Two Approaches for Achieving Efficient Code-Based Cryptosystems

Rafael Misoczki

THÈSE DE DOCTORAT PRÉSENTÉE À
L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École Doctorale Informatique, Télécommunications et Électronique
(Paris)

Présentée par

Rafael Misoczki

Pour obtenir le grade de

DOCTEUR

Sujet de la thèse:

Two Approaches for Achieving Efficient Code-Based Cryptosystems

Soutenue le: 25 Novembre 2013

Devant la commission d'examen formée de:

Nicolas Sendrier	INRIA Paris-Rocquencourt	Directeur de thèse
Philippe Gaborit	Université de Limoges	Rapporteur
Gilles Zémor	Université Bordeaux 1	Rapporteur
Paulo Barreto	Universidade de São Paulo	Examinateur
Jean-Pierre Tillich	INRIA Paris-Rocquencourt	Examinateur
Jean-Claude Bajard	Université Pierre et Marie Curie	Examinateur

Rafael Misoczki

Two Approaches for Achieving Efficient Code-Based Cryptosystems

INRIA-Rocquencourt
Project SECRET
Domaine de Voluceau
78153 Le Chesnay

Dedicated to my mother, Rozires.

Acknowledgments

I would like to sincerely thank my supervisor, Nicolas Sendrier, for sharing his knowledge and for his incessant support and enthusiasm during these 3 years. Throughout this period, I could witness his remarkable passion for doing research in a perfectly balanced commitment between absorbing and sharing knowledge. I will definitely keep this as an example of what research is about. Obviously, I cannot forget to thank my friend and informal co-supervisor Paulo Barreto for his permanent remote support, for sharing his knowledge (even when conveyed in the Klingon language) and for having introduced me to the research world. Research has much to gain from so dedicated professors and researchers. In summary, I am aware how lucky am I to have had the opportunity to collaborate with two distinguished supervisors.

I also would like to thank my co-authors: Jean-Pierre, for countless fruitful discussions (but not for the countless attempts to pick up my dessert), Richard Lindner, Robert Niebuhr and Pierre-Louis Cayrel. Additionally, I would like to thank Pierre-Louis Cayrel for inviting me to visit him and his colleagues at the Center for Advanced Security Research Darmstadt – CASED, Darmstadt, Germany, in 2011, and for going skiing with me when I had hardly seen any snow before. Indeed, $-17\,^{\circ}\mathrm{C}$ is not an ideal temperature for whoever comes from the tropics.

I also would like to thank the people I met in the project SECRET, during these 3 years. A special thanks to: Christina, the keystone of our social events (we are happy that you are nearly back!), Valentin, for his sophisticated not-always-understood sense of humor (which I nevertheless particularly appreciate), and Christelle for the endless bureaucracy she helped me to deal with. And, of course: The *baby-foot* crowd: J-P, Dimitris, Gregory, Yann and Valentin. I really hope that, by this time, Dimitris has finally understood the *rateau*'s rule. An equally respectful thanks to: Anne, Pascale, Maria, Joëlle, Marion, Denise, Virginie, Anthony, Vincent, Matthieu, Baudoin, Mamdouh, Gaëtan, André, Ayoub, Céline, Stéphane Manuel, Stéphane Jacob and Assia.

During these 3 years, I also had the opportunity to travel to a few conferences/workshops. Besides the academic benefits, these travels introduced me to quite "interesting" moments. I want to thank the people who shared them with me. To Nicolas and J.-P, for discovering the peculiar Taiwanese cuisine, specially what I got at the Night Market. The absence of nouns is on purpose, since I have no idea what that was. To Dimitris, for our frozen time in Copenhagen, Denmark. To Audrey, for our apprehensive moments at Istanbul's ghetto, Turkey. Tear gas and water cannon are not lethal weapons, indeed.

At last, but far from being less important, I would like to thank my whole family. In special, my mother and my brother for their incessant support, encouragement and example of honesty and perseverance. You share a great deal of this achievement! To my dad for his attention even during his personal battle; now it is time to relax and go fishing! Special thanks to my girlfriend Tainá, who endured with love: our distance, our trips, the easy and hard moments, and especially for patiently listening my remarks on the importance of cryptography in her everyday situations. I also want to thank all my friends. In special, those ones whom I met in Paris and who made my stay in France even more awesome: Paulinho, Guilherme, Samir, Nuno and Victorien; and those who, even being in Brazil, continued to be very important for me, namely my step-brothers: Bruno and Carlos. Yes, now you both must prepare me a *churrasco*.

To conclude, this thesis is indeed the result of untold effort; but I am sure that it could not have been successfully done without the help of many people.

Thanks to all of you!

Contents

Glossary

Notation	Description
$\oplus$:	Bitwise exclusive-or operation.
$\mathbb{F}_q$:	Finite field of q elements.
v^T:	Transposition of a vector v.
M^T:	Transposition of a matrix M.
$\mathcal{C}^\perp$:	Dual code of a linear code $\mathcal{C}$.
$\mathcal{S}_n(0,t)$:	Sphere centered in zero of radius t in the Hamming space $\mathbb{F}_2^n$.
$u \xleftarrow{\$} U$:	Variable u is uniformly sampled at random from set U.
$(\ x \ \|\| \ y\ )$:	Concatenation of vector x followed by vector y.

Overview

Nowadays, cryptography is undoubtedly present everywhere. The necessity of secrecy, and more often privacy, is a crucial requirement for the modern world. Financial transactions, e-commerce and military applications are only a few examples that demonstrate the huge impact this research field has on our lives.

Due to its importance, many researchers have dedicated enormous amount of effort and time to propose and analyze efficient and secure cryptographic schemes. Number-theory cryptosystems, such as RSA and Elliptic Curves cryptosystems, are good candidates and widely deployed in practice. Although they provide a good trade-off between efficiency and security, they are not optimal in other features. For example, they are vulnerable to attacks mounted with the help of quantum computers.

This is not the case of cryptography based on coding-theory, where hard problems related to linear codes are exploited. Its security relies on the hardness of distinguishing a public code from random and on correcting errors in a seemingly random code. No quantum (nor classical) polynomial algorithm able to solve the decoding problem is known. The distinguishing problem has a more complex assessment and strongly depends on the choice of the code-family.

Besides its post-quantum resistance, code-based cryptography is several times faster than its number-theory counterparts. Nonetheless, it is not widely deployed in practice. Mostly due to its important drawback: huge key sizes. In this thesis, we propose two different approaches to address this issue.

The first one uses algebraic codes, presenting a way to construct Goppa codes that admit compact representation. These are the p-adic Goppa codes. We show how to construct these codes to instantiate public-key encryption schemes, how to extend this approach to a signature scheme and, finally, how to generalize the approach to codes defined over characteristic $p \geq 2$. In summary, we managed to produce very compact keys based on the reputable family of Goppa codes.

Although efficient, p-adic Goppa codes have a non-desirable property: strong algebraic structure. This leads to our second approach, using LDPC codes of increased density, or simply MDPC codes. These are graph-based codes, which are free of algebraic structure. It is quite reasonable to assume that MPDC codes are only distinguishable by finding their dual low-weight codewords, a largely studied coding-theory problem. This is an important advantage not only in comparison to all previous compact-keys McEliece-like variants but also regarding the classical McEliece based on binary Goppa codes. Here, compact keys are obtained by using a quasi-cyclic structure.

