make them extremely efficient [30, 36, 37, 50] (we will see latter examples of these techniques).

LDPC codes differ greatly from algebraic geometry codes (like, for example, Goppa codes). For the later, it can be extremely difficult to find decoding algorithms, since they are constructed using combinatorial arguments and the decoding is not taken into account when creating the code. On the other hand, LDPC codes have very easy and fast decoding algorithms, since the decoding is taken into account when creating the code.

Here, we will take a quick look on a couple of different families of LDPC codes. These families are the regular LDPC codes and the expander codes. But before we present these families, we will need some background on graph theory.

Graph Theory

We will begin with some basic definitions on graph theory. A *graph* X of size n is a tuple (V, E), where $V = \{v_1, \ldots, v_n\}$ is the set of vertices and $E = \{(u, v) : u, v \in V\}$ is the set of edges. The *degree* of a vertex is the number of edges that are connected to it.

Definition 7 (Regular graph)**.** *A* a*-regular graph is a graph where each vertex has degree* a*.*

By an (a, d)*-regular graph*, we mean a bipartite graph where: i) we can divide the vertex set into two disjoint sets such that there are no connections between vertices of the same set; and ii) all vertices in one set have degree a and all vertices in the other set have degree d.

Definition 8 (Cayley graph)**.** *Let* G *be a group and* S *a generating set of* G*. The* Cayley graph $X = X(G, S)$ *is the graph where each element* $g \in G$ *is assigned a vertex and the edge set* E *of* X *is formed by pairs* (g, gs) *for all* $g \in G$ *and* $s \in S$*.*

Definition 9 (Edge-vertex incidence graph)**.** *Let* X *be a graph. The edge-vertex incidence graph of* X *is a bipartite graph with vertex set* $V' = E \cup V$ *and edge set* $E' = \{(e, v) \in E \times V : v$ *is an endpoint of* $e\}$*.*

By a *factor expansion* δ we mean that, for every subset of vertices $S \subset V$ with at most a fixed number of vertices (i.e., $|S| \leq n$ for some $n \in \mathbb{N}$), the number of adjacent vertices of the elements of S is at least $\delta|S|$.

Definition 10 (Expander graph)**.** *A (a, d, ϵ, δ) expander graph is a (a, d)-regular graph, where V is the set of vertices with degree a, such that every $S \subset V$ expands by a factor of at least δ, where S has at most an ϵ fraction of the vertices with degree a.*

Expander graphs are sparse graphs that each small set of vertices have a lot of adjacency vertices.

The existence of these graphs was proven using probabilistic methods [27]. In fact, if we generate a random (a, d)-regular graph following simple heuristics, we will most likely get an expander graph [58]. Also, there is a deterministic way to construct expander graphs [33] which we will analyse.

There is a relation between expander graphs and the first and second largest eigenvalue (in absolute value) of the adjacency matrix of the graph [2]: the greater the difference between the first and the second eigenvalues, the greater the expansion factor of the graph. So now, we will study graphs in which this difference is maximal.

Let us now assume that X is a connected d-regular graph of n vertices. It is known that its adjacency matrix has eigenvalues $\lambda_0 > \lambda_1 \geq \cdots \geq \lambda_{n-1}$, where $\lambda_0 = d$ and $|\lambda_j| \leq d$ [33]. From now on we will define $\lambda(X) = \max_i |\lambda_i|$ such that $|\lambda_i| \neq d$.

Definition 11 (Ramanujan graph [33])**.** *A graph X is called a Ramanujan graph if $\lambda(X) \leq 2\sqrt{d-1}$.*

Asymptotically, this bound is the smallest possible value for $\lambda(X)$ [2], since it is proven that

$$\lim_{n\to\infty} \lambda(X) \geq 2\sqrt{d-1}.$$

So, for a Ramanujan graph, when the number of vertices n is great enough, the equality holds. And since the difference between the first and the second eigenvalues will be maximal, these graphs will have good expander factors (because of the relations between this difference and the expander factors referred above).

We need to introduce one last notion before we present the algorithm to create explicit Ramanujan graphs and that is the notion of *Projective Linear Group* and *Projective Special Linear Group*.

Definition 12 (Projective Linear Group)**.** *Let V be a vector space. We define the Projective Linear Group*

$$PGL_n(V) := GL_n(V)/Z_n(V)$$

where $GL_n(V)$ is the General Linear Group[3] of degree n and $Z_n(V) \leq GL_n(V)$ its center.[4] Identically, we define the Projective Special Linear Group $PSL_n(V) = SL_n(V)/SZ_n(V)$ where $SL_n(V)$ is the special linear group of size n and $SZ_n(V) \leq SL_n(V)$ its center. Recall that $SL_n(V)$ is a subgroup of $GL_n(V)$ formed by the matrices whose determinant is 1.

[3] The group of invertible matrices.

[4] The center of a group is the set of elements that commute with all other elements in the group.

For our purpose we are only interested in the case where $n = 2$ and $V = \mathbb{Z}_q$, for some prime number q.

The center of $GL_2(\mathbb{Z}_q)$ is the subgroup $Z_2(\mathbb{Z}_q)$ formed by the scalar matrices (the matrices that are a multiple of the identity).

By definition, we have that two elements of $GL_2(\mathbb{Z}_q)$ belong to the same equivalence class in $PGL_2(\mathbb{Z}_q)$ if they differ by multiplication of a scalar in $\mathbb{Z}_q$. In other words, for some $\alpha \in \mathbb{Z}_q^*$ and $A, B \in GL_2(\mathbb{Z}_q)$

$$A \sim B \iff A = \alpha B.$$

Proposition 13. *The order of $PGL_2(\mathbb{Z}_q)$ is $(q^2-1)q$. The order of $PSL_2(\mathbb{Z}_q)$ is $(q^2-1)q/2$.*

Proof. To derive the order of the group $PGL_2(\mathbb{Z}_q)$, we use the Lagrange theorem which tells us that the order of the quotient group G/H is the order of G divided by the order of H, for the finite case. So, we have that the order of $GL_2(\mathbb{Z}_q)$ is $(q^2-1)(q^2-q)$ which corresponds to choose a vector $v \neq 0$ (and we have q^2-1 choices for that) and then choose another vector that is not a multiple of v (and we have q^2-q choices for that). On the other hand, we have that the order of $Z_n(\mathbb{Z}_q)$ is $q-1$. So, we conclude that the order of $PGL_2(\mathbb{Z}_q)$ is $(q^2-1)q$.

To derive the order of the group

$$PSL_2(\mathbb{Z}_q) = SL_2(\mathbb{Z}_q)/SZ_2(\mathbb{Z}_q)$$

we will use again Lagrange theorem. The order of $SL_2(\mathbb{Z}_q)$ can be computed noting that the homomorphism

$$\phi : GL_2(\mathbb{Z}_q) \to \mathbb{F}_q^*$$

that gives the determinant of the matrix is surjective and that its kernel is $SL_2(\mathbb{Z}_q)$; by the first isomorphism theorem[5] and Lagrange theorem we have that

$$|SL_2(\mathbb{Z}_q)| = \frac{|GL_2(\mathbb{Z}_q)|}{|\mathbb{F}_q^*|} = \frac{(q^2-1)(q^2-q)}{q-1}.$$

To compute the order of $SZ_2(\mathbb{Z}_q)$ note that this group is the intersection between $SL_2(\mathbb{Z}_p)$ and $Z_2(\mathbb{Z}_2)$. So, an element in $SZ_2(\mathbb{Z}_q)$ is a scalar matrix, since all the matrices in $Z_2(\mathbb{Z}_2)$ are scalar ones. Let A be a scalar matrix of size 2, then $A = \lambda I$ where I is the identity matrix and $\lambda \in \mathbb{Z}_p$. The determinant of A is λ^2. The equation $x^2 = 1$ in $\mathbb{Z}_p$ has only two solutions in $\mathbb{Z}_p$ (since p is prime) so the order of $SZ_2(\mathbb{Z}_q)$ is 2.

We conclude that

$$|PSL_2(\mathbb{Z}_2)| = \frac{|SL_2(\mathbb{Z}_q)|}{|SZ_2(\mathbb{Z}_q)|} = \frac{(q^2-1)q}{2}.$$

□

[5]Recall that if $\phi : A \to B$ is a surjective homomorphism, then $A/Ker(\phi) \cong B$. This result is called the first isomorphism theorem.

Construction of Ramanujan graphs

Algorithm 2 describes the construction of explicit Ramanujan graphs and it was first presented in [33].

Algorithm 2 Construction of Ramanujan Graphs

Choose p, q primes s.t. $p \neq q$ and $p, q = 1 \mod 4$. Choose $i \in \mathbb{Z}$ s.t. $i^2 = -1 \mod q$;

Compute the $p+1$ possible solutions $\alpha = (a_0, a_1, a_2, a_3)$ of $p = a_0^2 + a_1^2 + a_2^2 + a_3^2$ where $a_0 > 0$ is odd and a_1, a_2, a_3 are even;

Use these solutions $\alpha = (a_0, a_1, a_2, a_3)$ to construct the set

$$S = \left\{ \begin{bmatrix} a_0 + i.a_1 & a_2 + ia_3 \\ -a_2 + ia_3 & a_0 - ia_1 \end{bmatrix} : \alpha = (a_0, a_1, a_2, a_3) \right\} \in PGL_2(\mathbb{Z}_q);$$

if $\left(\frac{p}{q}\right) = 1$ **then**

Form the Cayley Graph $X(G, S)$ where $G = PSL_2(\mathbb{Z}_q)$;

else $\left(\frac{p}{q}\right) = -1$

Form the Cayley Graph $X(G, S)$ where $G = PGL_2(\mathbb{Z}_q)$;

end if

Here, $\left(\frac{p}{q}\right)$ denotes the Jacobi symbol, which is defined as

$$\left(\frac{p}{q}\right) = \begin{cases} 0 & \text{if } p = 0 \\ 1 & \text{if there is } x \text{ s.t. } p = x^2 \mod q \\ -1 & \text{otherwise} \end{cases}.$$

This algorithm gives us a $(p+1)$-regular graph of size $n = q(q^2-1)$, if $\left(\frac{p}{q}\right) = -1$, or $n = q(q^2-1)/2$, if $\left(\frac{p}{q}\right) = 1$.

In the case where $\left(\frac{p}{q}\right) = -1$, the graph obtained is also bipartite (between the subgroup $PSL_2(\mathbb{Z}_q)$ and its complement).

Theorem 14 ([33])**.** *The graph given by Algorithm 2 is a $(p+1)$-regular Ramanujan graph.*

Example 15. *Let us consider a Ramanujan graph built using Algorithm 2 with $p = 13$ and $q = 5$. Since the Jacobi symbol $\left(\frac{13}{5}\right)$ is equal to -1, this graph has $5(5^2 - 1) = 120$ vertices. It is a (14)-regular graph and it is bipartite. In Figure 2.2 we can see a representation of this graph, implemented using $Mathematica$.*

Regular LDPC codes

Regular LDPC codes are a subclass LDPC codes in which the graph associated with the code is a (a, d)-regular bipartite graph, where all the variables nodes have degree a and all the contraints nodes have degree d. They were first presented by Gallager along with a decoding algorithm [22]. Figure 2.3

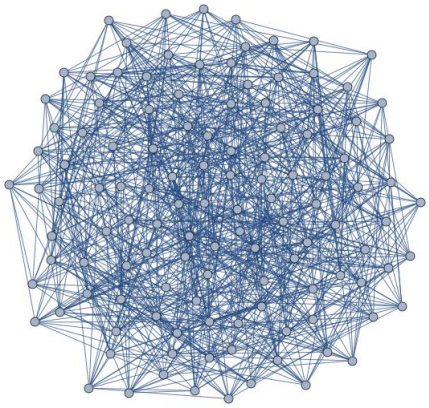

Figure 2.2: Ramanujan graph with $p = 13$ and $q = 5$.

shows an example of a regular LDPC where the variable nodes have degree 3 and the constraints nodes have degree 5.

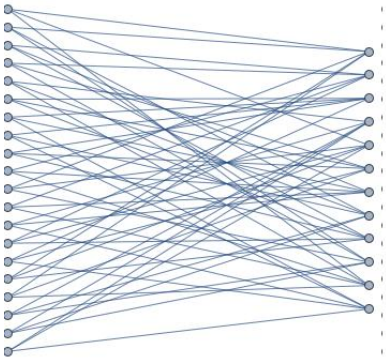

Figure 2.3: Regular LDPC.

The decoding algorithm for these codes is called *belief propagation* or *message passing algorithm* [50]. *Messages* are passed between the variables and constraints with the probability that a certain variable has a certain value. The value of the variable is sent to its adjacent constraints nodes, where this probability is computed (how this computation is done depends on the chosen algorithm). Then, this probability is sent to the variables, and its value is flipped depending on the message (probability) received. The new value of the variable is sent to its adjacent constraints nodes and the process is repeated. This process is done until all the probabilities are 1 or for a fixed number of times. In the end of this process we can evaluate if the word was well decoded or not (by checking if the resulting word belongs to the code). If the word was not well decoded, we assume that we are not able to decode this word.

We present a simplified version of the belief propagation decoding algorithm for regular graphs in Algorithm 3, tipically called the *hard decision decoding*. This simplified version is the result of discretizing the belief propagation algorithm, i.e., the messages passing from nodes are bits instead of probabilities. It is worth mentioning that this version is the one used in practice (or similar versions) since it leads to a increase of speed [34, 35, 37, 50]. It is also the version proposed by Gallager in his paper [22].

Unfortunately there is no result on the bound of how many errors can this algorithm (or any other for

Algorithm 3 A simpler version of belief propagation decoding algorithm

input: Corrupt word $y = w + e$;

output: Codeword w;

The variable nodes send their value to the adjacent contraint nodes;

for a fixed number of times **do**

Message from a constraint C to an adjacent variable v: The sum (modulo 2) of all the values of the adjacent variables, except v, of C is sent to v;

Message from a variable v to an adjacent C: If, for all the adjacent constraints , except for C, of v, the value computed is the same, then v takes this value and it is send to C. Otherwise v keeps its inicial value and this value is sent to C.

end for

regular LDPC codes) correct.

Expander codes

Expander codes are also a type of low density parity-check codes (LDPC codes). They were first presented by Sipser and Spielman in [54]. As we will see, this family of codes has nice properties and can be decoded very easily. Expander codes also have constant rate and distance unlike generic LDPC codes.

We will now define what an expander code is. The code is defined by a graph and by an error-correcting *inner* code.

Definition 16 (Expander Code [54])**.** *Let $a, d \in \mathbb{N}$ such that $a < d$. Let $B = (V, E)$ be a (a, d)-regular bipartite graph between a set of n vertices $\{v_1, \ldots, v_n\}$ (called variables) with degree a and a set of an/d vertices $\{C_1, \ldots, C_{an/d}\}$ (called check nodes or constraints) with degree d. Let $b(i, j)$ be a function that, for each constraint C_i, we have that $v_{b(i,1)}, ..., v_{b(i,d)}$ are the variables neighboring C_i. Let $\mathcal{C}$ be an error-correcting code of length d. The expander code $\mathfrak{E} = (B, \mathcal{C})$ is the code of length n defined as follows:*

$$\{(w_1, \ldots, w_n) \in \mathbb{Z}_2 : (w_{b(i,1)}, \ldots, w_{b(i,d)}) \in \mathcal{C}, \quad for \quad 1 \leq i \leq cn/d\}.$$

In other words, the variables neighboring each constraint must form a codeword of $\mathcal{C}$.

As an example, note that if $\mathcal{C}$ in the definition above is the code formed by all the words of even weight, then the code is a regular LDPC code.

There is a very intuitive and efficient decoding algorithm for expander codes [54]. In the algorithm, one just has to see which of the variables is connected to more unsatisfied constraints and flip the bit corresponding to this variable (if there are two or more variables in this condition, choose one at random). This process is repeated until we get a codeword of for a fixed number of times (if, after this number of times, we do not get a codeword, then the word is considered not decodable).

This algorithm is proved to converge for expander codes for which the expander graph as a expansion factor greater than $3a/4$ and for which the inner code is the code of all even weight words (Theorem 17).

This algorithm can also be used for regular LDPC codes that are not expander, although convergence is not guaranteed [37].

Note that the algorithm runs in polynomial time and it is extremely fast, as it only involves bit flipping operations.

Theorem 17 ([54]). *Let B be a $(a, d, \alpha, 3a/4)$ expander graph and $\mathcal{C}$ the code of all even weight words of length d. Given a corrupt codeword of the expander code $\mathfrak{E} = (B, \mathcal{C})$, the decoding algorithm can correct up to $n\alpha/2$ errors, where n is the length of $\mathfrak{E} = (B, \mathcal{C})$.*

It is not known explicit constructions for graphs with this expansion, since we can only guarantee expansion greater or equal to $a/2$, as it happens with Ramanujan graphs constructed using Algorithm 2 [28]. However, if we choose a random (a, d)-regular graph it is likely that this graph is a good expander graph [54, 58].

We can also build expander codes from Ramanujan graphs. The construction is exactly the same as in Definition 16, but the graph used is the edge-vertex incidence matrix of a d-regular Ramanujan graph (which gives us a $(2, d)$-regular graph with good expansion properties). For this case, Sipser and Spielman obtained lower bounds on the rate and minimum distance for the resultant code. They also proved a lower bound on the number of errors corrected using a simple variation of decoding algorithm for expander codes [54].

Proposition 18 ([54]). *Let $\mathfrak{E} = (B, \mathcal{C})$ be an expander code where B is the edge-vertex incidence matrix of a d-regular Ramanujan graph, with second largest eigenvalue λ, and $\mathcal{C}$ is a code with rate r and minimum relative distance[6] D. Then $\mathfrak{E} = (B, \mathcal{C})$ has rate $r' \geq 2r - 1$ and minimum relative distance $D' \geq \left(\frac{D - \lambda/d}{1 - \lambda/d}\right)^2$.*

In the following, $\widetilde{H}(x)$ denotes the *binary entropy function* and it is defined as follows:

$$\widetilde{H}(x) = -x \log_2 x - (1 - x) \log_2 (1 - x)$$

and $\widetilde{H}(0) = 0$.