Résumé

Aujourd'hui, la cryptographie est indubitablement présente partout. La nécessité de confidentialité est une condition essentielle pour le monde moderne. Les transactions financières, les applications de commerce électronique et militaires ne sont que quelques exemples qui démontrent l'impact énorme que ce domaine de recherche a sur nos vies.

En raison de son importance, de nombreux chercheurs ont consacré énormément de temps et d'efforts pour proposer et analyser des systèmes cryptographiques qui sont à la fois efficaces et sûrs. Les cryptosystèmes basés sur la théorie des nombres, tels que RSA et les courbes elliptiques, sont de bons candidats. Ils sont largement déployés dans la pratique, puisqu'ils offrent un bon compromis entre efficacité et sécurité. Néanmoins, ils ne sont pas optimaux pour d'autres caractéristiques. Par exemple, ils sont vulnérables aux attaques menées à l'aide d'ordinateurs quantiques.

Ce n'est pas le cas de la cryptographie basée sur des problèmes difficiles liés aux codes linéaires. La sécurité ici repose sur la difficulté à distinguer un code public d'un code aléatoire et sur la correction des erreurs dans un code apparemment aléatoire. Il n'existe pas d'algorithme quantique (ni classique) capable de résoudre le problème de décodage en temps polynomial. Par contre, le problème de la distinction de codes a une complexité fortement dépendant du choix de la famille de codes.

En plus de sa résistance post-quantique, la cryptographie basée sur les codes est largement plus rapide que ses homologues basés sur la théorie des nombres. Néanmoins, elle est très peu déployée dans la pratique, son inconvénient majeur étant: des tailles de clés énormes. Dans cette thèse, nous proposons deux approches différentes pour résoudre ce problème.

Le premier utilise des codes algébriques, présentant un moyen de construire des codes de Goppa qui admettent une représentation compacte. Ce sont les Codes de Goppa p-adiques. Nous montrons comment construire ces codes pour instancier des systèmes de chiffrement à clé publique, comment étendre cette approche pour instancier un schéma de signature et, enfin, comment généraliser cet approche pour définir des codes de caractéristique $p \geq 2$. En résumé, nous avons réussi à produire des clés très compactes basées sur la famille reputée de codes de Goppa.

Bien qu'efficace, les codes de Goppa p-adiques ont une propriété non désirée: une forte structure algébrique. Cela nous amène à notre deuxième approche, en utilisant des codes LDPC avec densité augmentée, notés codes MDPC. Ce sont

des codes basés sur des graphes, qui sont libres de toute structure algébrique. Il est très raisonnable de supposer que les codes MDPC sont distinguable seulement en trouvant des mots de code de poids faible dans son dual, un problème de la théorie de codes largement étudié. Ceci constitue un avantage important non seulement par rapport à toutes les autres variantes du système de McEliece à clés compactes, mais aussi en ce qui concerne la version classique basée sur les codes de Goppa binaires. Ici, les clés compactes sont obtenues en utilisant une structure quasi-cyclique.

Chapter 1

Introduction

Cryptography is the research field where techniques for achieving secure communication in the presence of adversaries are studied. These techniques rely on challenges to legitimate the right to access the exchanged information. In general, these challenges are computational problems that are easy to solve when the user is equipped with some privileged information and hard otherwise.

In the last decades, a wide range of mathematical problems have been analyzed in order to identify those which hold interesting cryptographic properties. Number theory is a rich field for such a purpose. Integer factorization and discrete logarithm are remarkable examples from this field, supporting the wide-spread RSA cryptosystem [RSA78] and the cryptosystems based on elliptic curves [Mil86], respectively. These schemes ensure, at the same time, adequate security levels and reasonable time and space complexity.

Despite some good features, problems from number theory are not optimal in many other particularities. For example, they are probably insecure against attacks mounted with the help of quantum computers. This distrust comes from the work of Peter Shor [Sho97], where quantum polynomial algorithms for solving the integer factorization and discrete logarithm problems were presented. Although quantum computers are not available yet, the modern society must dispose of secure alternatives if or when this technology becomes a reality. Fortunately, some work has already been done on this direction.

At the moment, there exist four categories of cryptography that seem to overcome this weakness. Cryptography based on: coding-theory, hash functions, lattices and multivariate quadratic equations (MQE). The first two categories date from the 1970's and present cryptosystems whose security remain almost unscathed so far (see Merkle's hash-tree public-key signature [Mer79] and McEliece cryptosystem using binary Goppa codes [McE78]). The two other categories (lattices and MQE based cryptography) are more recent and date from the 1990's. It is worth to mention that lattice-based cryptography has already presented interesting security-reduction proofs.

In this thesis, we will focus on the first category, code-based cryptography.

1.1 Code-Based Cryptography in a Nutshell

Coding-theory proposes the study of techniques to efficiently transmit information through channels subject to noise. This is done by correcting the noise at the receiver end. Equivalently, it means decoding the received message into the original one. Linear codes are usually employed for this purpose.

Code-Based Cryptography (CBC) makes use of hard problems related to linear codes to implement different cryptographic primitives. For example, public-key encryption schemes, digital signature schemes and hash functions have been successfully designed employing coding-theory problems.

The McEliece public-key encryption scheme [McE78] is the best known code-based scheme. It dates from 1978, the same period when the public-key cryptography concept was invented [DH76]. In 1986, when elliptic curves cryptography first appeared [Mil86], the Niederreiter public-key code-based encryption scheme was proposed [Nie86]. It is very similar to the McEliece scheme, being called its *dual version* since it uses the dual code instead of the original one. In 2001, the first code-based signature scheme, the CFS scheme [CFS01], was proposed. It is built upon the Niederreiter scheme.

CBC benefits from practical advantages when compared to number-theory cryptography. For example, it is several orders of magnitude faster than their RSA and elliptic-curves counterparts. Besides, it is easily implemented since its operations boil down to the manipulation of simple vectors and matrices.

These code-based schemes share the very same *modus operandi*.

Setup: A linear code that has an efficient decoding algorithm is selected at random. The private-key is a code description that enables efficient decoding. The public-key is a disguised version of this code which should not enable efficient decoding.

Challenge: The idea is to purposely add errors to a codeword of this code in such a way that only the owner of the private code would be able to efficiently decode. Obviously, it is expected that recovering the private code from the public description is computationally unfeasible. Since it is hard to foresee which properties might be leaked by the public description, it is desirable to have the public description indistinguishable from random.

In this sense, code-based cryptography has its security based on two things:

- The hardness of correcting errors in linear codes.
- The hardness of distinguishing the public code description from random.

The decoding problem is believed to be hard after decades of research. No quantum (nor classical) polynomial algorithm able to solve such a problem is known, promoting CBC to the select group called *post-quantum cryptography*. This computational problem has critical importance in contexts such as telecommunications and information systems. All algorithms to solve it have exponential complexity [Pra62, Ste89, BLP08, FS09, MMT11, BJMM12]. Thus any important progress contradicting this hardness would definitely be a breakthrough.

On the other hand, the distinguishing problem has a more complex assessment and strongly depends on the choice of the code-family.

Choosing the Code-Family

The classical McEliece encryption scheme was proposed using binary Goppa codes. Since then, many other code-families have been investigated to instantiate code-based cryptosystems.

In 1986, the Niederreiter scheme was suggested with the use of Generalized Reed-Solomon codes [Nie86]. Six years later, it was proved that this choice for the code-family was insecure [SS92]. In 1991, Gabidulin, Paramonov and Tretjakov studied the use of rank-metric codes [GPT91] for the McEliece scheme. Due to the strong code structure, the proposal has been subsequently broken in [Gib95, Gib96]. In 1994, Sidelnikov studied the use of Reed-Muller codes [Sid94] and it was subsequently broken in [MS07]. In 1996, Janwa and Moreno studied the use of algebraic geometric codes [JM96], which was also broken [FM08]. More attempts to replace Goppa codes by other linear codes will be discussed in Section 1.2, but with these few examples we can already see that the task of choosing secure code-families for CBC has not been easy.

Curiously, the classical McEliece proposal using binary Goppa codes has resisted almost unscathed from practical attacks for more than thirty years. For this reason, they became a consensus in the code-based community. They have been suggested to instantiate both Niederreiter encryption scheme and CFS digital signature scheme as well.

Although reputable, it is worth mentioning that Goppa codes might not be the optimal choice. A recent result [FGO+11] presented a distinguisher for some Goppa codes (of high rate). A distinguisher does not necessarily lead to a practical attack, but should be seen as a first-step on that direction. In short, it reinforces that distinguishing Goppa codes might not be always a hard problem.

Practical applicability

Although efficient in terms of complexity, simple to implement and probably secure against quantum attacks, code-based cryptography is not widely employed. This is mainly due to two major drawbacks. One is the excessive overhead per message, which can be partially circumvented by introducing information data into the errors. The other one refers to the key sizes which are much larger than those used in number-theory cryptography. This discrepancy is showed in Table 1.1. It presents the key sizes of elliptic curves (EC), RSA and the classical McEliece scheme instantiated with binary Goppa codes [BLP08], for different security levels.

Security Level	EC	RSA	McEliece
2^{80}	160	1 024	460 647
2^{128}	256	3 072	1 537 536
2^{256}	512	15 360	7 667 855

Table 1.1: key size comparison in bits.

Finding ways to reduce the key sizes of code-based cryptosystem without compromising its security is therefore a relevant research problem. It may finally promote CBC to real-world applications, strengthens its position as a competitive alternative to number theory cryptography. In next section, we discuss some recent results toward this goal.

1.2 Related Work

In this section, we describe the works that influenced our contributions. In summary, they studied the impacts of replacing binary Goppa codes by other linear codes to instantiate code-based cryptosystems, aiming at reducing the key sizes. We divide this discussion in two parts, regarding algebraic and graph-based codes.

Algebraic Codes: In 2005, Gaborit presented a way to significantly reduce the key sizes of McEliece-type schemes [Gab05] which consists of replacing binary Goppa codes by quasi-cyclic BCH codes. These codes benefit from the compact representation of cyclic blocks. Unfortunately, this proposal contains weaknesses. The private code is selected from a too restricted set of possible codes, implying that the secret code is, in practice, publicly known. Moreover the the permutation used to map the public and the private codes is too restrictive, leading thus to a total break of the scheme.

In [BCGO09], Berger *et al.* presented another way to reduce the McEliece keys. It consists of using quasi-cyclic alternant codes. The authors introduced an interesting way to hide the secret code structure, making public only a punctured-permuted subfield subcode version of the secret code. Thus the public and private code are not permutation-equivalent as usually adopted. However, the combination of algebraic and cyclic structures led to successful structural attacks[1]. These attacks boil down to setting up a system of equations derived from the alternant structure. Among other reasons, the cyclicity adds an important redundancy to this system, drastically reducing the number of unknown variables, thus enabling the computational feasibility of the attack.

Graph-Based Codes: In 2000, Monico *et al.* investigated the use of LDPC codes to instantiate the McEliece cryptosystem [MRS00]. These codes have sparse parity-check matrices, which are used to perform efficient decoding using belief-propagation decoding techniques and can be efficiently stored. In the end, the authors concluded that pure LDPC codes, like those used for telecommunication applications, are not secure to code-based cryptography. The problem is related to the easiness of recovering the low-weight codewords present in the dual code. Once recovered, they allow the adversary to build an alternative sparse parity-check matrix which permits efficient decoding.

In [BCG06, BCGM07, BC07], a new trend suggesting QC-LDPC codes for achieving compact keys started. Here, the authors suggest the use of auxiliary matrices to artificially increase the dual codeword weight. Unfortunately, this extra structure was defined in a too constrained way, leading to successful attacks. Finally, in [BBC08], Baldi *et al.* managed to present a QC-LDPC McEliece variant that apparently resists to structural attacks and achieve compact-keys. Note this last proposal still makes use of auxiliary matrices to artificially hide the low-weight dual codewords.

[1] These attacks have also affected some parameters of one of our results (p-adic codes defined over intermediate fields).

1.3 Thesis

1.3.1 Objectives

The objective of this thesis is twofold:

1. Improve the efficiency of code-based cryptosystems, possibly allowing one to deploy such solutions in practical scenarios. Since the main obstacle to this purpose has been its excessively long key-sizes, one of our subjects of investigation consists of finding ways to reduce the key-sizes of code-based cryptosystem without compromising its security.

2. The distinguishing problem as we have now is dangerously obscure. It is not a desirable feature for code-based cryptography, being the keystone of many recent attacks. Thus, the other subject of investigation consists of reducing the distinguishing problem to some well known problem.

1.3.2 Metrics and Methodology

The *metrics* to measure if we attained the aforementioned objectives are:

1. key sizes of a few kilobits.

2. Distinguishing problem reduced to a well known coding-theory problem.

The *methodology* to accomplish our objectives will be mostly founded in investigating the use of different families or sub-families of linear codes to instantiate code-based cryptosystems.

1.3.3 Original Contributions

In this section, we describe our contributions. We split it in two parts, regarding our progresses using algebraic codes and graph-based codes.

Algebraic Codes: Since 1978, when the original proposal of the McEliece cryptosystem was presented, binary Goppa codes have resisted against structural attacks. Remarking that all previous (in most cases, unsuccessful) attempts to reduce key-sizes of code-based cryptosystem have replaced Goppa codes by other code families, we decided to investigate the possibility of achieving compact keys without leaving the Goppa family. The use of cyclicity did not seem to be adequate for Goppa codes and thus we sought for alternative compact-representation matrix structures. Dyadic matrices ended up to be good candidates for this purpose. Using similar techniques to hide the secret code as introduced in [BCGO09], we managed to construct the innovative class of binary Quasi-Dyadic Goppa codes [MB09a]. As a main result, it achieves public keys of only 9216 bits for 80-bits of security. Later, this approach was extended to instantiate the CFS signature scheme [BCMN10] and generalized to Goppa codes defined over finite fields of characteristic greater than 2 [BLM11].

Graph-Based Codes: LDPC codes are free of algebraic structure. This is an interesting feature for code-based cryptography. Nonetheless, the easiness of finding their dual low-weight codewords would compromise the security of an LDPC-McEliece variant. Thus some countermeasures are required. All previous attempts have tried to use auxiliary matrices to hide these low-weight codewords, being successful attacked (except [BBC08]). In this scenario, we came up with an innovative approach: the use of LDPC codes of higher density, *a.k.a.* Moderate Density Parity-Check (MDPC) codes, discarding all auxiliary structures. The parity-check equations of increased weight prevent the effectiveness of dual low-weight codeword attacks, at the price of degrading the error-correction capability. Fortunately, in code-based cryptography, we are not necessarily interested in correcting many errors, but only a number of errors which ensures an adequate security level, a condition satisfied by MDPC codes. This work resulted in the proposal of two McEliece variants [MTSB13]. One from MDPC and another from QC-MDPC codes. It is quite reasonable to assume that these codes are only distinguishable by finding their dual low-weight codewords, a largely studied coding-theory problem. This represents an important advantage not only in comparison to all previous compact-keys McEliece-like variants but also regarding the classical McEliece based on binary Goppa codes. Moreover, the QC-MDPC variant provides keys of only 4801 bits, for 80-bits of security.