Theorem 19 ([54]). *For all δ_0 such that $1 - 2\widetilde{H}(\delta_0) > 0$, there exists a family of expander codes, that can be constructed in polynomial time, with rate $1 - 2\widetilde{H}(\delta_0)$ and minimum relative distance arbitrarily close to δ_0^2 in which any $\alpha < \delta_0/48$ fraction of errors can be corrected by the decoding algorithm.*

Theorem 19 gives us a bound on the number of errors that Expander codes can correct. Unfortunately, the number of errors corrected is very low but this result can be improved using a slightly different decoding algorithm [60] (this decoding algorithm can decode up to four times more errors). Nevertheless, this result is very useful for cryptography since in code-based cryptography it is essencial to know exactly the number of errors that the code can correct in order to prove the correctness of the protocols. It seems that the only LDPC codes that have a decoding algorithm that garantees a fraction of errors corrected are LDPC codes based on expanders. Examples of such codes, their constructions and their decoding algorithms are presented in [24, 54, 55, 60].

[6] Recall that the minimum relative distance is the minimum distance of the code divided by its length.

Chapter 3

Public-key encryption scheme

In this chapter we will analyze public-key encryption schemes and some security notions regarding them. We will begin by defining what a public-key encryption scheme is. In Section 3.2, we continue our analysis by introducing two important security notions: (in)distinguishability, which is the ability of distinguish different ciphertexts, and malleability, which is the ability of creating new ciphertexts from a known ciphertext. Also, we give the relations between those security notions. Plaintext awareness is a security notion that is introduced in Section 3.3, where we also present an important result that links this notion with the others introduced before. Finally, we will end this chapter by giving a generic technique, the Fujisaki-Okamoto generic conversion, to achieve some of the security notions. This technique is relevant because it can be applied to any public-key encryption scheme, provided that some conditions are verified.

3.1 Basic notions

Public-key encryption allows for two parties to communicate privately. Unlike the symmetric setting, where the two parties need to share a secret-key, in public-key encryption schemes each of the parties creates a pair of public and secret keys. The public-key is public so that anyone can encrypt messages. The secret-key is kept private by the owner of the key so that no one else can decrypt messages. We begin this chapter with a simple definition. In this definition and throughout this thesis, when we say that some value is chosen randomly, we mean that it is chosen accordingly to a uniform distribution. When a vector or string is chosen randomly, we mean that each bit is chosen accordingly to a uniform distribution.

Definition 20 (Public-key cryptosystem)**.** *A public-key cryptosystem, or public-key encryption scheme,* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *is composed by three algorithms:*

Key creation *is a probabilistic polynomial-time algorithm* $\mathcal{K}$ *that receives as input* 1^k*, where* k *is a security parameter, and outputs a pair* $\mathcal{K}(1^k) = (pk, sk)$ *where* pk *is the public-key and* sk *is the secret-key (or private-key).*

Encryption *is a probabilistic polynomial-time algorithm* $\mathcal{E}$ *that receives as input a public-key* pk *and*

a message to encrypt $\mathbf{m}$, and a string of random coins r, and outputs a ciphertext $\mathcal{E}(pk, \mathbf{m}; k) = \mathbf{c}$. Informally, r represents the randomness of the cryptosystem. Throughout this thesis, if we omit r it means that these coins were chosen randomly.

***Decryption** is a deterministic polynomial-time algorithm $\mathcal{D}$ that receives as input a ciphertext $\mathbf{c}$ and a secret-key sk, and outputs a message $\mathcal{D}(sk, \mathbf{c}) = \mathbf{m}$, if $\mathbf{c}$ is a valid ciphertext; it halts otherwise.*

A Public-Key Cryptosystem (PKC) also has to have two more properties: i) *Correctness*, which means that a valid ciphertext should be uniquely decrypted to a message using a valid secret-key, i.e.,

$$\mathcal{D}[\mathcal{E}(pk, \mathbf{m}; r), sk] = \mathbf{m}$$

for every pair (pk, sk) generated by $\mathcal{K}$. And ii) *security* which means that it should be computationally hard to recover the message from its ciphertext without the secret-key.

Before we proceed, let us define hash function and cryptographic hash function.

Definition 21 (Hash function)**.** *A hash function h is a function that takes as input a string x of arbitrary size and maps it to a string of a fixed size. We will often call hash value of x to the output of a hash function given x as input.*

Definition 22 (Cryptographic hash function)**.** *A cryptographic hash function $\mathcal{H}$ is a hash function that has the following properties:*

- *It is hard to find collisions, i.e., it is hard to find two strings x and y such that $\mathcal{H}(x) = \mathcal{H}(y)$;*
- *It is hard to invert, i.e., given the hash h it is hard to find x such that $\mathcal{H}(x) = h$;*
- *It is a honest function, i.e., the output size is polynomially bounded by the size of the input.*

A cryptographic hash function is a hash function that has an unpredictable behaviour. Hash functions and, in particular cryptographic hash functions, have a lot of application in cryptography, as we will see.

In the rest of the chapter, we will focus our attention on security notions for public-key encryption schemes.

3.2 Security Notions

In this section, we will present two important security notions: indistinguishability and non-malleability.

First, we need to define the models and assumptions with which we are going to work with. We will consider two models of security: the standard model and the random oracle model.

Definition 23 (Standard model)**.** *The standard model is a model of security in which the adversary is limited by the computational time and space available.*

Definition 24 (Random oracle)**.** *The random oracle model is a model similar to the standard model but where we assume the existence of an oracle that is an ideal hash $\mathcal{H}$ (by an ideal hash we mean a hash function that returns truly uniformly random numbers for a new query and returns the same value for the same query).*

The random oracle simulates cryptographic hash functions and assumes that they have a completely random behaviour. This is a theoretical model which is very useful to establish security proofs. The problem is that random oracles are not realistic in the sense that there is no theoretical or empirical proof that such an oracle exists. In practice, the function used as a random oracle will not behave as a truly random oracle. Nevertheless, a proof in the random oracle gives us some guaranty and some hope that it could be used in practice. In other words: a proof in the random oracle is better than no proof at all.

3.2.1 Indistinguishability

A Public-Key Cryptosystem is indistinguishable if, given two messages and the encryption of one of them, an adversary is not able to tell which of the messages was encrypted. More precisely, given this scenario (where one of the messages is encrypted and it is asked to the adversary to tell which one was), the adversary has a negligible probability of guessing it right. There are three different notions of indistinguishability: *indistinguishability under chosen-plaintext attack* (IND-CPA), *indistinguishability under chosen-ciphertext attack* (IND-CCA) and *indistinguishability under adaptive chosen-ciphertext attack* (IND-CCA2).

In the following, let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a Public-Key Cryptosystem and let $\mathcal{A}$ be an probabilistic polynomial-time adversary and $\mathcal{C}$ a probabilistic polynomial-time challenger. Suppose that the challenger $\mathcal{C}$ generates a pair of keys $(pk, sk) = \mathcal{K}(1^k)$ where k is the security parameter of Π. $\mathcal{C}$ publishes pk and keeps sk as a secret.

IND-CPA

By an IND-CPA secure cryptosystem, we mean a cryptosystem for which the adversary $\mathcal{A}$ has a negligible probability to win the following game:

1. $\mathcal{A}$ can do an arbitrary polynomial number of encryptions;
2. At one point, $\mathcal{A}$ submits two different messages $\mathbf{m}_0$ and $\mathbf{m}_1$ to $\mathcal{C}$;
3. $\mathcal{C}$ chooses at random a bit $b \in \{0, 1\}$ and sends the challenge-ciphertext $\mathbf{c} = \mathcal{E}(pk, \mathbf{m}_b)$, the encryption of $\mathbf{m}_b$, to $\mathcal{A}$;
4. Again, $\mathcal{A}$ can do an arbitrary polynomial number of encryptions;
5. $\mathcal{A}$ outputs $\bar{b}$, a guess of b.

$\mathcal{A}$ wins the game if he can guess correctly the bit b.

INC-CCA

By an IND-CCA secure cryptosystem we mean a cryptosystem for which the adversary $\mathcal{A}$ has a negligible probability to win the following game:

1. $\mathcal{A}$ can do an arbitrary polynomial number of encryptions and calls to a decryption oracle;
2. At one point, $\mathcal{A}$ submits two different messages $\mathbf{m}_0$ and $\mathbf{m}_1$ to $\mathcal{C}$;
3. $\mathcal{C}$ chooses at random a bit $b \in \{0, 1\}$ and sends the challenge-ciphertext $\mathbf{c} = \mathcal{E}(pk, \mathbf{m}_b)$, the encryption of $\mathbf{m}_b$, to $\mathcal{A}$;
4. Again, $\mathcal{A}$ can do an arbitrary polynomial number of encryptions;
5. $\mathcal{A}$ outputs $\bar{b}$, a guess of b.

$\mathcal{A}$ wins the game if he can guess correctly the bit b.

IND-CCA2

By an IND-CCA2 secure cryptosystem we mean a cryptosystem for which the adversary $\mathcal{A}$ has a negligible probability to win the previous game, with the difference that in Step 4, the adversary $\mathcal{A}$ can also do a polynomial number of calls to the decription oracle, with the condition that he can not ask for the decryption of $\mathbf{c}$.

For a given public-key encryption scheme Π, we formally define the advantage of the adversary $\mathcal{A}$ for any of these games in the following way:

$$\mathtt{Adv}^{\Pi,\mathcal{A}}_{ind-aaa}(k) = 2.Pr(\bar{b} = b) - 1$$

where $aaa \in \{cpa, cca, cca2\}$. The advantage measures the adversary's chances of winning the game.

Obviously, the strongest property is the IND-CCA2 property: if a cryptosystem is IND-CCA2 then it is IND-CCA; and if it is IND-CCA then it is IND-CPA.

Note that if the encryption algorithm of public-key cryptosystem Π is deterministic, then Π is not IND-CPA secure: for an adversary to find the bit b given the challenge-ciphertext, it just has to encrypt $\mathbf{m}_0$ and $\mathbf{m}_1$ before hand and check which encryption is equal to the challenge-ciphertext.

3.2.2 Non-malleability

Another important security property is *non-malleability*. A cryptosystem that is non-malleable prevents an adversary from creating valid ciphertexts from another ciphertext. Malleability is generally considered an undesirable property for a cryptosystem as it threatens message integrity (although, in some cases, this is exactly what we are looking for, as in homomorphic encryption). As in indistinguishability, we can define 3 levels of non-malleable security: *non-malleability under chosen-plaintext attack* (NM-CPA), *non-malleability under chosen-ciphertext attack* (NM-CCA) and *non-malleability under adaptive chosen-ciphertext attack* (NM-CCA2). As in indistinguishability, these three levels differ only on the use of a decryption oracle by the adversary. Similarly to indistinguishability, non-malleability is defined by a game where the adversary has to build valid ciphertexts that correspond to valid messages given a valid ciphertext.

Once again, let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key cryptosystem and let $\mathcal{A}$ be an adversary and $\mathcal{C}$ a challenger. Suppose that the challenger $\mathcal{C}$ generates a pair of keys $(pk, sk) = \mathcal{K}(1^k)$ where k is the security parameter of Π. $\mathcal{C}$ publishes pk and keeps sk as a secret.

NM-CPA

By an NM-CPA secure cryptosystem, we mean a cryptosystem for which the adversary $\mathcal{A}$ has a negligible probability to win the following game:

1. $\mathcal{A}$ can do an arbitrary polynomial number encryptions;
2. $\mathcal{A}$ outputs a message space $\mathbf{M}$;
3. $\mathcal{C}$ chooses a message $\mathbf{m}$ from $\mathbf{M}$ at random and encrypts it. He sends $\mathbf{c} = \mathcal{E}(pk, \mathbf{m})$ to $\mathcal{A}$;
4. Given $\mathbf{c}$, $\mathcal{A}$ outputs $(R, \bar{\mathbf{c}})$, where R is a relation and $\bar{\mathbf{c}}$ is a vector $(\mathbf{c}_1, \ldots, \mathbf{c}_k)$ for some $k \in \mathbb{N}$. Here $\bar{\mathbf{c}}$ is a guess of ciphertexts such that their decryption is related to $\mathbf{m}$ by the relation R.

NM-CCA

By an NM-CCA secure cryptosystem we mean a cryptosystem for which the adversary $\mathcal{A}$ has a negligible probability to win the following game:

1. $\mathcal{A}$ can do an arbitrary polynomial number encryptions and calls to a decryption oracle;
2. $\mathcal{A}$ outputs a message space $\mathbf{M}$;
3. $\mathcal{C}$ chooses a message $\mathbf{m}$ from $\mathbf{M}$ at random and encrypts it. He sends $\mathbf{c} = \mathcal{E}(pk, \mathbf{m})$ to $\mathcal{A}$;
4. $\mathcal{A}$ outputs $(R, \bar{\mathbf{c}})$.

NM-CCA2

By an NM-CCA2 secure cryptosystem we mean a cryptosystem for which the adversary $\mathcal{A}$ has a negligible probability to win the previous game, with the difference that, before outputting $(R, \bar{\mathbf{c}})$, the adversary $\mathcal{A}$ is given access to a decryption oracle that she can query a polynomial number of times.

For a given public-key encryption scheme Π, we define the advantage of the adversary $\mathcal{A}$ for any of these games in the following way:

$$\mathtt{Adv}_{nm-aaa}^{\Pi,\mathcal{A}}(k) = Pr\left[\bar{\mathbf{m}} = \mathcal{D}(sk, \mathbf{c}) : \mathbf{c} \notin \bar{\mathbf{c}} \wedge \perp \notin \bar{\mathbf{m}} \wedge R(\mathbf{m}, \bar{\mathbf{m}})\right] - Pr\left[\bar{\mathbf{m}} = \mathcal{D}(sk, \mathbf{c}) : \mathbf{c} \notin \bar{\mathbf{c}} \wedge \perp \notin \bar{\mathbf{m}} \wedge R(\mathbf{m}', \bar{\mathbf{m}})\right]$$

where $aaa \in \{cpa, cca, cca2\}$ and $\mathbf{m}' \in M$ such that $\mathbf{m} \neq \mathbf{m}'$. To prevent the adversary from winning the game by giving trivial answers, $\bar{\mathbf{c}}$ can not have the challenge ciphertext and every ciphertext in $\bar{\mathbf{c}}$ must be a valid one a decrypt to a valid message (we prevent this by letting $\perp \notin \bar{\mathbf{m}}$, where $\perp$ is the result of decrypting an invalid ciphertext).

Again, note that the strongest property is that of NM-CCA2.

3.2.3 Relations between security notions

Indistinguishability and non-malleability may seem two very different concepts but, in fact, there are relations among them. These relations were proved by Belare *et al.* in [3]. We will present some of the most relevant results concerning these relations. In Figure 3.1 we can see the relations between all the security notions introduced so far.

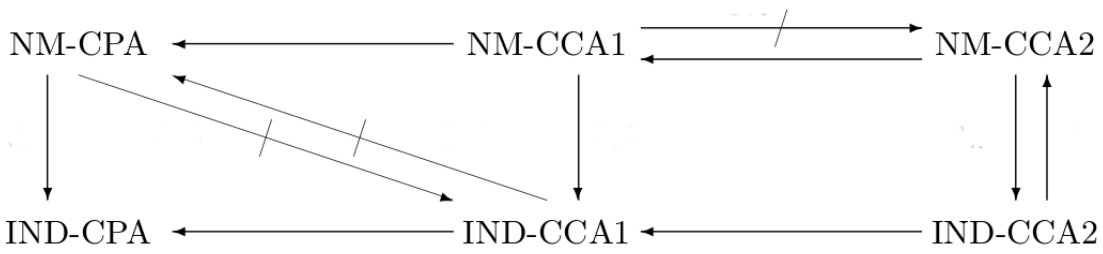

Figure 3.1: Scheme of security notions [3]. $A \rightarrow B$ means that A implies B. $A \not\rightarrow B$ means that there is, at least, one cryptosystem that has property A but does not have property B.

Theorem 25 ([3])**.** *Let* Π *be a public-key cryptosystem. If* Π *is NM-CPA secure then it is IND-CPA secure (the converse is not true).*

Theorem 26 ([3])**.** *Let* Π *be a public-key cryptosystem. If* Π *is NM-CCA secure then it is IND-CCA secure (the converse is not true).*

Theorem 27 ([3])**.** *Let* Π *be a public-key cryptosystem.*

$$\Pi \textit{ is IND-CCA2} \iff \Pi \textit{ is NM-CCA2 secure.}$$

In Theorems 25 and 26, when we say that the converse implication is not true, we mean that there is at least a cryptosystem that meets IND-CPA (IND-CCA, resp.) security that does not meet NM-CPA (NM-CCA, resp.) security. Informally, breaking non-malleability is usually harder than breaking indistinguishability, except if we are given a decryption oracle after the challenge ciphertext. In this case, although it may not seem intuitive, indistinguishability and non-malleability are the same thing.

3.3 Plaintext Awareness

For a given cryptosystem, an adversary can often produce valid ciphertexts for which she does not know the corresponding message. This can be done using, for example, the malleability of the cryptosystem. A cryptosystem is *plaintext aware* if an adversary cannot produce a valid ciphertext without knowing the corresponding message, i.e., the only way to create a valid ciphertext is by following the encryption process of a cryptosystem. Although very easy to explain intuitively, it is not trivial to formalize this definition. Formally, the definition of plaintext awareness was firstly presented in [4], and later it was refined in [3]. The definition that we will present here is the last one. Throughout this section, we will work exclusively in the random oracle model, since the definition of plaintext awareness that we will present only makes sense in this model of security.