We present as an appendix the preliminary investigation on extending the MDPC approach to construct nearly sparse parity-check matrices of random codes. Random codes have minimum weight close to what is known as the Gilbert-Varshamov (GV) distance. In short, the idea is to construct codes with parity-check equations of weight close to the GV distance of a code of same size. In this sense, this construction would produce nothing more than a convenient (nearly sparse) description of random codes.These codes should be able to correct a quite small number of errors using belief propagation techniques, but still good enough to provide an advantage in comparison to adversaries. So far, we have not been able to develop techniques to prove the validity of this construction. Nevertheless, this preliminary investigation may provide some interesting thoughts towards achieving public-key cryptography based on random codes.

1.3.4 Organization

The remainder of this thesis is organized as follows.

- Chapter 2: Background.
  It summarizes the essential concepts related to coding-theory, cryptography and code-based cryptography.

Part I: Algebraic Codes. This part encompasses our results on the domain of algebraic codes.

- Chapter 3: Introduction.
  It describes some algebraic codes useful to the understanding of our proposal, and some previous attempts of reducing the key size of code-based cryptosystems by using algebraic codes.

- Chapter 4: p-adic Goppa codes.
  It describes our proposal of using p-adic Goppa codes to reduce the key size of code-based cryptosystems.

Part II: Graph-Based Codes. This part encompasses our results on the domain of graph-based codes.

- Chapter 5: Introduction.
  It describes LDPC codes and how they are decoded and some previous attempts of reducing the key size of code-based cryptosystems by using graph-based codes.
- Chapter 6: Moderate-Density Parity-Check (MDPC) Codes.
  It describes our proposal of using MDPC codes to reduce the key size of code-based cryptosystems and achieve better indistinguishability condition.

Part III: Synopsis.

- Chapter 7: Conclusion.
  This chapter compares both algebraic and graph-based approaches, present our conclusions and ideas for future works.

Appendices:

- Appendix A: Random Codes.
  It describes our preliminary investigation on generalizing the MDPC approach to define random codes.

Chapter 2

Background

This introductory chapter is divided into the general background of coding theory, the basic concepts of cryptography and a discussion regarding the combination of both (coding-theory and cryptography) leading to the code-based cryptography topic.

2.1 Coding Theory

Coding theory proposes the study of techniques to efficiently transmit information through channels subject to noise. This goal is achieved by the employment of *codes*, which allow: encoding, decoding and, under certain conditions, error correction capability. Encoding is the process which maps a message, here called a *plaintext*, to an element of the code, here called a *codeword*. Decoding is a priori the process which performs the opposite, mapping a codeword to a plaintext. However, adding some redundancy during encoding, it might be possible to recover the plaintext even when a noisy codeword, here called simply *word*, is received. Figure 2.1 presents a flowchart describing this process.

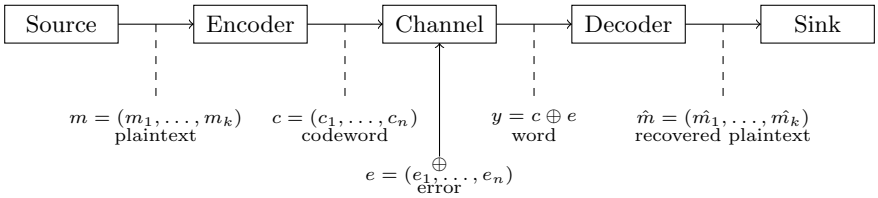

Figure 2.1: Communication channel.

There are different types of codes. A dictionary which maps univocally plaintexts into codewords can be seen as a code. However it is easy to see that this construction is rather restrictive and does not provide many tools for error-correction. Linear codes, on the other hand, provide many properties that are useful for this purpose.

2.1.1 Linear Codes

Let p be a prime number and $q = p^m$, for some positive integer m. Additionally, let $\mathbb{F}_q$ be a finite field of q elements. Informally, a linear code defined over $\mathbb{F}_q$ is a code whose the encryption function is a linear map.

Definition 2.1 (Linear map)**.** *A function $f : D \to I$ is a linear map iff for all $x, y \in D$:*

1. $f(x + y) = f(x) + f(y)$.
2. $f(\alpha x) = \alpha f(x)$, $\forall \alpha \in \mathbb{F}_q$.

This property leads to a very natural way to represent and operate over linear codes: using vector subspaces.

Definition 2.2 (Linear code)**.** *An (n, k)-linear code $\mathcal{C}$ is a vector subspace of length n and dimension k.*

Thus, codes can be defined and represented by matrices, and tuples (plaintexts, codewords and words) by vectors. Below we present two matrices used to represent a linear code.

Definition 2.3 (Generator matrix)**.** *A matrix $G \in \mathbb{F}_q^{k \times n}$ is called a* generator matrix *for a (n, k)-linear code $\mathcal{C}$ iff*

$$\mathcal{C} = \{mG \mid m \in \mathbb{F}_q^k\}.$$

Definition 2.4 (Parity check matrix)**.** *A matrix $H \in \mathbb{F}_q^{(n-k) \times n}$ is called a* parity-check matrix *for a (n, k)-linear code $\mathcal{C}$ iff*

$$\mathcal{C} = \{c \in \mathbb{F}_q^n \mid Hc^T = 0\}.$$

Remark 1. For any generator-matrix $G \in \mathbb{F}_q^{k \times n}$ and parity-check matrix $H \in \mathbb{F}_q^{r \times n}$ of an (n, k)-linear code $\mathcal{C}$, it holds that

$$HG^T = 0.$$

From the previous remark, a parity-check (generator) matrix can be computed from a generator (parity-check) matrix in polynomial time. A generator matrix provides a straightforward way to encode messages: the codeword $c \in \mathbb{F}_q^n$ corresponding to a plaintext $m \in \mathbb{F}_q^k$ is obtained from $c = mG$. On the other hand, a parity-check matrix allows the syndrome computation, an important step for decoding.

Definition 2.5 (Syndrome)**.** *The syndrome $s \in \mathbb{F}_q^r$ of a vector $c \in \mathbb{F}_q^n$ with respect to a parity check matrix $H \in \mathbb{F}_q^{r \times n}$ is $s^T = Hc^T \in \mathbb{F}_q^r$.*

Remark 2. From Definition 2.4, the syndrome of any codeword is always the zero-vector.

Along the text, we assume a codimension $r = n - k$. Next we gather other basic concepts related to linear codes.

Definition 2.6 (Hamming weight)**.** *The Hamming weight (or simply weight)* wt *of a vector $x \in \mathbb{F}_q^n$ is the number of its nonzero components.*

Definition 2.7 (Hamming distance)**.** *The Hamming distance (or simply distance) $dist(x, y)$ of two vectors $x, y \in \mathbb{F}_q^n$ is the number of components whose they differ, i.e. $dist(x, y) = \mathsf{wt}(x - y)$.*

Different linear codes have different properties. One of the most important property of a linear code is its *minimum distance*, which provides an upper-bound for its error correction capability.

Definition 2.8 (Minimum distance)**.** *The minimum distance d_0 of a linear code $\mathcal{C}$ is the minimum distance of its codewords*

$$\begin{aligned} d_0 &= \min dist(c, c') \\ &= \min wt(c - c'), c \in \mathcal{C}, c' \in C, c \neq c. \end{aligned}$$

Equivalently, it is the minimum weight of any nonzero codeword

$$d_0 = \min_{\substack{c \in \mathcal{C} \\ c \neq \mathbf{0}}} \mathsf{wt}(c), c \in \mathcal{C}.$$

Theorem 2.1 (Minimum distance and error correction capability [MS78])**.** *A linear code with minimum distance d_0 can correct up to $\lfloor \frac{d_0-1}{2} \rfloor$ errors.*

The relationship between minimum distance and error correction capability stated above can be explained through a vector space interpretation. Suppose that each codeword is a point in the vector space, rounded by a sphere. The radius of this sphere is such as the whole vector space is fulfilled by these spheres without intersections. Since each codeword is at least at distance d_0, the radius of this sphere should not be greater than $\lfloor \frac{d_0-1}{2} \rfloor$. In this scenario, a codeword that is perturbed by an error of weight not greater than $\lfloor \frac{d_0-1}{2} \rfloor$ can be unambiguously associated to the single, closest codeword. Along the text, we will work with different types of linear codes that perform this association between words and codewords (i.e., error-correction) using a wide variety of techniques.

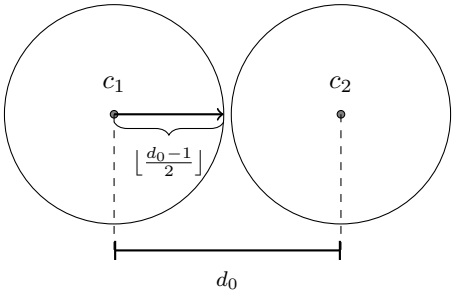

Figure 2.2: Two codewords separated by a distance of d_0.

2.1.2 Algebraic Codes

Algebraic codes perform efficient decoding taking advantage from some algebraic structure. In general, this structure consists of polynomials over a ring. By evaluating such polynomials, it is possible to determine the error positions and error values present in a noisy codeword. Examples of algebraic codes recently used in cryptography: Generalized Reed-Solomon (GRS), Alternant and Goppa. In Chapter 3 of Part I, the definitions and features of these codes are presented.

2.1.3 Graph-Based Codes

Low-Density Parity-Check (LDPC) codes, which are graph-based codes, have been (re-)discovered recently. They have interesting properties for error correction that had been unexploited for decades due to the limited computing hardware. This story begins in 1963, when Robert G. Gallager introduced what the LDPC codes [Gal63], which are able to approach the channel capacity. These codes have no algebraic structure and just meet a very simple combinatorial property: they admit a sparse parity-check matrix. In 1996, David J.C. MacKay and Radford M. Neal have re-enlightened this concept [MN96]. This work has been followed by several other works that, using the more advanced available hardware, could attest their excellent error correction attributes. Recently, they have also been used for cryptographic purposes. In Chapter 5 of Part II, we present the definitions and features of these codes.

2.2 Cryptography

As discussed before, cryptography gathers techniques for achieving secure communication in the presence of adversaries. This can be done in several ways and regarding different purposes. In this section, we describe a few different cryptographic concepts and primitives that will be useful along the text.

Hard Problems

Cryptosystems rely their security on computational problems that are *easy* to solve when the user is equipped with some privileged information and *hard* otherwise. The concepts of *easy* and *hard* are purposely loosely defined, although some insights on how classify computational problems within these two qualifiers are discussed next.

Computational problems are divided into *complexity classes*. We assume the familiarity of the reader with the concepts of deterministic and non-deterministic Turing machines. The most important complexity classes are:

- Class $\mathcal{P}$: Problems that can be solved in polynomial time in a deterministic Turing machine. In practice, these are the *easy* problems.
- Class $\mathcal{NP}$: Problems that can be solved in polynomial time in a non-deterministic Turing machine. Additionally, a solution for this problem can be verified in polynomial time in a deterministic Turing machine.
  - Class $\mathcal{NP}$-Hard: Problems that are at least as hard as the hardest problem in $\mathcal{NP}$.
  - Class $\mathcal{NP}$-Complete: Subclass of $\mathcal{NP}$ which contain decisional problems that are also in $\mathcal{NP}$-Hard. A problem in $\mathcal{NP}$ is in $\mathcal{NP}$-Complete if it is polynomially reducible to any other problem in $\mathcal{NP}$-Complete. In practice, these are the most difficult problems in $\mathcal{NP}$.

Figure 2.3 illustrates the relationship between these classes, assuming the validity of the conjecture $\mathcal{P} \neq \mathcal{NP}$.

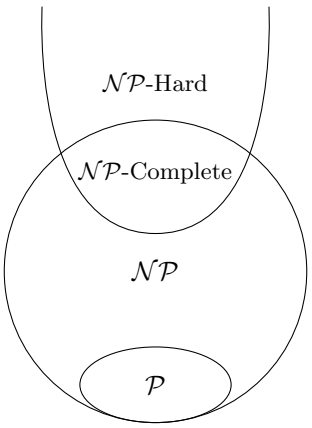

Figure 2.3: Relationship between some complexity classes.

Problems may not be always hard or easy. Their behavior can variate depending on the type of input. In this context, the complexity class of a problem only tells us its worst-case behavior, i.e. how difficult is to solve this problem when considering the worst input possible. In cryptography, however, we are interested on problems that will remain hard for any instance (or, except for a negligible number of instances that can be beforehand identified and eliminated). These are the hard problems in the average-case.

In practice, even problems not proven to be hard on average are considered for cryptography. This is the case of factoring numbers in prime numbers, a problem used in the widely deployed RSA cyrptosystem [RSA78]. Moreover, the way the cryptosystem or the protocol use the underlying problem may introduce vulnerabilities which do not necessarily depend on solving such a problem.

For practical purposes, we will be interested on problems that are most likely hard on average, for which no algorithm able to solve it in less than a given exponential number of steps is known. We call the exponent of this minimum number of steps as the security parameter of the scheme.

Definition 2.9 (Security parameter)**.** *λ is the security parameter of a scheme if there is no known algorithm that can break the scheme in less than 2^{λ} steps.*

Next, we discuss some cryptographic primitives.

2.2.1 Encryption

An encryption schemes is designed to ensure the secrecy of a transferred message. There are two types of encryption schemes, symmetric and asymmetric. In this text, we will be solely interested on asymmetric schemes, but to illustrate the concept, we describe both.

The *symmetric scheme* needs, at some point, a secure channel between the parts. The *asymmetric schemes* do not require this secure channel, but they are much less efficient in terms of algorithmic complexity.

Both schemes use keys, *i.e.* piece of information that when combined with a cryptographic procedure allows one to hide the content of a message from adversaries. A more practical difference between symmetric and asymmetric

schemes regards the different number of keys used in the process. Symmetric schemes use only one key. This explains the requirement of having a secure channel at some point (for sharing such a key between both users). Asymmetric schemes use a pair of keys that can be distributed by a trusted third party.

Next, we defined both types of encryption schemes. Let S be the key-space from where the keys are sampled. In the symmetric case, S is the set of all keys, and in the asymmetric case, S is the set of all key pairs. Let $K : \mathbb{Z} \to S$ be a key-generation function which receives the security parameter λ and returns an element randomly chosen from S which reaches such a security level. We begin with the symmetric scheme, which uses the cryptographic encryption and decryption function $\phi : S \times \mathbb{F}_q^n \to \mathbb{F}_q^n$.

KeyGen:

1. $sk \leftarrow K(\lambda)$.