Definition 28 (Plaintext Awareness)**.** *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme, $\mathcal{A}$ an adversary, $\mathcal{H}$ a cryptographic hash function and K an algorithm called the* knowledge extractor*. Consider the following game between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.*

- *$\mathcal{C}$ creates a pair of keys $(pk, sk) = \mathcal{K}(1^k)$. He publishes pk.*
- *$\mathcal{A}$ has access to the oracle $\mathcal{H}$ and to an encryption oracle. She can do a polynomial number of queries to each oracle. Eventually, she outputs (τ, η, y) where $\tau = \{(h_1, \mathcal{H}_1), \ldots, (h_{q_H}, \mathcal{H}_{q_h})\}$ is a list of queries $h_1, \ldots, h_{q_H}$ to the $\mathcal{H}$ oracle and the corresponding answers $\mathcal{H}_1, \ldots, \mathcal{H}_{q_H}$, $\eta = \{y_1, \ldots, y_{q_E}\}$ is a list of answers received by $\mathcal{A}$ of the queries done to the encryption oracle and $y \notin \eta$.*
- *$\mathcal{C}$ uses algorithm K, that receives as input (τ, η, y, pk), to try to find the decryption of y.*

We define

$$\mathtt{Adv}_{pa}^{K,\Pi,\mathcal{B}}(k) := Pr[K(\tau, \eta, y, pk) = \mathcal{D}(sk, y)].$$

We say that K is a $\lambda(k)$-knowledge extractor if K runs in polynomial time and $\mathtt{Adv}_{pa}^{K,\Pi,\mathcal{B}}(k) \geq \lambda(k)$.

If Π is IND-CPA secure and there exists a $\lambda(k)$-knowledge extractor K, where $1 - \lambda(k)$ is a negligible value, then we say that Π is plaintext aware (PA) secure.

Informally, a knowledge extractor receives a ciphertext and tries to find its decryption using only the queries made by $\mathcal{A}$ and respective answers to the hash oracle and answers from the encryption oracle.

Again, we can relate the concept of plaintext awareness with the security concepts of indistinguishability and non-malleability. In fact, plaintext awareness is the strongest security property. This means that if one can only build valid ciphertexts by following the encryption algorithm, then one can not distinguish ciphertexts.

Theorem 29 ([3])**.** *If a public-key cryptosystem is PA secure then it is IND-CCA2 secure.*

As a corollary, we can state that if a public-key cryptosystem is PA secure then it is NM-CCA2 secure.

3.4 Fujisaki-Okamoto IND-CCA2 generic conversion

Unfortunately, IND-CCA2 secure cryptosystems are not easy to construct. But, although most of the cryptosystems are not IND-CCA2 secure, there are some generic conversions, which we can apply to a cryptosystem that is not IND-CCA2 secure to obtain a new cryptosystem that is IND-CCA2 secure.

In this thesis we present one of these conversions, the Fujisaki-Okamoto simple IND-CCA2 conversion [21]. We believe that this is one of the simplest and most efficient conversions of which we are aware. The conversion aims to build a new IND-CCA2 secure cryptosystem $\bar{\Pi} = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$, in the random oracle model, from an IND-CPA secure cryptosystem $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where $\mathcal{K}$ is the key creation algorithm, $\mathcal{E}$ is the encryption algorithm and $\mathcal{D}$ is the decryption algorithm. The cryptosystem is presented in Algorithm 4. Before we take a look at the cryptosystem, we need a small definition.

Definition 30 (γ-uniform cryptosystem)**.** *Let Π be a public-key cryptosystem. For any message $\mathbf{m}$ and any ciphertext $\mathbf{c}$, we say that Π is γ-uniform if, given a message $\mathbf{m}$ and a ciphertext $\mathbf{c}$, the probability of guessing the encryption coins r at random such that $\mathcal{E}(pk, \mathbf{m}; r) = \mathbf{c}$ is less or equal than γ.*

Informally, this value γ measures the randomness that is used in the encryption of a message in the cryptosystem Π. If the encryption of a certain public-key encryption scheme is completely deterministic, then $\gamma = 1$. For this conversion to work, $\gamma < 1$. Recall that if the encryption of a cryptosystem Π is deterministic, then Π is not IND-CPA secure.

We now present the conversion in Algorithm 4. The idea is very simple: the message concatenated with a random string[1] will determine the random coins used in the encryption proccess. In other words, they will determine the randomness used in the encryption. When decrypting the ciphertext, one just has to verify if the encryption of the message using these coins is the received ciphertext; if this does not happen, then this ciphertext is an invalid one.

Algorithm 4 Generic conversion $\bar{\Pi} = (\bar{\mathcal{K}}, \bar{\mathcal{E}}, \bar{\mathcal{D}})$ of [21] based on a cryptosystem $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$

Parameters: Same as Π plus a cryptographic hash function $\mathcal{H}$.

Key Creation: Same as Π.

Encryption: We define the encryption

$$\mathbf{c} = \bar{\mathcal{E}}(pk, \mathbf{m}; s) = \mathcal{E}\left(pk, (\mathbf{m}|s); \mathcal{H}(\mathbf{m}|s)\right)$$

where s is a random string. The randomness of the encryption of Π is defined by the hash function $\mathcal{H}$.

Decryption: To decrypt a ciphertext $\mathbf{c}$, apply the decryption algorithm $\mathcal{D}$ of Π to get $(\mathbf{m}|s)$. Check if $\mathbf{c}$ is a valid ciphertext by testing if

$$\mathbf{c} = \mathcal{E}\Big(pk, \mathcal{D}(\mathbf{c}); \mathcal{H}\big(\mathcal{D}(\mathbf{c})\big)\Big).$$

Theorem 31 ([21])**.** *The conversion $\bar{\Pi}$ is IND-CCA2 secure in the random oracle model given that Π is IND-CPA secure and γ-uniform.*

The proof of this theorem is done by proving the existence of a knowledge extractor, such that

$$\mathtt{Adv}_{pa}^{K,\Pi,\mathcal{B}}(k) \geq \gamma,$$

and proving that Π is IND-CPA secure. Therefore, we can conclude that Π is PA secure and, by Theorem 29, Π is IND-CCA2 secure.

[1] We sometimes may call string to a vector, as it happens commonly in the literature. If x and y are two vectors (or strings) we will the denote their concatenation by $x|y$. We may also refer to this operation by padding.

Chapter 4

McEliece Cryptosystem

Since Diffie and Hellman introduced the concept of public-key cryptography [15], numerous public-key cryptosystems (PKC) have been proposed by the cryptographic community. These PKC are based on various conjectured computational hard problems, the most famous and used in practice being RSA, based on the integer factoring problem [47], and El Gamal PKC, based on the discrete logarithmic problem [17]. The situation could be dramatic if one was able to find a practical algorithm that could solve one of these problems, and thus break the cryptosystems. Unfortunately, no one can say if such an algorithm will be or will not be found.

In fact, Shor has already found a polynomial-time quantum algorithm that solves the integer factoring problem and the discrete logarithmic problem [51]. Since it is a quantum algorithm, it cannot yet be used in practice. But still, several alternative proposals are arising to replace RSA PKC and El Gamal PKC. These alternatives are based on computational hard problems that were recently presented and that are conjectured to be unsolvable in polynomial time even by quantum computers. Post-quantum cryptography deals with cryptographic classical algorithms based on these problems, thus it deals with cryptographic algorithms that are conjectured to be robust to quantum attacks, i.e., that are in the complexity class BQP. Code-based, lattice-based and multivariate polynomials problems are among these problems and they are being used for post-quantum cryptography right now [10].

Among these cryptosystems, we have the *McEliece cryptosystem* proposed by Robert McEliece [39] which is a code-based cryptosystem. This cryptosystem is one of the oldest public-key cryptosystems that is still secure and thus it is a very interesting candidate for post-quantum cryptography. Although there is no security reduction for the McEliece PKC, the extensive cryptoanalysis that it has been subject over the years has failed to find polynomial-time attacks. This fact has reinforced the conjecture that the McEliece cryptosystem is not efficiently breakable.

In this chapter we will analyze the McEliece PKC. We will begin by presenting, in Section 4.1 the cryptosystem along with its dual version, the Niederreiter PKC. In Section 4.2 we will analyze the security of the McEliece cryptosystem and we will explore some of its weaknesses. Section 4.3 is where we analyze the McEliece PKC computational complexity and efficiency, both of them very important aspects that must be taken into account for practical purposes. In Section 4.4, we will present some known vari-

ants that meet some important security properties, like IND-CPA and IND-CCA2. Finally, in Section 4.5, we will talk on the usage of LDPC codes in the McEliece PKC and its advantages and drawbacks.

4.1 The Cryptosystem

The McEliece PKC was presented by Robert McEliece in [39]. It remains unbreakable and it is also conjectured to be robust to quantum attacks. The McEliece cryptosystem is described in Algorithm 5.

Algorithm 5 McEliece public-key cryptosystem

Security parameters: $n, t \in \mathbb{N}$ with $t << n$.

Key Creation: Choose a Goppa Code $\mathcal{G} \subset \mathbb{Z}_2^n$: This has to be a k dimensional code of length n that can correct up to t errors. Generate the matrices G, S and P where:

G is a generator matrix of $\mathcal{G}$ of size $k \times n$;

S is a $k \times k$ non-singular matrix randomly chosen;

P is a $n \times n$ permutation matrix randomly chosen;

Compute $G^{pub} = SGP$, a $k \times n$ matrix.

Public key: (G^{pub}, t).

Private key: $(S, P, D_{\mathcal{G}})$ where $D_{\mathcal{G}}$ is a decoding algorithm for $\mathcal{G}$, tipically Patterson's algorithm.

Encryption: Choose $e \in \mathbb{Z}_2^n$ randomly such that e has weight t. To encrypt a message $\mathbf{m} \in \mathbb{Z}_2^k$, one must compute the ciphertext $\mathbf{c}$

$$\mathbf{c} = \mathbf{m}G^{pub} + e.$$

Decryption: To decrypt $\mathbf{c}$ one must compute $\mathbf{c}P^{-1}$, then apply $D_{\mathcal{G}}$ to the result and finally multiply by S^{-1} to get the message $\mathbf{m}$.

The security parameters of the McEliece cryptosystem are n and t that will define the code used to create the keys. The dimension of the code k cannot be chosen independently, since it is a function of n and t (Chapter 2). For the attacks to be inefficient, t must grow as n grows (we will discuss the choice of parameters later).

Proposition 32 tells us that the McEliece cryptosystem is correct.

Proposition 32. *Let $\mathbf{c}$ be the encryption of $\mathbf{m}$ using the McEliece cryptosystem. By applying the decryption algorithm of the McEliece to $\mathbf{c}$ we get $\mathbf{m}$.*

Proof. Let (G^{pub}, t) be a public key of the McEliece cryptosystem let $\mathbf{c} = \mathbf{m}G^{pub} + e$ be the encryption of a message $\mathbf{m}$ using the this cryptosystem. We will apply the decryption algorithm to $\mathbf{c}$. By multiplying $\mathbf{c}$ by P^{-1} we get

$$\mathbf{c}P^{-1} = (\mathbf{m}G^{pub} + e)P^{-1} = (\mathbf{m}SGPP^{-1} + eP^{-1}) = \mathbf{m}SG + eP^{-1}$$

and note that eP^{-1} is still a vector of weight t. So, applying the Patterson's decoding algorithm we can extract $\mathbf{m}S$ from $\mathbf{m}SG + eP^{-1}$. Then, we just need to multiply by S^{-1} to recover $\mathbf{m}$. □

A simple example of the Algorithm 5 is presented.

Example 33. *Suppose that Bob wants to send a message* $\mathbf{m}$ *to Alice. First Alice chooses a Goppa code* $\mathcal{G}$*; suppose that she chooses the code with the same generator matrix* G *as in Example 4:*

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

Next she chooses randomly a non-singular matrix S *of size 4 and a permutation matrix* P *of size 12, for example,*

$$S = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}; \quad P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

She computes $G^{pub} = SGP$ *and publishes the pair* $(G^{pub}, t = 2)$ *as her public key:*

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

Suppose that Bob wants to encrypt the message $\mathbf{m} = (1, 0, 1, 1)$ *using Alice's public key. He first computes*

$$\mathbf{m}G^{pub} = (1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1)$$

and then he chooses a random vector e *of weight* $t = 2$*, for example*

$$e = (0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0).$$

He computes

$$\mathbf{c} = \mathbf{m}G^{pub} + e = (1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1)$$

and sends $\mathbf{c}$ *to Alice.*

For Alice to decrypt the cipher $\mathbf{c}$ *using her private key, she first computes*

$$\mathbf{c}P^{-1} = (1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0)$$

then she applies Patterson's Algorithm (Algorithm 1 in page 9) to $\mathbf{c}P^{-1}$ *to get* $\mathbf{m}' = (0, 1, 1, 0)$ *and finally she computes* $\mathbf{m}'S^{-1}$ *to recover the message* $\mathbf{m}$*.*

4.1.1 The Niederreiter Cryptosystem

The *Niederreiter cryptosystem* was proposed by Harald Niederreiter in 1986, in [43]. It is very similar to the McEliece cryptosystem; in fact the Niederreiter cryptosystem is the dual version of the previous cryptosystem. The main difference is that the message space is the set of errors with weight t of a given code $\mathcal{C}$ and the message is encrypted as a syndrome.

Since this protocol only encrypts words of length n and weight t, we need a function to transform each word in $\mathbb{Z}_2^l$ in a word that can be encrypted. Let $\phi_{n,t} : \mathbb{Z}_2^l \to W_{n,t}$, where $W_{n,t} = \{e \in \mathbb{Z}_2^n : wt(e) = t\}$ and $l = \lfloor \log |W_{n,t}| \rfloor$, be that function. An explicit construction for this function can be found in [20] but we will treat it as a black-box.

Also, and since this cryptosystem is the dual version of the previous one, we will need a different decoding algorithm. We will call it the *syndrome decoding algorithm*.

Definition 34. *Let* $\mathcal{C}$ *be a code that corrects up to* t *errors and* H *its parity-check matrix. Let* e *be an error vector with* $wt(e) \leq t$*. A syndrome decoding algorithm* SD *is an algorithm such that given a syndrome* $s = eH^T$*, it outputs* e*.*

Proposition 35. *Let* $\mathcal{C}$ *be a code with parameters* $[n, k, d]$ *that corrects up to* t *errors. If there is a decoding algorithm* D *then there exists a syndrome decoding algorithm* SD*, and vice versa.*

Proof. Let G and H be the generating and parity-check matrix of $\mathcal{C}$, respectively. Let e be a error vector with $wt(e) \leq t$.

Let SD be a syndrome decoding algorithm of $\mathcal{C}$ such that for every $e \in W_{n,t}$ we have $SD(eH^T) = e$. Given a word $\mathbf{m}G+e$ we want to derive a decoding algorithm D such that $D(\mathbf{m}G+e) = \mathbf{m}G$. We can do that in the following way: multiply $\mathbf{m}G+e$ by H^T to get eH^T, then calculate $\mathbf{m}G = (\mathbf{m}G+e)+SD(eH^T)$.

Now, given a decoding algorithm D we will construct a syndrome decoding algorithm. Given a syndrome $s = eH^T$, first we need to find a vector e' such that $e'H^T = s$ (this can be done in polynomial-time, see Chapter 2). We know that $e' - e$ is a codeword of $\mathcal{C}$, since

$$(e' - e)H^T = e'H^T - eH^T = s - s = 0.$$

So $e' = w + e$ where w is a codeword of $\mathcal{C}$. Then we apply D to e' to get w and, finally, we can calculate $e = e' - w$. □

A description of the Niederreiter PKC is presented in Algorithm 6. Usually, the code used in the cryptosystem is a Goppa code.

Algorithm 6 Niederreiter public-key cryptosystem

Security parameters: $n, t \in \mathbb{N}$ with $t << n$.

Key Creation: Choose a k dimensional binary code $\mathcal{C}$ with length n that can correct up to t errors. Generate the matrices H, S and P where:

H is a parity-check matrix of $\mathcal{C}$ of size $(n-k) \times k$;

S is a $k \times k$ non-singular matrix randomly chosen;

P is a $n \times n$ permutation matrix randomly chosen.

Compute $H^{pub}_{k \times n} = SHP$.

Public key: (H^{pub}, t).

Private key: $(S, P, SD_{\mathcal{C}})$ where $SD_{\mathcal{C}}$ is a syndrom decoding algorithm for $\mathcal{C}$.

Encryption: To encrypt $\mathbf{m} \in \mathbb{Z}_2^l$, calculate $\phi_{n,t}(\mathbf{m}) = e$ where $\phi_{n,t}$ is a function that takes a message in $\mathbb{Z}_2^l$ and maps it to a vector of length n and weight t. Compute the ciphertext $\mathbf{c}$ as

$$\mathbf{c} = e(H^{pub})^T.$$

Decryption: To decrypt $\mathbf{c}$, compute $\mathbf{c}(S^{-1})^T = eP^T H^T$, then apply $SD_{\mathcal{C}}$ to get eP^T. Finally compute $e = eP^T(P^{-1})^T$ and recover $\mathbf{m}$ with $\mathbf{m} = \phi_{n,t}^{-1}(e)$.

The advantage of using this protocol instead of McEliece cryptosystem is that the keys are much smaller and the encryption is faster (although the decryption takes more time) [10]. Another advantage is that it can be used to produce a secure signature scheme [14].

4.2 Security

In this section, we will first talk about the assumptions on which the security of the McEliece cryptosystem is based. Then we will go through some of the most important attacks on the cryptosystem. Finally, we will analyze the security parameters of the McEliece and the corresponding security level and key size.

Security of the McEliece

We define the McEliece problem:

Problem 36 (McEliece Problem)**.** *Given a a public key pk and a ciphertext $\mathbf{c}$, that was obtained encrypting a message $\mathbf{m}$ using the McEliece cryptosystem with pk, recover the message $\mathbf{m}$.*

The McEliece cryptosystem, like other cryptosystems, relies on the exhaustive search problem (which is a computationally hard problem) since we can define the McEliece problem as follows: given a public key (G^{pub}, t) and a ciphertext $\mathbf{c} \in \mathbb{Z}_2^n$, find the message $\mathbf{m} \in \mathbb{Z}_2^k$ such that $wt(\mathbf{m}G^{pub} - \mathbf{c}) = t$.