Enc: The encryption of $m \in \mathbb{F}_q^n$ is:

1. $x \leftarrow \phi(sk, m)$.

Dec: The decryption of $x \in \mathbb{F}_q^n$ is:

1. $m \leftarrow \phi(sk, x)$.

Table 2.1: Symmetric Encryption Scheme.

An asymmetric encryption scheme uses two keys, a secret to encrypt, and a public to decrypt. There is no previous communication between sender and receiver for agreeing about the key, at the price of requiring a third party authority trusted by both users to distribute the key pair.

We consider S as a tuple of (S_s, S_p), where S_s is the set of secret keys and S_p is the set of public keys. They are univocally associated to each-other. Consider $\phi : S_s \times \mathbb{F}_q^k \to \mathbb{F}_q^n$ the encryption function and $\theta : S_p \times \mathbb{F}_q^k \to \mathbb{F}_q^n$ the decryption function. For any pair $(sk, pk) \in S$ and message $m \in \mathbb{F}_q^k$, it holds: $\theta(sk, \phi(pk, m)) = m$. The asymmetric encryption scheme is defined in Table 2.2.

KeyGen:

1. $(sk, pk) \leftarrow K(\lambda)$.
2. pk is public. sk is kept in secret by its owner.

Enc: The encryption of $m \in \mathbb{F}_q^k$ is:

1. $x \leftarrow \phi(pk, m)$.

Dec: The decryption of $x \in \mathbb{F}_q^n$ is:

1. $m \leftarrow \theta(sk, x)$.

Table 2.2: Asymmetric Encryption Scheme.

2.2.2 Digital Signature

A digital signature scheme permits to validate the authenticity of a digital message or document. For this purpose, the scheme must attain three features.

- *Authentication:* It allows a receiver of a digitally signed document to attest that such a document was produced by a known sender.
- *Non-repudiation:* It states that the sender cannot deny the message's authority.
- *Integrity:* It allows the receiver to attest the message was not modified during transmission.

Digital signature schemes are asymmetric schemes. Its private-key is able to sign documents. Its public-key is able to verify signatures. For the signing process, it is convenient when the document to be signed, here denoted by $\mathcal{D}$, is an element of the ciphertext space. In this setting, the *signer* uses its private key to decrypt $\mathcal{D}$, obtaining a corresponding plaintext x. Then he sends the tuple $(\mathcal{D}, x)$ to the receiver. The verification is done by re-encrypting x and comparing it with $\mathcal{D}$. The aforementioned explanation, although illustrative, is not quite precise. The document has an unknown size, which prejudices the mapping of documents into ciphertexts. It is preferable to use a fixed-length identifier for each document. This is done by a *hash* function.

Definition 2.10 (Hash function)**.** *A* hash function *is a function that maps strings of arbitrary length to strings of fixed-length, called* hash-values*.*

We need hash functions that are *collision-resistant*, *i.e.* two different documents should not have the same signature. Moreover, it is not ensured that the hash-values can be seen as ciphertext. When this condition is not satisfied, an intermediate step must map the hash-value to ciphertexts. On the other hand, we call The Full Domain Hash (FDH) approach, when it is satisfied. For simplicity, we assume a collision-resistant hash $\mathcal{H}$ in the FDH case. C, P and S denote the ciphertext, plaintext and key spaces. The notation come from the asymmetric encryption schemes. Table 2.3 describes a signature scheme.

KeyGen:

1. $(sk, pk) \leftarrow S$.

Sign: To sign a document $\mathcal{D} \in \mathbb{F}_q^{n>0}$:

1. $h \leftarrow \mathcal{H}(\mathcal{D}) \in C$.
2. $x \leftarrow \theta(k_s, h) \in P$.
3. The signature is: $(\mathcal{D}, x)$.

Verify: To verify a signature $(\mathcal{D}, x)$:

1. $h' \leftarrow \mathcal{H}(\mathcal{D})$.
2. If $h' = \psi(k_p, x)$, then **accept**. Otherwise, **reject**.

Table 2.3: *Simplified* Digital Signature Scheme

2.3 Code-Based Cryptography

In this section, we formally describe the most important code-based cryptosystems and provide their theoretical and practical security assessment.

2.3.1 Code-Based Cryptosystems

The cryptosystems here described are based on the same security idea.

> **General security idea.** A linear code that disposes of an efficient decoding algorithm is selected at random. The private-key is a code description that enables efficient decoding. The public-key is a disguised version of this code which should not enable efficient decoding. The challenge consists of purposely adding errors to a codeword of this code in such a way that only the owner of the private code is able to efficiently decode.

Obviously, it is expected that recovering the private code from the public description is computationally unfeasible. Since it is hard to foresee which properties might be leaked by the public description, it is reasonable to have this public description indistinguishable from random.

We denote by $\Psi_{\mathcal{C}}$ a t-error correction procedure for a linear code $\mathcal{C}$. Suppose that a word $y = x + e$ is transmitted, where $x \in \mathcal{C}$ and $\mathsf{wt}(e) = t$. In the McEliece scheme, the output of $\Psi_{\mathcal{C}}(y)$ will be the codeword x, whilst in the Niederreiter and CFS scheme it will be the error vector e.

McEliece Cryptosystem

The McEliece cryptosystem [McE78] is described in Table 2.4. The key-generation was already discussed in the general security idea. The encryption consists of encoding a message and adding it to an error vector. The decryption consists of correcting such errors and then recovering the original plaintext.

KeyGen:

1. Generate a linear code $\mathcal{C} - (n, k)$ able to correct t errors.

– Secret-Key: $\Psi_{\mathcal{C}}$, a t-error correcting procedure for $\mathcal{C}$.
– Public-Key: $G \in \mathbb{F}_q^{k \times n}$, a systematic generator-matrix for $\mathcal{C}$.

Enc: The encryption of $m \in \mathbb{F}_q^k$ is:

1. Select at random a vector $e \in \mathbb{F}_q^n$ of weight t.
2. $x \leftarrow mG + e$.

Dec: The decryption of $x \in \mathbb{F}_q^n$ is:

1. $m \leftarrow$ Extract the first k positions of $\Psi_{\mathcal{C}}(x)$.

Table 2.4: McEliece Encryption Scheme.

Note that we use the public matrix in systematic form. This is possible due to the use of a CCA2-secure conversion, e.g. [KI01], preventing the leakage of information when using such a compact representation.

The McEliece cryptosystem is often described using a scrambling matrix $S \in \mathbb{F}_q^{k\times k}$ and a permutation matrix $P \in \mathbb{F}_q^{n\times n}$. Thus the transformation from the secret to the public code description are made through matrix products. However, the only security requirement necessary for McEliece-like cryptosystems is that the public-key does not reveal any useful information for decoding. Since this can be done in several ways, we stick to the simplest and more general McEliece description as presented in Table 2.4, which is in accordance with the security reduction to be presented in Section 2.3.2.

Niederreiter Cryptosystem

The Niederreiter cryptosystem [Nie86] is known as the dual version of McEliece cryptosystem. It permits smaller public key-sizes and equivalent security [LDW94]. The main differences from McEliece scheme is that the message is represented by the error vector and that parity-check matrices are used instead of generator matrices.

In this scheme, we need a function that translates messages into vectors of a given length and weight. This is done by what is called a *constant-weight encoding function*, here denoted by φ (see [Sen05]). This function receives a vector of length n' and convert it into another vector of length $n > n'$ of constant weight t. Obviously, not all vectors can be encoded uniquely into vectors of fixed length and weight. In fact, working on a finite field of q elements, only $\log_q \binom{n}{t}$ messages can be univocally encoded into vectors of weight t and length n. The scheme is described in Table 2.5.