Obviously, solving the McEliece problem is equivalent to breaking the McEliece cryptosystem. This can be done in two ways: recovering the message $\mathbf{m}$ directly or recovering the secret key from the public key.

First, we will talk about the security of the message. The security of the message in the McEliece cryptosystem relies on the hardness of the Computational Syndrome Vector since we can define the McEliece problem as follow: given an irreducible Goppa code $\mathcal{G}$ over $\mathbb{Z}_2$ and a ciphertext $\mathbf{c}$, find a vector e with weight $wt(e) \leq t$ such that $\mathbf{c} + e$ is a permuted codeword of $\mathcal{C}$. So it is obvious that if one solves the Computational Syndrome Vector problem, then one breaks the cryptosystem (apart from the permutation). But the problems are not known to equivalent. The reciprocal is not known to be true, since if one breaks the cryptosystem, one only solves the Computational Syndrome Vector problem for a certain class of codes, namely, in this case the Goppa codes. But we can choose the parameters of the code $\mathcal{G}$ in such way that the McEliece cryptosystem is as difficult as possible as the Computational Syndrome Problem. Equivalently, we can say that the McEliece cryptosystem relies on the *Learning with Parity Noise* (LPN) problem, a classical problem in learning theory.

Problem 37 (Learning with Parity Noise)**.** *Let s (the secret) be a word of length l. Given the pair*

$$(a, \langle s, a \rangle + e)$$

where $a \in \mathbb{Z}_2^l$, $\langle s, a \rangle$ denotes the usual inner product of s and a, and e is chosen using the Bernoulli distribution $\mathfrak{B}_\theta$ with parameter $\theta \in [0, 1/2]$, find s.

This problem is also conjectured to be hard, and it is often used to argue that the McEliece cryptosystem is hard to break. It is useful, in some circumstances, to use the hardness of this problem for proofs instead of the hardness of the Computational Syndrome Vector problem, as we will see later.

Another assumption on which the security of the McEliece PKC is based is the difficulty of distinguish a permuted Goppa code generating matrix from a randomly (uniformly) chosen binary matrix [14].

Problem 38 (Goppa Distinguisher Problem)**.** *Given a matrix $M_{k \times n}$, output 1 if M represents a generating matrix of a Goppa code and output 0 if M was chosen uniformly at random (i.e., each coordinate of M was chosen uniformly at random).*

Although there is no proof that this problem is NP-complete, it remains unsolvable for a long time and, thus, it is conjectured to be a hard problem. This does not happen with other classes of codes. Several attempts of using other codes in the McEliece, with the objective of reducing key size or increasing efficiency, have turned out failures since all of them almost immediatly reveal their structure and this can be used to break the cryptosystem [10, 41, 52]. But surprisingly, coding theorists have not found a way of distinguish Goppa codes generating matrices from random matrices.

We conclude that, in the McEliece cryptosystem, the security of the message is based on the Computational Syndrome Vector problem (or, sometimes, on the Learning with Parity Noise problem) and the security of the public key is based on the Goppa Distinguisher problem. Assuming that the Computational Syndrome Vector and the Goppa Distinguisher problems are hard, we can conjectured that the McEliece problem is hard.

Attacks on the McEliece

Several attacks have been found to the McEliece cryptosystem. While some of them can be avoided just by increasing the security parameters, others are more difficult to prevent since they are based on the structure of the cryptosystem. In the following, we will use the same notation as in Algorithm 5. We now present some of these attacks:

- ***Information set decoding attack:*** The information set decoding is an algorithm for decoding any arbitrary code. Although it can be applied to any code, it is not very efficient if the parameters of the McEliece cryptosystem are chosen properly. But still, this (or variants of it) is the best known attack. The idea of this attack is to pick a set of coordinates of the ciphertext not affected by the error vector and this set has to be sufficiently large such that the columns of the public generator matrix is an invertible matrix. That is, choose $K \subset \{1, \dots n\}$ with k elements (k is the dimension of the code). G_K^{pub}, $\mathbf{c}_K$ and e_K are formed by the columns of G^{pub}, $\mathbf{c}$ and e (respectively) in positions of elements in K. So if $e_K = 0$ and G_K^{pub} is invertible then

$$\mathbf{m} = \mathbf{c}_K (G_K^{pub})^{-1} = (\mathbf{c}_k + e_k)(G_K^{pub})^{-1}$$

  and the message $\mathbf{m}$ can be recovered. This process takes at least $\Omega(n^2)$ operations (to multiply matrices, we need at least to write them) and to be successful, it needs to be iterated $\binom{n}{k}/\binom{n-t}{k}$ times. Thus the complexity[1] of this attack is

$$\Omega\left(n^2 \frac{\binom{n}{k}}{\binom{n-t}{k}}\right).$$

  So, simply by enlarging the parameters, the attack becomes infeasible.

  The best known attacks to the McEliece PKC are variants and optimizations of this algorithm, like Stern's attack [56].

- **Quantum attack:** The best known quantum attack to the McEliece cryptosystem is the quantum set decoding attack, which is based on Grover's algorithm [9]. The main idea is to use Grover's algorithm to increase the speed of the information set decoding attack. This approach increases

[1] In this thesis, we present lower bounds on the complexity of the attacks (Big-Omega notation), instead of the usual upper-bounds used to analyze the complexity of algorithms (Big-O notation).

the attack speed considerably and its complexity is

$$\Omega\left(n^2\sqrt{\frac{\binom{n}{k}}{0.29\binom{n-t}{k}}}\right)$$

which means that the parameters have to be increase substantially if one desires the McEliece cryptosystem to be robust to quantum attacks.

- ***Attack knowing a part of the plaintext:*** Knowing part of the plaintext reduces dramaticaly the computational cost of a brute force attack to the McEliece cryptosystem [13]. Suppose that $\mathbf{m} = \mathbf{m}_l|\mathbf{m}_r$ and the adversary knows $\mathbf{m}_l$. We have that $k = k_l + k_r$ where k_l (k_r, resp.) is the length of $\mathbf{m}_l$ ($\mathbf{m}_r$, resp.). The difficulty of recovering the message $\mathbf{m}$ is then reduced to recovering a message from a McEliece cryptosystem with smaller parameters, namely n and k_l. Empirical evidences show that knowing about 23% of the plaintext is sufficient to recover the whole message for the original security parameters of the PKC (these parameters will be presented later).

- ***Malleability of McEliece PKC:*** Since the McEliece PKC is based on linear algebra, the cryptosystem is extremely vulnerable to malleability attacks [57]. Note that, even without knowing the plaintext, an adversary can modify the ciphertext in such a way that the plaintext will be also modified. If $\mathbf{c}$ is the encryption of $\mathbf{m}$, then the adversary chooses another message $\mathbf{m}'$ and he can create a valid ciphertext $\mathbf{c}'$ for $\mathbf{m} + \mathbf{m}'$ by computing

$$\mathbf{c} + \mathbf{m}'G^{pub} = \mathbf{m}G^{pub} + e + \mathbf{m}'G^{pub} = (\mathbf{m} + \mathbf{m}')G^{pub} + e = \mathbf{c}'.$$

  Because of that McEliece is not robust against chosen-ciphertext attacks, as we will see later.

  Malleability also allows related message attacks [12]. These types of attacks happen when the adversary encrypts two messages for which he knows there is a relation between them. For example, when the adversary encrypts the same message $\mathbf{m}$ twice we get $\mathbf{c}_1 = \mathbf{m}G^{pub} + e_1$ and $\mathbf{c}_2 = \mathbf{m}G^{pub} + e_2$. If an adversary has these ciphertexts, he gets

$$\mathbf{c}_1 + \mathbf{c}_2 = (\mathbf{m}G^{pub} + e_1) + (\mathbf{m}G^{pub} + e_2) = e_1 + e_2$$

  and if $t << n/2$, if a coordinate is equal to 0 in e_1+e_2, it is 0 in e_1 and in e_2, with very high probability. Given this information, it would be easier for an adversary to use, for example, an information set decoding attack to recover the message.

- **Weak keys:** Another important aspect that we have to take in account to have a secure McEliece PKC is the choice of the keys. The number of different polynomials of Goppa codes is very high, which does not allow an exhaustive search of a given key. However, it is proved that, for some choices of keys, the cryptosystem becomes breakable [32]. These keys are obtained by choosing polynomials of the form $g(x) = x^t$, where g is the polynomial that defines the Goppa code used as public-key. But the number of keys that allow this attack is too small, so we just need to avoid

choosing any of these keys.

- **High rate attack:** McEliece PKC using Goppa codes with a high rate (i.e., with very little redundancy) are also breakable [19]. By choosing a Goppa code with rate close to 1, there is an algorithm that is able to break the Goppa Distinguisher problem for those parameters. This means that an adversary is able to find the secret key associated with this code.

As we can see, there are several attacks to McEliece PKC, some of them pretty efficient but, as far as we know, we can always change the parameters or make small changes to the cryptosystem to prevent the attacks. We conclude that there are keys that should not be chosen in order to avoid the Weak keys and the High rate attacks. Later we will see how we can avoid malleability attacks on the McEliece PKC.

It is worth mentioning that the security of the Niederreiter cryptosystem is equivalent to the security of the McEliece [31]. They are both based on the same computational problems and it is proved that for a proper choice of parameters they achieve the same level of security.

Choice of parameters

We will now analyse the choice of parameters for the cryptosystem. Originally in [39], the author proposes to choose a Goppa code $\mathcal{G}$ of length $n = 1024 = 2^{10}$ and to fix $t = 50$, and thus the code has dimension $1024 - 50 \times 10 = 524$. The reason for the choice of these parameters is to try to make the problem of breaking the cryptosystem as hard as possible as the Computational Syndrome Vector problem. For this choice of parameters, we also make brute force attacks infeasible. The McEliece cryptosystem has become less secure for these parameters and throughout the years several new parameters have been proposed [1, 11, 42].

In Table 4.1 we can see the key parameters and security estimates for the McEliece cryptosystem. All the values are in bits, unless otherwise mentioned. The security of the message is an estimation of the time it takes for the information set decoding attack to be successful (in the Security column) and the time of the quantum information set decoding attack (PQ security column) to recover the message. We will use the complexity presented above to compute the estimations, although these are not the best known bounds on the complexity of the attacks (there are optimizations for them [56]). But still, this will give us an idea of the security that we can achieve for some choice of parameters. Note that the public key is assumed to be in the systematic form, which means that the first k columns represent the identity matrix. So we only need to store the last $n - k$ columns of the public-key and, thus, the size in bits of the public-key can be computed by the formula $(n - k)k$. Usually, this method reduces a lot the size of the public key [10]. In Table 4.1, the public key size is expressed in kilobytes (KB) and megabytes (MB).[2]

In the first row, the parameters $n = 1024$, $k = 524$ and $t = 50$ were the ones proposed originally by McEliece. The last row, the parameters $n = 6960$, $k = 5413$ and $t = 119$ are the last ones proposed as far as we know and they achieve, after some optimizations to the quantum attack, a post-quantum security of 2^{128} bits [1].

[2] Recall that a byte is equivalent to eight bits.

Parameters			Public key size (systematic)	Security	PQ security
n	k	t			
1024	524	50	32.8 KB	$\approx 2^{73}$	$\approx 2^{48}$
2048	1696	32	74.6 KB	$\approx 2^{105}$	$\approx 2^{64}$
4096	3616	40	217 KB	$\approx 2^{158}$	$\approx 2^{88}$
6960	5413	119	1.05 MB	$\approx 2^{289}$	$\approx 2^{158}$

Table 4.1: Parameters of the McEliece

As we can see, the public key of the McEliece cryptosystem is very big and this makes this cryptosystem not suitable for certain applications. At this moment, researchers are trying to find efficient and secure ways to reduce the key size.

4.3 Efficiency

McEliece PKC is quite efficient comparing to other cryptosystems. Its main drawback is its huge key size.

We present some results on the efficiency of the McEliece cryptosystem. We present results concerning encryption, decryption and key generation. Proofs for Propositions 39, 40 and 41 can be found in [18]. In the following, consider a McEliece PKC instance with a Goppa code $\mathcal{G} = (g, L)$ with parameters $[n, k, d]$, where g is defined in $\mathbb{F}_{2^m}$, that can correct up to t errors.

Proposition 39. *Generating a pair of keys for the McEliece cryptosystem takes*

$$\mathcal{O}\left(k^2 n + n^2 + t^3(n-k) + (n-k)^3\right)$$

operations.

Proposition 40. *The McEliece encryption takes $\mathcal{O}(k.n)$ operations.*

Proposition 41. *The McEliece decryption takes $\mathcal{O}\left(ntm^2\right)$ operations.*

Although the decryption complexity is acceptable for practical use, we note that implementing the decryption algorithm in hardware can be very difficult and inefficient has the length of the ciphertext grows [37]. While implementing basic operations (like sum or multiplication) in hardware is quite simple, some operations required for Patterson's algorithm, like computing roots of a polynomial, can be extremely painful to implement and it may come with a prohibitive time cost.

4.4 IND-CCA2 Variants

Note that the original McEliece cryptosystem is not IND-CCA2 secure. We can use the malleability of the cryptosystem to attack its indistinguishability.

In fact, the cryptosystem is not even IND-CPA secure. For the sake of completeness, we show that the McEliece cryptosystem is not IND-CPA nor IND-CCA2.

Proposition 42. *The original McEliece cryptosystem is not IND-CPA secure.*

Proof. To prove that the original McEliece is not IND-CPA secure, we will build an adversary $\mathcal{A}$ that has a non-negligible probability of winning the IND-CPA game.

$\mathcal{A}$ follows this simple strategy: when $\mathcal{A}$ receives the challenge-ciphertext $\mathbf{c}$, she computes

$$e_0 = \mathbf{c} + \mathbf{m}_0 G^{pub} \text{ and } e_1 = \mathbf{c} + \mathbf{m}_1 G^{pub}.$$

Note that

$$e_0 = \mathbf{c} + \mathbf{m}_0 G^{pub} = \mathbf{m}_b G^{pub} + e + \mathbf{m}_0 G^{pub} = (\mathbf{m}_b + \mathbf{m}_0) G^{pub} + e$$

and similarly $e_1 = (\mathbf{m}_b + \mathbf{m}_1) G^{pub} + e$.

If $b = 0$ then $e_0 = e$ and $e_1 = (\mathbf{m}_0 + \mathbf{m}_1) G^{pub} + e$. So e_0 has weight t or less (because $wt(e) \leq t$) and e_1 has weight greater or equal than t because $(\mathbf{m}_0 + \mathbf{m}_1) G^{pub}$ will be a codeword different from 0, since $\mathbf{m}_0 + \mathbf{m}_1 \neq 0$, and $wt\big[(\mathbf{m}_0 + \mathbf{m}_1) G^{pub}\big] \geq 2t + 1$ (the minimum distance of the code is $2t + 1$); so

$$wt(e_1) = wt\big[(\mathbf{m}_0 + \mathbf{m}_1) G^{pub} + e\big] > t.$$

Similarly we have that, if $b = 1$ then $wt(e_1) \leq t$ and $wt(e_0) > t$.

$\mathcal{A}$ just has to see which one of the e_i has weight equal or less to t (the number of errors corrected by the code) which is public. It outputs b where $wt(e_b) \leq t$ and $\mathcal{A}$ will surely guess the bit b right. We have that

$$\mathtt{Adv}_{ind-cpa}^{McEliece,\mathcal{A}}(n, t) = 2.Pr(\bar{b} = b) - 1 = 1$$

which means that $\mathcal{A}$ always wins the game. □

Corollary 43. *The original McEliece is not IND-CCA2.*

Proof. The proof is a direct consequence of the previous proposition. But still, we present the attack because the idea of this attack is different from the previous one and to emphasize that the malleability of the original McEliece is a problem.

Given the challenge ciphertext $\mathbf{c}$, which is the encryption of $\mathbf{m}_b$, to the adversary $\mathcal{A}$, she can follow this strategy: she chooses a message $\mathbf{m}'$, different from $\mathbf{m}_0$ and $\mathbf{m}_1$. She computes

$$\begin{aligned} \mathbf{c}' &= \mathbf{c} + \mathbf{m}' G^{pub} \\ &= (\mathbf{m}_b + \mathbf{m}') G^{pub} + e \end{aligned}$$

and asks the decryption oracle for the decryption of $\mathbf{c}'$. She will receive $\mathbf{m}_b + \mathbf{m}'$ and, since she knows $\mathbf{m}_0$,$\mathbf{m}_1$ and $\mathbf{m}'$ she can recover the bit b. We have that

$$\mathtt{Adv}_{ind-cca2}^{McEliece,\mathcal{A}}(n, t) = 1.$$

and this concludes the proof. □

IND-CPA variant in the standard model

Proposition 42 tells us that the McEliece cryptosystem unfortunately is not IND-CPA secure. To solve this problem we need to find a way such that the relations among plaintexts do not result in relations among ciphertexts. In [44], the authors proposed a very simple variant of the original McEliece PKC, the *randomized* McEliece PKC. The idea is just to add some randomness to a part of the message that will be encrypted, turning the ciphertext pseudorandom for an attacker. They do this by padding a random binary string s to the message $\mathbf{m}$ that we want to encrypt[3] and then apply the original McEliece, i.e. the ciphertext $\mathbf{c}$ is computed in the following way:

$$\mathbf{c} = (\mathbf{m}|s)\, G^{pub} + e$$

where G^{pub} is the public key of the McEliece PKC, e the random vector and s is a random string. The random string s should be large enough, so that the ciphertext looks like a pseudorandom value (it is proposed that s has 9 times the length of the message). It is proved that this simple variant achieves IND-CPA security in the standard model.

Theorem 44 ([44])**.** *The randomized McEliece is IND-CPA secure given that the LPN and the Goppa Distinguisher problems are computationally hard.*

IND-CCA2 variants in the random oracle model

Generic conversions to obtain an IND-CCA2 secure cryptosystem from another cryptosystem (that has not this property) exist [21, 46], however these conversions can add a lot of redundancy to encrypted messages when applied to the McEliece cryptosystem [10, 29]. So in [29], the authors presented conversions especially for the McEliece cryptosystem that are IND-CCA2 secure. In Algorithm 7 it is presented a conversion from [29]; for the sake of briefness and simplicity, we just present here one of the conversions (we will call it conversion α and it is presented in Algorithm 7), although the authors of [29] have presented three possible conversions. We just mentioned here that one of the conversions (not the one we present in this thesis) achieves a reduction on the redundancy of the ciphertext, comparing to the original version of the McEliece cryptosystem.

Recall that a pseudorandom number generator is an algorithm that generates pseudorandom numbers. The pseudorandom generator should have the property that it should be infeasible to distinguish between a number generated using the pseudorandom number generator and a number chosen at random. This algorithm receives a seed as input; the output (the pseudorandom number) depends on the seed. More information on pseudorandom number generators in [26, 40].

Theorem 45 ([29])**.** *Breaking the IND-CCA2 property of the conversion presented in Algorithm 7 in the random oracle model is as hard as breaking the original McEliece.*

Another variant that we will analyze is the one presented in [21] and presented before in Chapter 3. This is a generic conversion for any public-key cryptosystem, but we will present this conversion applied

[3] Recall that $x|y$ denotes the concatenation (or padding) of the strings x and y.

Algorithm 7 Conversion α of the McEliece PKC [29]

Security Parameters: Same as in the original McEliece, plus a pseudorandom number generator $\texttt{Gen}$ and a cryptographic hash function $\mathcal{H}$ with output in the set $\{1, \ldots, K\}$ where $K = \binom{n}{t}$. Recall that n is the length of the ciphertext and t the number of errors corrected by the code used in the cryptosystem. We will also use $\texttt{Conv}$ which is a bijection between the set $\{1, \ldots, K\}$ and the error vectors of weight t.

Key Creation: Same as in the original McEliece.

Encryption: To encrypt the message $\mathbf{m}$, first choose a random value s. Set $\bar{z} = \mathcal{H}(s|\mathbf{m})$ and

$$(y_1|y_2) = \texttt{Gen}(\bar{z}) + (s|\mathbf{m})\,.$$

Create the error vector $z = \texttt{Conv}(\bar{z})$. Return the ciphertext

$$\mathbf{c} = \left(y_1 G^{pub} + z\right)|y_2.$$

Decryption: To decrypt the ciphertext $\mathbf{c}$, take the first n bits of $\mathbf{c}$ and using a decoding algorithm recover y_1. Recover the error vector z used in the encryption by taking the first n bits of $\mathbf{c}$ and XOR them with $y_1 G^{pub}$. Compute $\bar{z} = \texttt{Conv}^{-1}(z)$ and

$$(s|\mathbf{m}) = \texttt{Gen}(\bar{z}) + (y_1|y_2).$$

Check if

$$\bar{z} = \mathcal{H}\left(s|\mathbf{m}\right);$$

if so, return $\mathbf{m}$; otherwise, reject $\mathbf{c}$.

to the McEliece cryptosystem (Algorithm 8 in page 38).

Algorithm 8 Generic conversion of [21] applied to the McEliece

Parameters: Same as in the original McEliece, plus a cryptographic hash function $\mathcal{H}$ (definition on page 18). Also, let $K = \binom{n}{t}$ where n is the length of the ciphertext and t the number of errors corrected by the code used in the cryptosystem.

Key Creation: Same as in the original McEliece.

Encryption: To encrypt a message $\mathbf{m}$, compute

$$\mathbf{c} = (\mathbf{m}|s)G^{pub} + \mathtt{Conv}[\mathcal{H}(\mathbf{m}|s)]$$

where s is a random string.

Decryption: To decrypt a ciphertext $\mathbf{c}$, apply the same decoding algorithm as in the original McEliece to get $\mathbf{m}|s$. Compute the error vector $e = \mathbf{c} + (\mathbf{m}|s)G^{pub}$. Check if $\mathbf{c}$ is a valid ciphertext by checking if e is equal to $\mathtt{Conv}\,[\mathcal{H}(\mathbf{m}|s)]$. If it is not, reject $\mathbf{c}$.

The generic conversion is proven to be IND-CCA2 secure if the original cryptosystem is IND-CPA secure. We know that the original McEliece is not IND-CPA secure so we would expect that this conversion would not work when applied to the original McEliece. However we can apply it to the randomized McEliece which we know that it is IND-CPA secure. The result is the same as applying it to the original McEliece, since the concatenation operation is associative and thus $(\mathbf{m}|s)|s' = \mathbf{m}|(s|s')$ where s and s' are random strings. To avoid attacks, namely the attack knowing a part of the plaintext, we just have to increase the size of the random string s. With this in mind, we can state the following result.

Theorem 46 ([21])**.** *Breaking the IND-CCA2 property of the conversion presented in Algorithm 8 in the random oracle model is as hard as breaking the IND-CPA of the randomized McEliece. Thus, it is as hard as breaking the LPN and the Goppa Distinguisher problems.*

Although very efficient, the previous conversions are only proved to be secure on the random oracle model. But we would prefer a contruction that is proven to be secure in the standard model, and recently a new variant of the McEliece was proposed [16].

IND-CCA2 variant in the standard model

First, we need to introduce the notion of k-repetition PKC. Roughly speaking, a k-repetition PKC is a PKC algorithm that is repeated k times on a single PKC.

Definition 47. *Consider a public-key cryptosystem $\Pi = (\mathcal{K}, \mathcal{E}, Dc)$. A k-repetition public-key cryptosystem $\Pi_k = (\mathcal{K}_k, \mathcal{E}_k, Dc_k)$ is composed by the three algorithms:*

***Key creation** is a probabilistic polynomial time (PPT) algorithm $\mathcal{K}_k$ that calls the key creation algorithm $\mathcal{K}$ of Π, k times in order to get $pk = (pk_1, \ldots, pk_k)$ and the corresponding secret keys $sk = (sk_1, \ldots, sk_k)$;*

Encryption *is a PPT algorithm $\mathcal{E}_k$ that calls the encryption algorithm $\mathcal{E}$ of Π, k times with each of the public keys $pk = (pk_1, \ldots, pk_k)$. It outputs $\mathcal{E}_k(pk, \mathbf{m})\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$;*

Decryption *is a deterministis polynomial time algorithm $\mathcal{D}_k$ that calls the decryption algorithm $\mathcal{D}$ of Π, k times in order to decrypt $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ with the secret key $sk = (sk_1, \ldots, sk_k)$. It halts if one of the c_i is an invalid ciphertext.*

With this in mind, given a PKC, we can construct a variant of the PKC that is IND-CCA2 from its k-repetition variant. Note that this is a generic construction, thus it can be applied to any PKC.

Before presenting the protocol we need to introduce the notion of signature and some notions of semantic security regarding signatures.

Definition 48 (Signature scheme)**.** *A signature scheme is defined by the following algorithms:*

Key Creation *is a PPT algorithm that takes as input a security parameter and outputs a pair (dsk, vk) where dsk is the signing and secret key and vk is the corresponding verification key;*

Signing *is a PPT algorithm that takes as input a tuple $(\mathbf{m}, dsk)$, a message to be sign and a signing key, an outputs a signature σ.*

Verifying *is a deterministic polynomial time algorithm that takes as input a tuple $(\sigma, \mathbf{m}, vk)$, a signature, a message and a verification key, and outputs 1 if σ is a valid signature for $\mathbf{m}$; outputs 0, otherwise.*

We will need on more notion of security concerning the signature scheme the we will use in our construction. By an *one-time strong unforgeable signature* scheme we mean a signature scheme for which any probabilistic polynomial time adversary $\mathcal{A}$ has a negligible probability of winning the following game:

1. The challenger creates a pair of signing and verification keys (dsk, vk). dsk is kept private, while vk is public;
2. $\mathcal{A}$ asks for the signature of a given message $\mathbf{m}'$;
3. $\mathcal{C}$ signs the message $\mathbf{m}'$ using dsk and sends it to $\mathcal{A}$;
4. At some point, $\mathcal{A}$ submits the pair $(\sigma, \mathbf{m})$ to $\mathcal{C}$, where σ is the signature of a message $\mathbf{m} \neq \mathbf{m}'$;
5. $\mathcal{C}$ verifies if the signature is valid.

The adversary $\mathcal{A}$ wins the game if he can correctly forge the signature.

We will need one more definition before we present the conversion.

Definition 49 (Verifiable PKC)**.** *A k-repetition public-key cryptosystem is verifiable if there exists an algorithm that receives as input a ciphertext $\mathbf{c}$, a public key $pk = (pk_1, \ldots, pk_k)$ and any sk_i, for $i \in \{1, \ldots, k\}$ and with the condition that, if this algorithm outputs 1, it means that the ciphertext can be decrypted to a valid message.*

Theorem 50 ([16])**.** *Assume that $\mathcal{SS}$ is a one-time strong unforgeable signature scheme and that Π_k is IND-CPA secure and verifiable. Then Π_{CCA2} is IND-CCA2 secure in the standard model.*

Algorithm 9 Π_{CCA2} encryption scheme from Π

Security parameters: A public-key cryptosystem and an one-time strong unforgeable signature scheme $\mathcal{SS}$;

Key Creation: Call $2k$ times the key creation algorithm of Π. It outputs $pk = (pk_1^0, pk_1^1, \ldots, pk_k^0, pk_k^1)$ and the corresponding secret key

$$sk = (sk_1^0, sk_1^1, \ldots, sk_k^0, sk_k^1).$$

Encryption: First, create a pair of signing and verifying (dsk, vk) keys of $\mathcal{SS}$. Encrypt $\mathbf{m}$ using the encryption algorithm $\mathcal{E}_k$ (of Π_k) with public key $(pk_1^{vk_1}, \ldots, pk_k^{vk_k})$. The resulting ciphertext is

$$\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_k).$$

Sign the message $\mathbf{c}$, with dsk, the resulting signature is σ. Send $\mathbf{c}' = (\mathbf{c}, vk, \sigma)$;

Decryption: Decrypt using Π_k with private key $sk = (sk_1^{vk_1}, \ldots, sk_k^{vk_k})$.

Another version that achieves IND-CCA2 security, based on correlated products, is presented in [16]. We will not present it here since the idea is very similar to the protocol presented in Algorithm 9. Although this version is proved to be IND-CCA2 secure in the standard model, we would like to highlight that this construction is purely theoretical as its use in practice is not possible. The reasons for that are the cost of encryption for a single message (a message is encrypted multiple times) and its large key size.

4.5 Using LDPC codes in the McEliece PKC

As we have said before, one of the main drawbacks of the McEliece cryptosystem is its large key size. Several solutions have been presented to overcome this problems. One possible solution is to use another family of codes instead of Goppa codes, one that has a matrix that can be stored in a more efficient way. For example, one can think of using an LDPC code in the cryptosystem: since LDPC codes are based on sparse matrices, maybe they have some structure that allows compression of the public-key. Also, LDPC codes scale very well, i.e, decoding algorithms are very fast in software and hardware implementations for large code lengths, which would improve the efficiency of the cryptosystem for large messages.

As far as we know, the only attempt to use LDPC codes in the McEliece cryptosystem is presented in [41]. Since the parity-check matrix is very sparse, we could keep it as the public-key (multiplied by another matrix, so that it is hidden[4]) which would reduce a lot the size of the public-key. However, there is an attack, where the secret-key can be recovered, based on the sparsity of the parity-check matrix [41]. Let us sketch the idea of the attack.

Suppose that H is the parity-check matrix of an LDPC code. The public key is $H^{pub} = RH$ where

[4]The parity-check matrix of an LDPC code needs to be hidden because this matrix is often used in the decryption algorithm. In most of the cases, knowing the parity-check matrix gives us the decryption algorithm.

R is a non-singular matrix. If R is sparse, then the public key RH is also sparse and this may allow the compression of the key. To encrypt a message $\mathbf{m}$, one just has to compute the generating matrix G associated with H^{pub} and compute $\mathbf{m}SG + e$ where S is a non-singular public matrix that is used to hide the message and e is a error vector of weight t. To decrypt, one just has to note that G and SG are equivalent codes. So one can use H to decode the corrupt codeword and take out the error vector.

For the cryptosystem to be secure, H^{pub} should not admit the decoding using any decoding technique for the LDPC codes and an adversary should not be able to retrieve H, or other H' equivalent to H and that admits decoding, from H^{pub}. However, the sparsity of the public key matrix can be used to recover some lines of the original parity-check matrix H which can be sufficient to allow the adversary to recover the secret-key H [41].

We could avoid this attack if we choose R such that it is a dense matrix, and thus H^{pub} would be dense. But then we could not reduce the public key size. So, the authors conclude that there is no advantage in using a LDPC code over a Goppa code in the McEliece cryptosystem.

We believe that there is a big advantage on the usage of LDPC codes in cryptography. LDPC codes are far more efficient than Goppa codes and we believe that this can be useful for hardware implementations. We also believe that there are methods to prevent attacks like this one and take advantage of the low complexity decoding algorithms of the LDPC codes in the McEliece cryptosystem. That is what we are going to study in the next chapter.

Chapter 5

A New Cryptosystem: *LDPC-based McEliece*

One of the most important features an asymmetric cryptosystem must fullfil is to be hardware implementable. Both RSA and elliptic curve cryptosystems (ECC) are highly demanding from the processing time point of view, making them very slow to use, even in dedicated hardware implementations [38]. Moreover, both are not quantum resistant [51], motivating the community to look for post-quantum cryptosystems [10]. One of the most promising candidates is the McEliece cryptosystem [39] that is based on error-correcting codes, which are usually well behaved in hardware implementations. This cryptosystem is the oldest one that is yet to be broken, by both classical and quantum algorithms.

Unfortunately, although much more efficient than RSA and ECC [18], the McEliece cryptosystem has still some downsides: the key size is much larger than both RSA and ECC, which is a problem for implementations (especially hardware implementations, where the memory available to store data is very limited); and the decryption algorithm adquires a bottleneck effect as the key size grows since the decoding algorithm of Goppa codes (the codes used in the original McEliece) takes to long as the key size increases [38]. On the other hand, we know that LDPC codes [50] scale very well in hardware implementations since their decoding algorithms are much simpler. But using LDPC codes in the McEliece may not be as secure as using Goppa codes [41].

The McEliece cryptosystem has yet another downside: it is not indistinguishable against adaptive chosen-ciphertext attacks (IND-CCA2). IND-CCA2 security is a crucial property for cryptosystems nowadays as this notion reveals how much information an adversary can get on the message from the ciphertext.

We propose a public-key encryption scheme that, given a message to encrypt, divides it into several parts and encrypts each part individually using the McEliece cryptosystem with a Goppa code (such that it can be implemented in parallel) and then encrypts the result using the McEliece cryptosystem with an LDPC code. We can say that we add a little extra security to the traditional McEliece by encrypting again using a LDPC-based McEliece cryptosystem. With this proposal, we aim to achieve the security provided by traditional McEliece and the scalability provided by LDPC codes. Since our approach differs

from the one in [41], the attack presented there does not work on our construction.

In Section 5.1, we present the proposal. In the next sections we begin to analyze the proposed cryptosystem: in Section 5.2 we prove its security, in Section 5.3 we analyze its efficiency and in Section 5.4 we compute the estimated security for some parameters and compare the results with the original McEliece. Finally, in Section 5.5 we present a variant of the cryptosystem that is IND-CCA2 secure together with a security proof.

5.1 Proposal

Before delving into the formal construction, we give the main idea behind the construction. As it is well known, Goppa codes do not scale well since Patterson's algorithm is relatively inefficient for large block length messages [38]. To overcome this difficulty, we propose a cryptosystem where a message $\mathbf{m}$ is divided into several small blocks. Each one of these blocks will be encoded using a Goppa code. The resulting codeword, which is the concatenation of codewords of the chosen Goppa code, will then be encoded using an LDPC code. Errors will be added to the resulting codeword such that these errors can be corrected by the LDPC code and by each of the small blocks, using Patterson's algorithm. Note that, since these blocks are chosen to be relatively small, Patterson's algorithm is expected to be quite efficient.

We now rigorously present the public-key encryption scheme. Suppose a system with Alice and Bob, where Bob wants to send an encrypted message $\mathbf{m}$ to Alice using her public key. Following the description with Figures 5.1, 5.3 and 5.2 might be helpful.

Key creation:

To create a pair of public and secret keys, Alice must choose a Goppa code $\mathcal{G}$, with generating matrix G of size $k \times n$, that is able to correct t errors. Then she must randomly choose a non-singular square matrix S, of size k, and a permutation matrix P, of size n. She computes $U_1 = SGP$, as in the original McEliece PKC. Now, consider

$$U_1^{\oplus \ell} = \begin{pmatrix} U_1 & 0 & \dots & 0 \\ 0 & U_1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & U_1 \end{pmatrix}$$

where the matrix U_1 appears ℓ times in the diagonal of $U_1^{\oplus \ell}$. Note that $U_1^{\oplus \ell}$ is a matrix of size $k\ell \times n\ell$.

Now, Alice chooses a binary LDPC code $\mathcal{L}$ with generating matrix $\hat{G}$, a $n\ell \times m$ matrix, that is able to correct δ errors. She then chooses randomly another non-singular matrix $\hat{S}$, of size $n\ell \times n\ell$, and another permutation matrix $\hat{P}$, of size $m \times m$. She computes $U_2 = \hat{S}\hat{G}\hat{P}$. The matrix $U = U_1^{\oplus \ell} U_2$ will be part of her public key.

To complete the key creation process, she constructs a list of vectors, that we will denote by A, in the following way: let $\{e_1, \dots, e_n\}$ be the canonical base of $\mathbb{Z}_2^n$. Let $\{e_1^j, \dots, e_\ell^j\}$ be a random permutation of

ℓ uniformly chosen vectors of the canonical base. Set

$$v_j = (e_1^j | \dots | e_\ell^j) U_2$$

and repeat this process c times to create the list $A = \{v_1, \dots, v_c\}$, where $c \in \mathbb{N}$ is a constant such that $c >> t$. Note that each vector of A is the concatenation of l error vectors of size n and weight 1 that were expanded by U_2. To encrypt a message, Bob will choose error vectors of this list to create the ciphertext, as we will see later.