KeyGen:

1. Generate a linear code $\mathcal{C} - (n, k)$ able to correct t errors.

- Secret-Key: $\Psi_{\mathcal{C}}$, a t-error correcting procedure for $\mathcal{C}$.
- Public-Key: $H \in \mathbb{F}_q^{(n-k)\times n}$, a parity-check matrix for $\mathcal{C}$.

Enc: The encryption of $m \in \mathbb{F}_q^{n'}$ is:

1. $s \leftarrow H(\varphi(m))^T$.

Dec: The decryption of $s \in \mathbb{F}_q^{(n-k)}$ is:

1. $m \leftarrow \varphi^{-1}(\Psi_{\mathcal{C}}(s))$.

Table 2.5: Niederreiter Encryption Scheme.

CFS Digital Signature Scheme

As discussed in Section 2.2.2, cryptosystems that permit see any fixed-length vector as a ciphertext can be easily translated into a digital signature scheme. Unfortunately, this is not the case in code-based cryptography.

The CFS signature scheme [CFS01] was the first proposal to present a solution to this problem. It is built upon the Niederreiter scheme. This encryption scheme has decodable syndromes as its ciphertexts and error vectors of fixed weight as its plaintexts. Note that we cannot directly construct a digital signature from Niederreiter scheme, since a fixed length vector (*e.g.* a hash of a document) is not necessarily a decodable syndrome. Therefore we need a way to map any input of fixed length to the set of decodable syndromes. There are two ideas to overcome this issue:

Adding a random vector. The idea here is to sample a random vector and add it to the hash of the document. Then we check if it is a decodable syndrome. This process is repeated until a decodable syndrome be found.

Complete decoding. The idea here is to be able to decode any output given by the hash function. Thus, one needs to modify the decoding algorithm in order to be able to correct more than t errors. A simple and sufficient approach is to use exhaustive search for correcting the additional errors. More precisely, for a code of length n and codimension r, we need to consider a δ additional errors as the smallest integer such that $\binom{n}{t+\delta} > 2^r$. Then, any syndrome can (with a good probability) be decoded into an error vector of weight $t + \delta$ or less.

Obviously the efficiency of this scheme strongly depends on how many attempts one has to try (random vectors or patterns of weight at most δ) until a decodable syndrome be found. This condition makes the choice of the code family quite challenging. Only codes that have a high density of decodable syndromes are suitable, *e.g.* binary Goppa codes. This issue is discussed in details in Section 4.1.5. We denote by $\mathcal{H}$ the hash function and by φ the constant-weight encoding function. The scheme is described in Table 2.6.

Parallel-CFS

An unpublished attack against the CFS scheme due to D. Bleichenbacher is described in [FS09]. It represents a successful way to forge valid CFS signatures, for its usual parameters. Improved parameters would result in a signature scheme with excessive cost of signing time or key length. This attack is described in Section 2.3.3.

In [Fin10], Finiasz introduces the Parallel-CFS, which fixes the security issues related to Bleichenbacher's attack. The idea here is to produce not only one hash (using a function $\mathcal{H}$) from a document D. Instead one produces σ hashes (using σ different functions $\mathcal{H}_1, \ldots, \mathcal{H}_\sigma$) and sign all $\mathcal{H}_1(D), \ldots, \mathcal{H}_\sigma(D)$ in parallel.

For well chosen parameters, this strategy avoids the effectiveness of Bleichenbacher's attack, at the price of increasing the cost of signing, verifying and storing signatures by a factor σ.

KeyGen:

1. Generate a linear code $\mathcal{C} - (n, k)$ able to correct t errors.

– Secret-Key: Ψ, a t-error correcting procedure for $\mathcal{C}$.
– Public-Key: $H \in \mathbb{F}_q^{(n-k)\times n}$, a parity-check matrix for $\mathcal{C}$.

Sign: The signature s of a document $\mathcal{D}$ is:

1. $z \leftarrow \mathcal{H}(\mathcal{D})$.
2. Choose a vector i at random.
3. $s \leftarrow \mathcal{H}(z||i)$.
4. While s is not decodable, repeat steps 2 and 3.
5. $e \leftarrow \Psi(s)$.
6. $s \leftarrow (\varphi^{-1}(e)||i)$.

Verify: The verification of a document $\mathcal{D}$ of signature $(\varphi^{-1}(e)||i)$ is:

1. $e \leftarrow \varphi(\varphi^{-1}(e))$.
2. $s_1 \leftarrow He^T$.
3. $s_2 \leftarrow \mathcal{H}(\mathcal{H}(\mathcal{D})||i)$
4. if $s_1 = s_2$ then **accept**.
   Otherwise, **reject**.

Table 2.6: CFS Digital Signature Scheme.

2.3.2 Security-Reduction Proof

A security reduction is a proof that an adversary able to attack the scheme is able to solve some (presumably hard) computational problem with a similar effort. In this section, we recall the security reduction presented in [Sen09] to the Niederreiter scheme [Nie86], which is as secure as the McEliece scheme [LDW94].

Let $\mathcal{S}_n(0, t)$ denote the sphere centered in zero of radius t in the Hamming space $\mathbb{F}_2^n$ and let Ω denote the probability space consisting of the sample space $\mathcal{H}_{n,k} \times \mathcal{S}_n(0, t)$ equipped with a uniform distribution. Moreover let:

– $\mathcal{F}_{n,k}$: A code family of (n, k)-linear codes able to correct t.

– $\mathcal{H}_{n,k}$: The set of all full rank matrices in $\mathbb{F}_2^{k\times n}$.

– $\mathcal{K}_{n,k} \subset \mathcal{H}_{n,k}$: The public-key space of $\mathcal{F}_{n,k}$.

A distinguisher, decoder and adversary against $\mathcal{K}_{n,k}$-Niederreiter scheme are described in Table 2.7.

Distinguisher. A program $\mathcal{D} : \mathcal{H}_{n,k} \longrightarrow \{0,1\}$ is a (T,ϵ)-distinguisher for $\mathcal{K}_{n,k}$ (*vs.* $\mathcal{H}_{n,k}$) if it runs in time at most T and the *advantage of* $\mathcal{D}$ *for* $\mathcal{K}_{n,k}$

$$Adv(\mathcal{D}, \mathcal{K}_{n,k}) = |\Pr_{\Omega}[\mathcal{D}(H) = 1 \mid H \in \mathcal{K}_{n,k}] - \Pr_{\Omega}[\mathcal{D}(H) = 1]|$$

is greater than ϵ.

Decoder. A program $\phi : \mathcal{H}_{n,k} \times \mathbb{F}_2^r \longrightarrow \mathcal{S}_n(0,t)$ is a (T,ϵ)-decoder for $(\mathcal{H}_{n,k}, t)$ if it runs in time at most T and its *success probability*

$$Succ(\phi) = \Pr_{\Omega}[\phi(H, eH^{\mathrm{T}}) = e]$$

is greater than ϵ.

Adversary. A program $\mathcal{A} : \mathcal{H}_{n,k} \times \mathbb{F}_2^n \longrightarrow \mathcal{S}_n(0,t)$ is a (T,ϵ)-adversary against $\mathcal{K}_{n,k}$-Niederreiter if it runs in time at most T its *success probability*

$$Succ(\mathcal{A}, \mathcal{K}_{n,k}) = \Pr_{\Omega}[\mathcal{A}(H, eH^{\mathrm{T}}) = e \mid H \in \mathcal{K}_{n,k}]$$

is greater than ϵ.