The public key consists of (U, A, t, δ). The private key consists of $(S, G, P, \hat{S}, \hat{G}, \hat{P})$ and the corresponding decoding algorithms for $\mathcal{G}$ and $\mathcal{L}$, which we will denote by $D_\mathcal{G}$ and $D_\mathcal{L}$ respectively.

Encryption:

To encrypt, Bob randomly chooses t vectors from A and XOR them. The result of this operation is

$$E = v_{i_1} + \dots + v_{i_t}.$$

The encryption of a block of message $\mathbf{m} = (\mathbf{m}_1 | \dots | \mathbf{m}_\ell)$ is obtained by computing

$$\mathbf{c} = \mathbf{m}U + E + r_\delta$$

where r_δ is random vector of $\mathbb{Z}_2^n$ with weight δ chosen by Bob (see Figure 5.1 for a scheme of the encryption algorithm). The factor $E + r_\delta$ represents the errors that will be corrected by the decoding algorithms (see Figure 5.2).

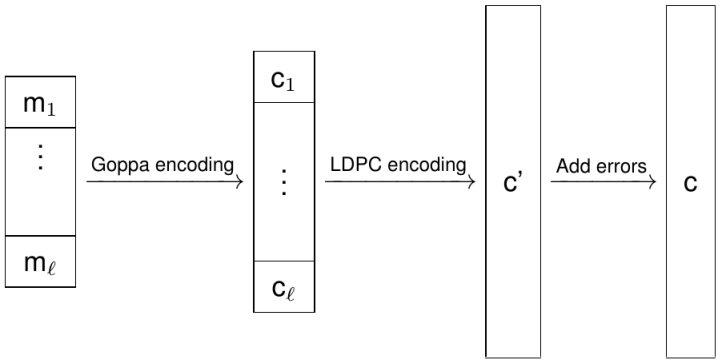

Figure 5.1: Encryption scheme

$$\begin{pmatrix} \\ r_\delta \\ \\ \end{pmatrix}^T + \begin{pmatrix} e_1^1 \\ \vdots \\ e_\ell^1 \end{pmatrix}^T U_2 + \dots + \begin{pmatrix} e_1^t \\ \vdots \\ e_\ell^t \end{pmatrix}^T U_2$$

Figure 5.2: Error construction scheme

Decryption

To decrypt a ciphertext $\mathbf{c}$, Alice first computes $\mathbf{c}\hat{P}^{-1}$ and applies the decoding algorithm $D_{\mathcal{L}}$ of the LDPC code to the result. Then, she multiplies the result by $\hat{S}^{-1}$ to obtain

$$\mathbf{m}U_1^{\oplus \ell} + \bigoplus_{j=1}^{t}(e_1^j|\ldots|e_\ell^j) = \mathbf{c}'.$$

Note that $\mathbf{c}'$ is just the concatenation of $\mathbf{c}_1, \ldots, \mathbf{c}_\ell$ where each $\mathbf{c}_i$ is the encryption of $\mathbf{m}_i$ by the original McEliece. Then, Alice can use the decryption algorithm of the original McEliece to get each $\mathbf{m}_i$ from $\mathbf{c}_i$ (i.e., multiply by P^{-1}, apply Patterson's algorithm to correct errors and multiply by S^{-1}). This task can be done in parallel, since each of the $\mathbf{m}_i$ is independent from the others. After she recovers all of the $\mathbf{m}_i$, she concatenates them to get $\mathbf{m}$. A scheme of the decryption algorithm is presented in Figure 5.3.

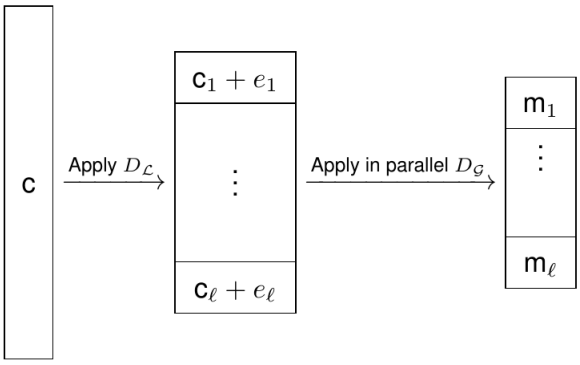

Figure 5.3: Decryption scheme: $D_{\mathcal{L}}$ and $D_{\mathcal{G}}$ denote the decoding algorithms of the LDPC and the Goppa codes, respectively.

The public-key cryptosystem is in Algorithm 10. We will call ***LDPC-based McEliece*** to this cryptosystem.

The reason for creating and publishing the list of errors A (that will be corrected by the Patterson's algorithm) is that we do not have to publish the matrices U_1 and U_2. This is useful for two reasons: we had a little extra security since we strongly believe that it is hard to factorize U into two matrices U_1 and U_2 and, if U_1 and U_2 were public, the cryptosystem would be just a composition of cryptosystems.

The list of errors A that is made public can not have all errors of the form $(e_i^1|\ldots|e_i^\ell)U_2$, since this list grows exponentially in the number of blocks (it takes time $\mathcal{O}(n^\ell)$ to produce this list). However, for practical purposes, we only need a constant number of them as long as this number is much greater than t (the number of errors corrected by the inner Goppa code). The owner of the public key can update this part of the key from time to time, to avoid that the same error patterns are used in different encryptions. We will see later that an adversary gains no information about the secret key from this list.

There are two particular cases that we would like to refer. The first extreme case is when $\ell = 1$. In this case, the protocol is just the composition of two McEliece-based cryptosystems, the original and a variant using an LDPC code. In this particular case, we could find the factorization of U up to permutation (which we want to keep secret) using the list A. Another particular case is when $l = |\mathbf{m}|$ (where $|\mathbf{m}|$ is the size of the message $\mathbf{m}$), which is not possible since there are no Goppa codes of size 1. For these

Algorithm 10 LDPC-based McEliece cryptosystem

Security parameters: $n, m, t, \delta, \ell, c \in \mathbb{N}$.

Key Creation: Choose a Goppa Code $\mathcal{G}$ able to correct t errors using a decoding algorithm $D_{\mathcal{G}}$. This code has length n and dimension k. Choose an LDPC code $\mathcal{L}$ able to correct δ errors using a decoding algorithm $D_{\mathcal{L}}$. This code must have length m and dimension $n\ell$. Generate the matrices G, S, P, $\hat{G}$, $\hat{S}$ and $\hat{P}$ where:

G is a generating matrix of $\mathcal{G}$ of size $k \times n$;

$\hat{G}$ is a generating matrix of $\mathcal{L}$ of size $n\ell \times m$;

S and $\hat{S}$ are non-singular matrices, of size k and size $n\ell$ (resp.), randomly chosen;

P and $\hat{P}$ are permutation matrices, of size n and size m (resp.), randomly chosen;

Compute $U = U_1^{\oplus \ell} U_2 = (SGP)^{\oplus \ell} \hat{S} \hat{G} \hat{P}$;

Construct the list $A = \{v_1, \ldots, v_c\}$, where each $v_j = (e_1^j | \ldots | e_\ell^j) U^2$ and $\{e_1^j, \ldots, e_\ell^j\}$ is a random permutation of ℓ uniformly chosen vectors of the canonical base of $\mathbb{Z}_2^{n\ell}$.

Public key: $(U, A = \{v_1, \ldots, v_c\}, t, \delta)$.

Private key: $(S, P, D_{\mathcal{G}}, \hat{S}, \hat{P}, D_{\mathcal{L}})$ where $D_{\mathcal{G}}$ is a decoding algorithm for $\mathcal{G}$ and $D_{\mathcal{L}}$ is a decoding algorithm for $\mathcal{L}$.

Encryption: Choose t vectors of A and sum them and let the result of this operation be E. Choose $r_\delta \in \mathbb{Z}_2^m$ randomly such that r_δ has weight δ. To encrypt a message $\mathbf{m} \in \mathbb{Z}_2^{k\ell}$ into a ciphertext $\mathbf{c}$, one must compute

$$\mathbf{c} = \mathbf{m}U + E + r_\delta.$$

Decryption: To decrypt $\mathbf{c}$, multiply $\mathbf{c}$ by $\hat{P}^{-1}$, apply $D_{\mathcal{L}}$ to take the error r_δ and multiply by $\hat{S}^{-1}$. We get $\mathbf{c}'$ after these operations. Then, in parallel, multiply each part of length n of $\mathbf{c}'$ by P^{-1}, apply the decoding algorithm $D_{\mathcal{G}}$ to take each error vector of weight t and multiply by S^{-1}. Concatenate in order each of the resulting $\mathbf{m}_i$ to get $\mathbf{m}$.

reasons, these cases should be avoided.

The choice of the LDPC code is a crucial point in order to guarantee the efficiency of our proposal. On the one hand, the LDPC chosen must have a low decoding complexity. On the other hand, the LDPC must correct any pattern of errors of weight δ, for sake of correctness of our cryptosystem. As we have mentioned in previous chapters and as far as we know, it seems that the only LDPC codes that have a decoding algorithm that guarantees a fraction of errors corrected are LDPC codes based on expanders [24, 54, 55, 60].

Another aspect that we have to take into account while choosing the LDPC code is the redundancy of the code. Redundancy of the ciphertext is a problem that can not be avoided in the McEliece cryptosystem as we have seen before. But still, we should avoid LDPC codes that had too much redundancy in the ciphertext. For better information ratio, one must choose LDPC codes with the highest information ratio possible, without compromising the security.

5.2 Security

In this section we present some arguments for the security of our cryptosystem. We were able to prove that to recover the message just by knowing the ciphertext and the public key is at least as hard as recovering the message if it was encrypted using the original McEliece cryptosystem and that to attack the public key of the LDPC-based McEliece cryptosystem is at least as hard as attacking the public key of the original McEliece. Also, we give some arguments on why we believe that it is hard to factorize the public key of the cryptosystem.

Given this, we present the following lemma which proves that it is hard to recover messages from their ciphertexts.

Proposition 51. *Breaking the security of the message of the LDPC-based McEliece cryptosystem is at least as hard as breaking the security of the message of the original McEliece.*

Proof. We can see the protocol as the composition of two different protocols, one of them being the original McEliece.

Consider the original McEliece PKC with public key (G, t). Let $\mathcal{A}$ be an algorithm that, given a ciphertext of the LDPC-based cryptosystem and its public key, returns the corresponding message. Let us also suppose that this algorithm runs in polynomial time. We will construct an algorithm $\mathcal{B}$ that breaks the original McEliece. Let $\mathbf{c}'$ be the encryption of a message using the original McEliece. $\mathcal{B}$ receives as input $\mathbf{c}'$ and the public key of the original McEliece (G, t) and does the following:

- Builds $\mathbf{d} = (\mathbf{c}'|\dots|\mathbf{c}')$ where $\mathbf{c}'$ is repeated ℓ times;
- Chooses a LDPC code $\mathcal{L}$, and following the key creation algorithm of the LDPC-based McEliece PKC, computes $U_2 = \hat{S}\hat{G}\hat{P}$ where $\hat{G}$ is a generating matrix of $\mathcal{L}$, and the list A;
- Computes $\mathbf{c} = \mathbf{d}U_2 + r_\delta$ where r_δ is a random vector of weight δ (the number of errors corrected by $\mathcal{L}$);

- Applies $\mathcal{A}$ with inputs $\mathbf{c}$ and the public key $(U = G^{\oplus \ell} U_2, A, t, \delta)$; $\mathcal{A}$ will output $\mathbf{m} = (\mathbf{m}'|\ldots|\mathbf{m}')$ where $\mathbf{m}'$ is repeated ℓ times;
- Output $\mathbf{m}'$.

Each of these tasks can be done in polynomial time, given that $\mathcal{A}$ runs in polynomial time. Thus, $\mathcal{B}$ runs in polynomial time and therefore we can break McEliece in polynomial time. □

We have proved that in our cryptosystem it is hard to recover the message from the ciphertext. So now we will talk about the security of the public key.

Proposition 52. *Breaking the security of the public key of the LDPC-based McEliece cryptosystem is at least as hard as breaking the security of the public key of the original McEliece.*

Proof. The reduction is similar to the previous one: consider the original McEliece PKC with public key (G, t). Let $\mathcal{A}$ be an algorithm that, given a public key of the LDPC-based cryptosystem, returns the corresponding secret key. Let us also suppose that this algorithm runs in polynomial time. We will construct an algorithm $\mathcal{B}$ that breaks the public key of the original McEliece. $\mathcal{B}$ receives as input (G, t) and does the following:

- Builds $U_1^{\oplus \ell} = G^{\oplus \ell}$;
- Chooses a LDPC code $\mathcal{L}$ that corrects δ, and, following the key creation algorithm of the LDPC-based McEliece PKC, computes $U_2 = \hat{S}\hat{G}\hat{P}$ where $\hat{G}$ is a generating matrix of $\mathcal{L}$. Also, creates the list A;
- Computes $U = U_1^{\oplus \ell} U_2$;
- Applies $\mathcal{A}$ with inputs (U, A, t, δ); $\mathcal{A}$ will output $(S, P, D_{\mathcal{G}}, \hat{S}, \hat{P}, D_{\mathcal{L}})$ where S and P are part of the factorization of G (according to the McEliece protocol) and $D_{\mathcal{G}}$ and $D_{\mathcal{L}}$ are the decoding algorithms for the Goppa code and for the LDPC code respectively used;
- Output $(S, P, D_{\mathcal{G}})$ which is the secret key of the McEliece cryptosystem with public key G.

Each of these tasks can be done in polynomial time, given that $\mathcal{A}$ runs in polynomial time. Thus, $\mathcal{B}$ runs in polynomial time and therefore we can break the public key of the McEliece in polynomial time. □

Although recovering the secret key is hard (given that breaking the McEliece cryptosystem is hard), it may be a problem if an adversary is able to factor the public key U. Recall that the public key is $(U, A = \{v_1, \ldots, v_c\}, t, \delta)$ where U is a matrix with $k\ell$ lines and m columns and it is the product of $U_1^{\oplus \ell}$ (a $k\ell \times n\ell$ matrix) by U_2 ($n\ell \times m$ matrix). Since $U_1^{\oplus \ell}$ represents the concatenation of ℓ original McEliece cryptosystem (which is conjectured to be indistinguishable from a uniformly chosen matrix), the real menace to the cryptosystem is if an adversary could find the which LDPC is being used. If it could do that, then breaking the cryptosystem would be equivalent to break the original McEliece for small parameters.

To find the LDPC code used in the cryptosystem, an adversary would need to factor the matrix U. Note that some information is leaked from the list A, namely the parameters of the LDPC code used. To find the parameters, an adversary can proceed in the following way: The adversary chooses a vector from the list A. Then it chooses another vector from A that is linearly independent from the first chosen vector. Again, it chooses another vector that is linearly independent from the first two chosen vectors. It does this until there is no linearly independent vector to choose. Most likely, it will get a set of $n\ell$ vectors linearly independent since each vector

$$v_j = u_j U$$

where $u_j = (e_1^j | \dots | e_\ell^j)$ is a vector of size $n\ell$. Let $\{v'_1, \dots, v'_{n\ell}\}$ be the list obtained from this procedure. The matrix

$$U'_2 = \begin{pmatrix} — & v'_1 & — \\ & \vdots & \\ — & v'_{n\ell} & — \end{pmatrix}$$

is a change of basis of the matrix U_2. With this, the adversary can get the value $n\ell$ and, thus, it gets the parameters of the LDPC code, which is a $n\ell$ dimensional code of length m. But note that it has no more information, particularly on the new basis, since it does not know which basis $\{u'_1, \dots, u'_{n\ell}\}$ was used to construct the list of vectors $\{v'_1, \dots, v'_{n\ell}\}$. So, it can not recover U_2 from its matrix U'_2.

Another threat to our cryptosystem is the attack presented in [41] and in the previous chapter. But note that this attack only works if the parity-check matrix of the LDPC code is made public (more precisely, the parity-check times another sparse matrix is made public) , which is not the case in our approach.

With this in mind, we conclude that every known attack on the cryptosystem does not seem to work. Even if an adversary was able to somehow find the matrix U_2 or the corresponding parity-check matrix, it would still have to find the number of blocks ℓ and to break ℓ McEliece ciphertexts. This lead us to strongly believe that it is hard to recover the secret key from the public key. Given this, we believe that our cryptosystem is secure.

Note that the cryptosystem is also robust against quantum attacks, otherwise we could quantumly attack the original McEliece as well. Also, since our protocol is based in the McEliece, we strongly believe that every attack on the McEliece (and every way to defend it from attacks) can be adapt to our cryptosystem.

If instead of using an LDPC code, we use a Goppa code (with larger parameters) we can prove that it is hard to factor the public key. To prove that, we assume that the Goppa Distinguisher problem is hard and, therefore, we cannot distinguish the public key from a uniformly chosen matrix. But using a Goppa code may compromise the efficiency of the cryptosystem.

5.3 Efficiency

The efficiency of the LDPC-based McEliece cryptosystem depends greatly on the choice of the LDPC used. In this section, we will analyze the efficiency of the proposed cryptosystem in terms of time complexity and circuit complexity. For the former, a unit of time is a binary operation; we will measure the time of an algorithm as the number of binary operations it takes. For the circuit complexity analysis, we will analyze what is the size and depth of a circuit, using AND, OR and NOT gates, that computes the algorithm.

Time complexity

In the following, we will use the same notation as in the previous section. Consider a cryptosystem using a k- dimensional Goppa code of length n defined by (g, L), where g is a polynomial with coefficients in $\mathbb{F}_{2^p}$, that can correct t errors, and using a $(n\ell)$-dimensional LDPC code of length m that can correct δ errors.

First, we need an auxiliary lemma which tells us the time complexity of multiplying two matrices.

Lemma 53. *Let $M_{n\times m}$ and $M'_{m\times p}$ be two matrices. The complexity of multiplying these two matrices is $\mathcal{O}(nmp)$.*

The proof of this lemma is done by evaluating the complexity of schoolbook matrix multiplication between these two matrices. This lemma is obviously necessary for the analysis of our algorithm since it uses a lot of matrix operations.

Proposition 54. *The time complexity of the encryption algorithm of our cryptosystem is*

$$\mathcal{O}\big(mk\ell + T^t_{rand} + \hat{T}^\delta_{rand}\big)$$

where T^t_{rand} and $\hat{T}^\delta_{rand}$ are the times that takes to choose t vectors from A and to create an error vector of weight δ, respectively.

Proof. The computational complexity of the encryption is just the complexity of multiplying a vector by a matrix plus the time of choosing t vectors from the list A, summing them and the time of choosing a random vector of weight δ.

Using the lemma above, we conclude that the time complexity of computing the ciphertext of a given message is

$$\mathcal{O}(mk\ell) + \mathcal{O}(T^t_{rand}) + \mathcal{O}(\hat{T}^\delta_{rand})$$

where T^t_{rand} is the time needed to choose t vectors from the list A and some them, and $\hat{T}^\delta_{rand}$ is the time needed to create an error vector of size m and weight δ. The factor $mk\ell$ corresponds to the multiplication of the message $\mathbf{m}$ (of size $k\ell$) by the public key U (of size $k\ell \times m$). □

Proposition 55. *The time complexity of the decryption algorithm of our cryptosystem is*

$$\mathcal{O}\left(\ell ntp^2 + T^{\mathcal{D}}_{LDPC} + km^2 + kn^2\ell^2\right)$$

where $T^{\mathcal{D}}_{LDPC}$ is the time it takes to decode a corrupt word using the LDPC code.

Proof. Here, the analysis is a bit more complex. We will assume that the owner of the secret key has previously computed and stored the inverse of the matrices P, S, $\hat{P}$ and $\hat{S}$. So the time that the decryption algorithm takes is the time of multiply matrices plus the time of decoding a Goppa code ℓ times and the time of decoding an LDPC code.

As before, let us assume that the Goppa code used in the cryptosystem is a k-dimensional code of length n with generator polynomial g defined in $\mathbb{F}_{2^p}$ and that the LDPC used is a $(n\ell)$-dimensional code of length m. The decryption algorithm of the original McEliece cryptosystem, as we have seen in Chapter 4 (Theorem 41), takes $\mathcal{O}(ntp^2)$ binary operations. Note that the Patterson's algorithm will be applied ℓ times. So, the complexity of the decryption algorithm is

$$\mathcal{O}\left(\ell ntp^2\right) + \mathcal{O}\big(T^{\mathcal{D}}_{LDPC} + km^2 + kn^2\ell^2\big)$$

where the factor km^2 corresponds to the multiplication by the matrix $\hat{P}$, the factor $kn^2\ell^2$ corresponds to the multiplication by the matrix $\hat{S}$ (by Lemma 53) and $T^{\mathcal{D}}_{LDPC}$ is the time it takes to decode a word using the LDPC code. □

Proposition 56. *The time complexity of the key creation algorithm of our cryptosystem is*

$$\mathcal{O}\left(k^2n + n^2 + t^3(n-k) + (n-k)^3 + T^{\mathcal{C}}_{LDPC} + km^2 + kn^2\ell^2 + ck\ell m\right)$$

where $T^{\mathcal{C}}_{LDPC}$ is the time it takes to create an LDPC code.

Proof. Note that, to create a pair of keys for the LDPC-based McEliece cryptosystem, we create a pair of keys for the original McEliece cryptosystem plus an LDPC code, matrices $\hat{P}$ and $\hat{S}$ and multiply them. This takes

$$\mathcal{O}\left(T^{\mathcal{K}}_{McEliece}\right) + \mathcal{O}\big(T^{\mathcal{C}}_{LDPC} + km^2 + kn^2\ell^2\big)$$

where $T^{\mathcal{K}}_{McEliece}$ is the time it takes to create a pair of keys for the original McEliece, $T^{\mathcal{C}}_{LDPC}$ is the time it takes to create a generating matrix of an LDPC code and the factors km^2 and $kn^2\ell^2$ correspond to the multiplication by the matrices $\hat{P}$ and $\hat{S}$ respectively. The time it takes to create the matrix $U_1^{\oplus\ell}$ is $\mathcal{O}(\ell nk)$, which is negligible for the complexity. We also need to create a list A of errors, which can be done in $\mathcal{O}(ck\ell m)$ operations. Hence, the time complexity is

$$\mathcal{O}\left(k^2n + n^2 + t^3(n-k) + (n-k)^3 + T^{\mathcal{C}}_{LDPC} + km^2 + kn^2\ell^2 + ck\ell m\right).$$

□

Recall that the list A can be renewed from time to time. The renovation of the list of errors A can be done with $\mathcal{O}(ck\ell m)$ operations.

The key creation algorithm is not as fast as the algorithms of encryption and decryption, but we would like to stress that the key creation is called only once (hopefully) while the others are called several times.

The main idea to retain is that LDPC codes usually have low time complexity decoding algorithms comparing to Goppa codes. This means that LDPC codes typically scale much better than Goppa codes [37]. Thus, the encryption and decryption for a large m should be faster than the original McEliece, using a code with the same length.

Another aspect we would like to highlight is the fact that this cryptosystem is extremely parallelizable. All the Goppa blocks can be implemented in parallel making the decryption algorithm much faster than if it was implemented sequentially. Also, the cryptosystem uses a lot of matrix multiplication in its encryption and decryption algorithms, and this operation that can also be parallelized. This fact makes the LDPC-based McEliece cryptosystem very useful for hardware implementations. To support this fact, we will analyze the circuit complexity of the cryptosystem.

Circuit complexity

Before we start, let us check some basic notions on circuit complexity. While computational complexity analyzes the space and time resources needed for a program to run, circuit complexity studies the size (the number of gates) of the circuit that computes a program using only AND, OR and NOT gates. Also, we evaluate these circuits in terms of depth, which can express the parallelization capacity of a program. Note that there is a different circuit for inputs of different size, so we analyze the size of the circuit in terms of the size of the input.

The following result gives us an upper-bound for the circuit-size complexity of a program from its time complexity.

Lemma 57 ([53])**.** *Let $T(n) : \mathbb{N} \to \mathbb{N}$. If A is a program that runs in time $\mathcal{O}(T(n))$ then A can be implemented in a circuit of size $\mathcal{O}\left(T(n)^2\right)$ where n is the size of the input.*

The proof of this theorem is done by implementing each instruction of A sequentially in a circuit. So, the theorem tells us that A can be implemented in a circuit of size (and depth) $\mathcal{O}\left(T(n)^2\right)$ but does not guarantee that there is not a circuit that computes A with a smaller size and depth, i.e., it only gives us an upper-bound for the circuit size (and does not give a clue about the lower-bound). A smaller circuit may possibly be constructed by optimizing the number of gates or parallelizing some instructions of A for smaller depth complexity.

Before we start the analysis, recall that our cryptosytem uses many matrix operations, so we prove the following lemma that gives us an upper-bound for the circuit complexity of multiplying two matrices.

Lemma 58. *Let $\mathcal{C}$ be a circuit that computes the product of two matrices $A_{n\times k}$ and $B_{k\times m}$. Then $\mathcal{C}$ has size $\mathcal{O}(nkm)$ and depth $\mathcal{O}(\log k)$.*

Proof. The input size is the size of A plus the size of B, thus the total size is $nk + mk$. Let $C_{n\times m} = AB$. The entries of C can be computed in the following way

$$c_{ij} = \bigvee_{l=1}^{k} (a_{il} \wedge b_{lj}).$$

So we need an AND gate to compute each term $a_{il} \wedge b_{lj}$ which gives us a total of nmk AND gates. Then, we need $k-1$ OR gates for each of the $\bigvee_l (a_{il} \wedge b_{lj})$ for fixed i and j. We can compute that using a binary tree, computing two by two and this gives us a circuit of depth $\log k$.

The circuit has size $\mathcal{O}(nmk + (k-1)) = \mathcal{O}(nmk)$ and depth $\mathcal{O}(\log k)$. □

Let us start by analyzing the circuit complexity of the encryption algorithm.

Proposition 59. *The encryption algorithm of the LDPC-based McEliece cryptosystem can be implemented with a circuit of size $\mathcal{O}(mk\ell)$ and depth $\mathcal{O}(\log k\ell)$ plus the circuit size and depth of choosing t random vectors of A and a random vector of size m and weight δ.*

Proof. For the encryption, we need to compute the circuit complexity of multiplying a matrix $U_{k\ell \times m}$ by a message vector $\mathbf{m}$ of size $k\ell$,

$$\mathbf{c}_{1\times m} = \mathbf{m}_{1\times k\ell} U_{k\ell\times m}.$$

Here, the size of the input is $mk\ell + k\ell$.

Using the Lemma 58, the circuit complexity of the encryption is $\mathcal{O}(mk\ell)$ plus the size of the circuit to choose the random vectors. The circuit-depth complexity is $\mathcal{O}(\log k\ell)$ plus the circuit-depth to choose the random vectors. □

The next theorem gives us the circuit complexity of decrypting a ciphertext using the decryption algorithm presented for our cryptosystem.

Proposition 60. *The decryption algorithm of the LDPC-based McEliece cryptosystem can be implemented with a circuit of size*

$$\mathcal{O}\left(m^2 + (n\ell)^2 + (ntp^2)^2 + k^2 + \mathcal{S}^{\mathcal{D}}_{LDPC}\right)$$

and depth

$$\mathcal{O}\left(\log(mn^2\ell k) + (ntp^2)^2 + \mathcal{DS}^{\mathcal{D}}_{LDPC}\right)$$

where $\mathcal{S}^{\mathcal{D}}_{LDPC}$ is the circuit size of the decoding algorithm of the chosen LDPC and $\mathcal{DS}^{\mathcal{D}}_{LDPC}$ its depth.

Proof. Once again, we assume that we have previously computed and stored the inverse of the matrices in the secret key. These are the tasks to perform in the decryption process:

- matrix multiplication $\hat{P}^{-1}$ which can be implemented with a circuit of size $\mathcal{O}(m^2)$ and depth $\mathcal{O}(\log m)$ by Lemma 58;
- decode a corrupted codeword of the LDPC code, which depends on the choice of the LDPC code. Let $\mathcal{S}^{\mathcal{D}}_{LDPC}$ be the circuit size of the decoding algorithm of the chosen LDPC and $\mathcal{DS}^{\mathcal{D}}_{LDPC}$ its depth;
- matrix multiplication $\hat{S}^{-1}$ which can be implemented in a circuit of size $\mathcal{O}(n^2\ell^2)$ and depth $\mathcal{O}(\log n\ell)$;
- matrix multiplication P^{-1}, repeated ℓ times, which can be done in parallel with a circuit of size $\mathcal{O}(n^2)$ and depth $\mathcal{O}(\log n)$;

- decode a corrupted codeword of the Goppa code. By Lemma 57, this task can be implemented with a circuit of size (and depth) $\mathcal{O}\left((ntp^2)^2\right)$;
- matrix multiplication S^{-1}, repeated ℓ times, which can be done in parallel with a circuit of size $\mathcal{O}(k^2)$ and depth $\mathcal{O}(\log k)$

This gives an upper-bound for the size of the circuit of

$$\mathcal{O}\left(m^2 + n^2\ell^2 + n^2 + (ntp^2)^2 + k^2\right) = \mathcal{O}\left(m^2 + (n\ell)^2 + (ntp^2)^2 + k^2\right)$$

and depth of

$$\mathcal{O}\left(\log m + \log n\ell + \log n + (ntp^2)^2 + \log k\right) = \mathcal{O}\left(\log(mn^2\ell k) + (ntp^2)^2\right).$$

□

Note that, as long as we choose the length of the Goppa code to be small (parameter n and, consequently, both parameters t and p), the cryptosystem has a very low depth complexity and therefore should be quite efficient.

5.4 Parameters

We will analyze the parameters to use and the security expected of the cryptosystem taking into account the classical and the quantum information set decoding attacks (see Chapter 4). Recall that, for a k dimensional code of size n that corrects t errors, the complexity of the classical information set decoding attack is

$$\Omega\left(n^2 \frac{\binom{n}{k}}{\binom{n-t}{k}}\right)$$

and the complexity of the quantum information set decoding attack is

$$\Omega\left(n^2 \sqrt{\frac{\binom{n}{k}}{0.29\binom{n-t}{k}}}\right).$$

If one wants to use the classical information set decoding to our cryptosystem, note that one needs to use it to decode the LDPC code and then to decode each of the Goppa blocks. So, the complexity of this attack to our cryptosystem is $\Omega\left(m^2\binom{m}{n\ell}/\binom{m-\delta}{n\ell}\right)$, which corresponds to decode the LDPC code, plus $\Omega\left(\ell n^2\binom{n}{k}/\binom{n-t}{k}\right)$, which corresponds to decode each of the Goppa blocks. A lower-bound in the total complexity of the attack is

$$\Omega\left(m^2 \frac{\binom{m}{n\ell}}{\binom{m-\delta}{n\ell}} + \ell n^2 \frac{\binom{n}{k}}{\binom{n-t}{k}}\right).$$

For a choice of parameters $m = 1024$, $n = 64$, $\ell = 12$, $k = 30$, $\delta = 70$, $t = 4$ the expected number of operations of an information set decoding attack is roughly 2^{172}. For these parameters, the key size (in the systematic form) is $(m - k\ell) \times k\ell \approx 30KB$ (kilobyte), which is much smaller than the McEliece

public key. For the same level of security, the key size of our cryptosystem is roughly ten times smaller than the McEliece key size.

Analogously, the complexity of the quantum set decoding attack for our cryptosystem is

$$\Omega\left(m^2\sqrt{\frac{\binom{m}{n\ell}}{0.29\binom{m-\delta}{n\ell}}}+\ell n^2\sqrt{\frac{\binom{n}{k}}{0.29\binom{n-t}{k}}}\right).$$

For the same choice of parameters above, the cryptosystem has a quantum security of approximately 2^{97} operations.

With this in mind, we can estimate parameters for our cryptosystem. Those estimates are presented in Table 5.1.

Parameters						Public-key size (systematic)	Security	PQ Security
m	n	ℓ	k	δ	t			
1024	64	12	30	70	4	$30\ KB$	$\approx 2^{172}$	$\approx 2^{97}$
1000	12	50	4	60	2	$20\ KB$	$\approx 2^{103}$	$\approx 2^{63}$

Table 5.1: Parameters for the LDPC-based McEliece cryptosystem

To maximize the security we need to enlarge the size of the Goppa blocks. This leads to an increase in the number of operations for the information set decoding attack. But, as we have seen in the previous section, if each of the Goppa blocks become too large, the cryptosystem is not so efficient. So, there is a trade off between security and efficiency: if one needs efficiency then it is better to choose smaller Goppa blocks; if one is looking for maximizing the security, one needs to choose larger Goppa blocks.

5.5 On the indistinguishability of the LDPC-based McEliece

Unfortunately, the LDPC-based McEliece is not IND-CCA2 secure, since it is extremely malleable (like the original McEliece).

Proposition 61. *The LDPC-based McEliece is not IND-CCA2 secure.*

Proof. To prove that the LDPC-based McEliece is not IND-CCA2 secure, we will build an adversary $\mathcal{A}$ that has a non-negligible probability of winning the IND-CCA2 game presented in page 20.

$\mathcal{A}$ follows this strategy: when $\mathcal{A}$ receives the challenge-ciphertext $\mathbf{c}=\mathbf{m}_bU+E+r_\delta$, she chooses a message $\mathbf{m}$, different from $\mathbf{m}_0$ and $\mathbf{m}_1$, and computes $\mathbf{c}'=\mathbf{m}U+\mathbf{c}$. Note that $\mathbf{c}'$ is a valid ciphertext. So, $\mathcal{A}$ can ask the decryption oracle for the decryption of $\mathbf{c}'$, which is $\mathbf{m}+\mathbf{m}_b$. Now, since $\mathcal{A}$ knows $\mathbf{m}$, $\mathbf{m}_0$ and $\mathbf{m}_1$, she can discover exactly what is the value of b. We conclude that the advantage of $\mathcal{A}$ in the IND-CCA2 game for the LDPC-based McEliece is maximal. □

In this section we will propose an IND-CCA2 variant of the LDPC-based McEliece cryptosystem. As we have highlighted before in this thesis, IND-CCA2 property is a crucial security property for cryptosystems nowadays (see Chapter 3).

For our IND-CCA2 variant of the cryptosystem we will need the concept of cryptographic hash function, previously introduced in Chapter 3. Recall that a cryptographic hash function takes an input of any size and maps it to a fixed size output such that it is difficult to invert and to find collisions.

To achieve the IND-CCA2 property, we will take out the malleability of the cryptosystem. The idea is very simple: the error that will be corrected by the LDPC (denoted by r_δ, just like in previous sections) that will be used in the construction of a ciphertext will be chosen accordingly to a cryptographic hash function. Given a ciphertext, if we do that, it becomes infeasible to create other valid ciphertexts from a valid one, which is the idea of the IND-CCA2 attack presented in the proof above.

We choose a cryptographic hash function $\mathcal{H}$ such that $\mathcal{H}$ maps inputs of arbitrary size to a fixed size output. Let $K = \binom{m}{\delta}$, where δ is the number of errors corrected by the LDPC code of length m in the cryptosystem[1] and let us choose $\mathcal{H}$ such that its output set is $\{1, \ldots, K\}$. We define a new function $\mathcal{H}^e$ using $\mathcal{H}$:

$$\mathcal{H}^e(x) := \texttt{Conv}\left[\mathcal{H}(x)\right]$$

where $\texttt{Conv}$ is a bijection between $\{1, \ldots, K\}$ and the error vectors of size m and weight δ. Note that, as m grows, K grows as well. So the probability of finding collisions for this function is exponentially low.

We now formally present the construction. When we encrypt a message $\mathbf{m}$, first we choose a random value s of fixed size and compute

$$r = \mathcal{H}^e(\mathbf{m}|s).$$

Also, instead of just encrypting $\mathbf{m}$, we will encrypt $\mathbf{m}|s$. This will be the error vector to be added to the ciphertext in the encryption algorithm and that will be corrected by the LDPC decoding algorithm. In the decryption, one just has to verify if $r_\delta = \mathcal{H}^e(\mathbf{m}|s)$. If so, the ciphertext was a valid one. If not, then the ciphertext is rejected as invalid.

Algorithm 11 briefly describes the IND-CCA2 version of the LDPC-based McEliece.

Algorithm 11 IND-CCA2 version of LDPC-based McEliece

Security parameters: Same as in LDPC-based McEliece cryptosystem plus a cryptographic hash functions $\mathcal{H}$.

Key Creation: Same as in LDPC-based McEliece.

Encryption: Choose a random string s and encrypt $\mathbf{m}|s$ using the encryption algorithm of the LDPC-based McEliece, except that the random vector of weight δ will not be chosen randomly, but accordingly to $\mathcal{H}$, i.e., $r_\delta = \mathcal{H}^e(\mathbf{m}|s)$. The ciphertext will be

$$\mathbf{c} = (\mathbf{m}|s)U + E + \mathcal{H}^e(\mathbf{m}|s).$$

Decryption: Apply the decryption algorithm of the LDPC-based McEliece to get $\mathbf{m}|s$. Compute the error vector $r_\delta = \mathbf{c} + (\mathbf{m}|s)U + E$. Check if $\mathbf{c}$ is a valid ciphertext by checking if r_δ is equal to $\mathcal{H}^e(\mathbf{m}|s)$. If it is not, reject $\mathbf{c}$.

---

[1] Once again, we see the importance of choosing an LDPC that corrects a constant fraction of errors.

This protocol is similar to the Fujisako-Okamoto generic conversion, presented in Chapter 3. The difference is that, in our conversion, only part of the randomization of the cryptosystem is determined by the cryptographic hash function while, in the generic conversion of Fujisako and Okamoto, all the randomization is determined by the cryptographic hash function.

We will now prove that this algorithm is indeed IND-CCA2 secure. First, we show that it is IND-CPA secure.

Theorem 62. *The LDPC-based McEliece cryptosystem is IND-CPA secure in the random oracle model given that the randomized McEliece is IND-CPA secure. Thus, it is IND-CPA secure given that the LNP and the Goppa Distinguisher problems are hard.*

Proof. Consider the following two games:

Game 1: IND-CPA game where $\mathbf{m}$ is encrypted in the following way: $(\mathbf{m}|s)U + E + r$, where r is a random vector of weight δ.

Game 2: IND-CPA game where $\mathbf{m}$ is encrypted in the following way: $(\mathbf{m}|s)U + E + \mathcal{H}^e(\mathbf{m}|s)$.

We will prove the theorem by proving a couple of claims.

Claim 1: From the adversary's point of view, Game 1 and Game 2 are computationally indistinguishable given that $\mathcal{H}$ is a cryptographic hash function.

Since the adversary does not know the random value s chosen in the encryption and $\mathcal{H}$ is a cryptographic hash function, the value $\mathcal{H}^e(\mathbf{m}|s)$ will look completely random for the adversary which means that it can not distinguish between $\mathcal{H}^e(\mathbf{m}|s)$ and a random vector of weight δ. So, it can not distinguish between these two games.

Claim 2: The advantage of an adversary $\mathcal{A}$ in Game 1 is negligible.

The proof is by reduction. Suppose that the advantage of an adversary $\mathcal{A}$ is non-negligible for Game 1, i.e.,

$$\mathtt{Adv}^{\mathcal{A}}_{Game\ 1}(\gamma) > \epsilon$$

where γ is the security parameters and ϵ is a non-negligible value. Then we will construct an adversary $\mathcal{B}$ that has non-negligible advantage of breaking the IND-CPA security of the randomized McEliece cryptosystem (which is a hard problem, given that both LPN and Goppa Distinguisher problems are hard, as we have seen in the previous chapter).

$\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives a public key of the randomized McEliece cryptosystem U_1 from the challenger. $\mathcal{B}$ chooses an LDPC capable of correcting δ errors, with generating matrix U_2 (which it will keep it to itself) and computes $U = U_1.U_2$. $\mathcal{B}$ simulates $\mathcal{A}$ and gives it U as a public key for the cryptosystem;
2. Eventually, $\mathcal{A}$ outputs two messages $\mathbf{m}_0, \mathbf{m}_1$. $\mathcal{B}$ outputs these two messages to the challenger;
3. $\mathcal{B}$ receives $\mathbf{c}$ from the challenger, where $\mathbf{c}$ is a randomized McEliece ciphertext of $\mathbf{m}_b$, with $b \in \{0, 1\}$;
4. $\mathcal{B}$ computes $\mathbf{c}' = \mathbf{c}U_2 + r$, where r is a random vector of weight δ. It submits $\mathbf{c}'$ to $\mathcal{A}$ as the challenge ciphertext;

5. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

The advantage of $\mathcal{B}$ of breaking the IND-CPA security of the randomized McEliece is

$$\mathtt{Adv}_{ind-cpa}^{RM,\mathcal{B}}(\gamma') = \mathtt{Adv}_{Game\ 1}^{\mathcal{A}}(\gamma) > \epsilon$$

where γ' is the security parameter of the randomized McEliece. Since this advantage is proved to be negligible, we conclude that the advantage of breaking the IND-CPA security of the IND-CCA2 version of the LDPC-based McEliece cryptosystem is also negligible. □

Theorem 63. *For any q_H, there exists a (λ)-extractor K for the IND-CCA2 version of the LDPC-based McEliece, where $1-\lambda$ is a negligible value.*

Proof. Let (U, A, t, δ) be the public key of the LDPC-based McEliece. We will denote the range of U by $Im(U)$. The specification of K is presented in Algorithm 12. Let $\tau = \{(h_1, \mathcal{H}_1), \ldots, (h_{q_H}, \mathcal{H}_{q_H})\}$ be a list of questions to the random oracle from an adversary and respective answers, $\eta = \{y_1, \ldots, y_{q_E}\}$ be a list of answers to queries to the encryption oracle by the same adversary and let $\mathbf{c} \notin \eta$ be a ciphertext.

Algorithm 12 Specification of knowledge extractor K

input: $(\tau, \eta, \mathbf{c}, U)$;

output: $\mathbf{m}$.

```
for q_H times do
    c' = c − H_i where (h_i, H_i) ∈ τ;
    if c' ∈ Im(U) then
        m = (h_i)_{kℓ} (the first kℓ bits of h_i);
        break;
    else
        m =⊥, where ⊥ is the empty string;
    end if
end for
return m.
```

Checking if the vector $\mathbf{c}'$ is in the range of the linear transformation U can be done in polynomial time (for example, by checking if the equation $\mathbf{c}' = Ux$ has solution or not).

K runs in time

$$t' = q_{\mathcal{H}}\Big(|\mathbf{c}| + T_{Im(U)} + k\ell\Big)$$

where $|\mathbf{c}|$ is the length of $\mathbf{c}$ and $T_{Im(U)}$ is the time it takes to evaluate if $\mathbf{c}'$ is in the image of U. So the time complexity is

$$\mathcal{O}\big(q_{\mathcal{H}}(|c| + T_{Im(U)} + k\ell)\big).$$

Now, let us analyze the success probability of K. Let $\mathcal{D}$ be the decryption algorithm of the IND-CCA2 version of the LDPC-based McEliece, sk its secret key and let ζ be an upper-bound for the probability of

finding a collision for the hash function $\mathcal{H}$, i.e.,

$$Pr\big[\text{find } x, y : \mathcal{H}(x) = \mathcal{H}(y)\big] \leq \zeta.$$

Let us define two events: $Fail$ and $AskH$. $Fail$ happens when $\mathbf{m} \neq \mathcal{D}(\mathbf{c}, sk)$. $AskH$ happens when there exists $(h_i, \mathcal{H}_i) \in \tau$ such that $\mathbf{c}' = c - \mathcal{H}_i$ is in the image of U. Then, we have

$$\begin{aligned} Pr(Fail) &= Pr(Fail|AskH).Pr(AskH) + Pr(Fail|\neg AskH).Pr(\neg AskH) \\ &\leq Pr(Fail|AskH) + Pr(Fail|\neg AskH) \\ &\leq \zeta + \frac{1}{\binom{m}{\delta}}. \end{aligned}$$

If $AskH$ is true, then the probability of K failing is equal to the probability of finding a collision for the hash function $\mathcal{H}$. So

$$Pr(Fail|AskH) \leq \zeta.$$

When $AskH$ is false, then K will output $\mathbf{m} = \perp$, i.e., K will consider $\mathbf{c}$ an invalid ciphertext. If $\mathbf{c}$ is a valid ciphertext, that means that the adversary $\mathcal{B}$ managed to create a valid ciphertext without querying the oracle $\mathcal{H}$. So,

$$Pr(Fail|\neg AskH) = \frac{1}{\binom{m}{\delta}}$$

which is the probability of guessing the right error vector for a given message $\mathbf{m}$ and a random value s. Hence, the probability of K not failing is

$$\begin{aligned} Pr(\neg Fail) &= 1 - Pr(Fail) \\ &\geq 1 - \left(\zeta + \frac{1}{\binom{m}{\delta}}\right) \end{aligned}$$

and we conclude that K is a λ- knowledge extractor, where

$$\lambda \geq 1 - \left(\zeta + \frac{1}{\binom{m}{\delta}}\right).$$

Since $\zeta + \frac{1}{\binom{m}{\delta}}$ is a negligible value, we have that $\lambda \approx 1$. □

We are now able to state that this version of the cryptosystem has the IND-CCA2 property, by plugging the two previous theorems and noting that PA-security implies IND-CCA2 security (see Chapter 3, Theorem 29).

Theorem 64. *The IND-CCA2 version of the LDPC-based McEliece is IND-CCA2 secure given that the randomized LDPC-based McEliece is IND-CPA secure.*

As a corollary, we have that this version is also non-malleable.

Corollary 65. *The IND-CCA2 version of the LDPC-Based McEliece cryptosystem is NM-CCA2 secure.*

Chapter 6

Conclusions

Post-quantum will soon be a reality in our everyday lives. So, it is essential to find new efficient and secure post-quantum protocols and that is what the cryptographic community is doing at the moment.

Currently, the McEliece cryptosystem is a serious candidate to become one of the most used cryptosystems in practice very soon. It has a simple and elegant structure and, although there is no formal security proof, nobody has a single clue on how to break it, either classically or quantumly. However the cryptosystem has still some downsides being the key size the most problematic.

We were able to construct a variant of the McEliece cryptosystem which we called the LDPC-based McEliece cryptosystem. This cryptosystem uses LDPC codes which have very efficient decoding algorithms and turn the decryption even faster and simpler to implement. Also, the composition of Goppa codes and LDPC codes used in the new construction dramatically decreases the key size comparing to the original McEliece. We believe that these two upgrades turn our cryptosystem suitable for software and hardware applications.

We were able to prove both the security of the ciphertext and the security of the public-key for this new construction. We were not able to prove that it is hard to factorize the public-key. However, we gave some arguments on why we think it is hard to find attacks that factorize the public-key.

Although our construction does not meet the IND-CCA2 property, we presented a IND-CCA2 variant of the LDPC-based McEliece cryptosystem. This variant is proved to be plaintext aware in the random oracle model, so it is proved to be IND-CCA2 secure in this model.

Directions of future work

Although we were not able to prove that factoring the public-key is a hard problem, we strongly believe that it is. Finding a proof for this fact will be left as future work.

In the IND-CCA2 variant of the LDPC-based McEliece cryptosystem, we see no reason to use a cryptographic hash function over an universal hash function. We believe that using a universal hash function is enough so that an adversary can not create valid ciphertexts without knowing the corresponding message. If we could prove IND-CCA2 security for a protocol using an universal hash function, we would be proving IND-CCA2 security in the standard model which is a much stronger result than security in the

random oracle model. This is because cryptographic hash functions do not yet exist in real life, while constructions for universal hash functions exist.

Also, finding methods to reduce even more the key size of the McEliece cryptosystem is of great importance if we are expecting to replace the current cryptographic protocols by quantum resistant protocols, in particular, by the McEliece cryptosystem.

Bibliography

[1] Initial recommendations of long-term secure post-quantum systems. Technical report, 2015.

[2] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[3] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology—CRYPTO'98*, pages 26–45. Springer, 1998.

[4] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology—EUROCRYPT'94*, pages 92–111. Springer, 1995.

[5] Elwin R. Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19(5):590–592, 1973.

[6] Elwin. R. Berlekamp. *Algebraic Coding Theory - Revised Edition*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2015.

[7] Elwyn R. Berlekamp, Robert J. McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[8] Daniel Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. *Post-Quantum Cryptography*, pages 31–46, 2008.

[9] Daniel J. Bernstein. Grover vs. McEliece. In *International Workshop on Post-Quantum Cryptography*, pages 73–80. Springer, 2010.

[10] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography*. Springer Berlin Heidelberg, 2009.

[11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In *International Workshop on Post-Quantum Cryptography*, pages 31–46. Springer, 2008.

[12] Thomas A. Berson. Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 1997.

[13] Anne Canteaut and Nicolas Sendrier. Cryptanalysis of the original McEliece cryptosystem. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 187–199. Springer, 1998.

[14] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 157–174. Springer, 2001.

[15] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[16] Nico Dottling, Rafael Dowsley, Jörn Muller-Quade, and Anderson C. A. Nascimento. A CCA2 secure variant of the McEliece cryptosystem. *IEEE Transactions on Information Theory*, 58(10):6672–6680, 2012.

[17] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[18] Daniela Engelbert, Raphael Overbeck, and Arthur Schmidt. A summary of McEliece-type cryptosystems and their security. *J. Mathematical Cryptology*, 1(2):151–199, 2007.

[19] Jean-Charles Faugere, Valérie Gauthier-Umana, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high-rate McEliece cryptosystems. *IEEE Transactions on Information Theory*, 59(10):6830–6844, 2013.

[20] Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 245–255. Springer, 1996.

[21] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In *International Workshop on Public Key Cryptography*, pages 53–68. Springer, 1999.

[22] Robert G. Gallager. *Low-density Parity-check Codes*. M.I.T. Press research monographs. M.I.T. Press, 1963.

[23] Valerii D. Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970.

[24] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005.

[25] Raymond Hill. *A First Course in Coding Theory*. Oxford Applied Linguistics. Clarendon Press, 1986.

[26] Jeffrey Hoffstein, Jill Catherine Pipher, Joseph H Silverman, and Joseph H Silverman. *An introduction to mathematical cryptography*, volume 1. Springer, 2008.

[27] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

[28] Nabil Kahale. On the second eigenvalue and linear expansion of regular graphs. In *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, pages 296–303. IEEE, 1992.

[29] Kazukuni Kobara and Hideki Imai. Semantically secure McEliece public-key cryptosystems - Conversions for McEliece PKC-. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, pages 19–35, London, UK, 2001. Springer-Verlag.

[30] Frank R. Kschischang, Brendan J. Frey, and H-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[31] Yuan Xing Li, Robert H. Deng, and Xin Mei Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273, 1994.

[32] Pierre Loidreau and Nicolas Sendrier. Weak keys in the McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, 47(3):1207–1211, 2001.

[33] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[34] Michael G. Luby, Michael Mitzenmacher, Amin Shokrollahi, and Daniel A. Spielman. Analysis of low density codes and improved designs using irregular graphs. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 249–258, New York, NY, USA, 1998. ACM.

[35] Michael G. Luby, Michael Mitzenmacher, Amin Shokrollahi, and Daniel A. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on Information Theory*, 47(2):585–598, September 2006.

[36] David J. C. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[37] David J. C. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, September 2006.

[38] Pedro Maat C Massolino, Paulo SLM Barreto, and Wilson V Ruggiero. Optimized and scalable co-processor for mceliece with binary goppa codes. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(3):45, 2015.

[39] Robert. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report*, 42(44):114–116, 1978.

[40] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[41] Chris Monico, Joachim Rosenthal, and Amin Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *Information Theory, 2000. Proceedings. IEEE International Symposium on*, page 215. IEEE, 2000.

[42] Robert Niebuhr, Mohammed Meziani, Stanislav Bulygin, and Johannes Buchmann. Selecting parameters for secure McEliece-based cryptosystems. *International Journal of Information Security*, 11(3):137–147, 2012.

[43] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory - Problemy Upravleniya i Teorii Informatsii*, 15(2):159–166, 1986.

[44] Ryo Nojima, Hideki Imai, Kazukuni Kobara, and Kirill Morozov. Semantic security for the McEliece cryptosystem without random oracles. *Designs, Codes and Cryptography*, 49(1):289–305, 2008.

[45] Nicholas Patterson. The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory*, 21(2):203–207, Mar 1975.

[46] David Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '00, pages 129–146, London, UK, 2000. Springer-Verlag.

[47] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[48] Christopher Roering. Coding theory-based cryptography: McEliece cryptosystems in Sage. 2013.

[49] Claude E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[50] Amin Shokrollahi. LDPC codes: An introduction. *Digital Fountain, Inc., Tech. Rep*, page 2, 2003.

[51] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[52] Vladimir M Sidelnikov and Sergey O Shestakov. On insecurity of cryptosystems based on generalized reed-solomon codes. *Discrete Mathematics and Applications*, 2(4):439–444, 1992.

[53] Michael Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[54] Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42:1710–1722, 1996.

[55] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.

[56] Jacques Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.

[57] Hung-Min Sun. Further cryptanalysis of the McEliece public-key cryptosystem. *IEEE Communications Letters*, 4(1):18–19, 2000.

[58] Salil P. Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[59] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, November 1997.

[60] Gillés Zémor. On expander codes. *IEEE Transactions on Information Theory*, 47(2):835–837, 2001.

Index