Table 2.7: Distinguisher, Decoder and Adversary definitions.

An adversary against $\mathcal{K}_{n,k}$-McEliece could be defined as a program $\mathcal{H}_{n,k} \times \mathbb{F}_2^n \to \mathbb{F}_2^{(n-r)} \times \mathcal{S}_n(0,t)$ of probability space Ω and sample set $\mathcal{H}_{n,k} \times \mathbb{F}_2^k \times \mathcal{S}_n(0,t)$. As stated before, this setup would only make all the statements and proofs more cumbersome. Next, the proposition which supports the security reduction.

Proposition 2.1 ([Sen09])**.** *Given the security parameters* (n, r, t) *and* t*, if there exists a* (T,ϵ)*-adversary against* $\mathcal{K}_{n,k}$*-Niederreiter, then there exists either a* $(T,\epsilon/2)$*-decoder for* $(\mathcal{H}_{n,k}, t)$ *or a* $(T + O(n^2), \epsilon/2)$*-distinguisher for* $\mathcal{K}_{n,k}$ *vs.* $\mathcal{H}_{n,k}$*.*

Proof. Let $\mathcal{A} : \mathcal{H}_{n,k} \times \mathbb{F}_2^r \to \mathcal{S}_n(0,t)$ be a (T,ϵ)-adversary against $\mathcal{K}_{n,k}$-Niederreiter. We define the following distinguisher:

$\mathcal{D}$: input $H \in \mathcal{H}_{n,k}$.

$e \xleftarrow{\$} \mathcal{S}_n(0,t)$

if $(\mathcal{A}(H, eH^{\mathrm{T}}) = e)$ **then return** 1 **else return** 0.

which implies:

$$\begin{aligned} \Pr_{\Omega}[\mathcal{D}(H) = 1] &= \Pr_{\Omega}[\mathcal{A}(H, eH^{\mathrm{T}}) = e] \\ &= \mathrm{Succ}(\mathcal{A}) \\ \Pr_{\Omega}[\mathcal{D}(H) = 1 \mid H \in \mathcal{K}_{n,k}] &= \Pr_{\Omega}[\mathcal{A}(H, eH^{\mathrm{T}}) = e \mid H \in \mathcal{K}_{n,k}] \\ &= \mathrm{Succ}(\mathcal{A}, \mathcal{K}_{n,k}) \end{aligned}$$

thus $Adv(\mathcal{D}, \mathcal{K}_{n,k}) = |Succ(\mathcal{A}, \mathcal{K}_{n,k}) - Succ(\mathcal{A})|$ and particularly:

$$Adv(\mathcal{D}, \mathcal{K}_{n,k}) + Succ(\mathcal{A}, \mathcal{K}_{n,k}) \geq Succ(\mathcal{A})$$

Since $Succ(\mathcal{A}, \mathcal{K}_{n,k}) \geq \epsilon$, we either have $Adv(\mathcal{C}, \mathcal{K}_{n,k})$ or $Succ(\mathcal{A})$ greater or equal to $\epsilon/2$ (recall that both are positive). The running time of $\mathcal{D}$ is equal to the running time of $\mathcal{A}$ increased by the cost for picking e and computing the product eH^{T}, which cannot exceed $O(n^2)$. So either $\mathcal{A}$ is a (T, ϵ)-decoder for $(\mathcal{H}_{n,k}, t)$ or $\mathcal{D}$ is a $(T + O(n^2), \epsilon/2)$-distinguisher for $\mathcal{K}_{n,k}$.

□

A distinguisher for $\mathcal{K}_{n,k}$ *vs.* $\mathcal{H}_{n,k}$ and a decoder for $(\mathcal{H}_{n,k}, t)$ provide a solution respectively to the Code Distinguishing Problem and to Syndrome Decoding Problem as described next.

Problem 1 (Code Distinguishing Problem)**.**
Parameters: $\mathcal{K}_{n,k}$, $\mathcal{H}_{n,k}$.
Instance: a matrix $H \in \mathcal{H}_{n,k}$.
Question: is $H \in \mathcal{K}_{n,k}$*?*

Problem 2 (Computational Syndrome Decoding Problem)**.**
Parameters: $\mathcal{H}_{n,k}$*, an integer* $t > 0$.
Instance: a matrix $H \in \mathcal{H}_{n,k}$ *and a vector* $s \in \mathbb{F}_2^r$.
Problem: find a vector $e \in \mathcal{S}_n(0, t)$ *such that* $eH^T = s$.

Key-Distinguishing Assumption. $(\mathcal{K}_{n,k}, \mathcal{H}_{n,k})$-code distinguishing problem is a *hard* problem.
Decoding Assumption. $(\mathcal{H}_{n,k}, t)$-computational syndrome decoding problem is a *hard* problem.

Table 2.8: Security assumptions for code-based cryptography.

2.3.3 Generic Attacks

The practical security assessment of a public-key cryptosystems encompasses two categories of attacks. Key-recovery attacks and message attacks. The former aims at recovering the private-key from the public-key. The second aims at recovering the corresponding message from a particular ciphertext.

Key-recovery attacks are strongly connected to the concept of distinguishing a public-key from random. If an adversary can distinguish a public-key, it might be possible that some private information is being leaked by the public-key. On the other hand, if no adversary is able to distinguish a public-key from random, then surely no adversary is obtaining private information from the public-key.

In the code-based cryptography context, distinguishing means distinguishing the public generator or parity-check matrix of a linear code from a random matrix. Since this discussion is intended to be generic (and the distinguishing problem depends on which code-family is used), we move forward to the discussion regarding message attacks.

Assuming indistinguishable public-keys, the messages attacks in code-based cryptography consists of correcting errors in a random linear code. This problem is easily translated to Problem 2 (computational syndrome decoding). Here, we describe the best strategies to solve Problem 2 and provide their complexity.

Information Set Decoding techniques.

The best known algorithms for decoding random linear codes derive from the *Information Set Decoding* technique, first appeared in [Pra62].

Definition 2.11 (Information Set)**.** *Let G be a generator-matrix of a (n, k)-linear code, $\mathcal{I} = \{i_1, \ldots, i_k\}$ be a subset of $\{1, \ldots, n\}$ and $G_{\mathcal{I}}$ be the $k \times k$ submatrix of G defined by the columns of indices $\mathcal{I}$. If $G_{\mathcal{I}}$ is invertible then $\mathcal{I}$ is an information set.*

The matrices $G' = G_{\mathcal{I}}^{-1}G$ and G generate the same code, and for any codeword $c = mG'$, the entries corresponding to the indices in $\mathcal{I}$ contain the information symbols. For this reason, $\mathcal{I}$ is called an *information set*.

Using a parity-check matrix H, an information set $\mathcal{I}$ implies the non-singularity of the block formed by the columns of indices $\mathcal{I}' = \{1, \ldots, n\} \backslash \mathcal{I}$. The description in terms of parity-check matrices, although less intuitive, favors the explanation on how ISD algorithms work. Consider:

- A code $\mathcal{C}$ of parity-check matrix H.
- A word $y = c + e$, where $c \in \mathcal{C}$ and $\mathsf{wt}(e) = w$.

The classical information set decoding [Pra62] works as follows. It computes the syndrome $s^T \leftarrow Hy^T$ and samples a set $\mathcal{I}' = \{i_1, \ldots, i_{n-k}\}$ from $\{1, \ldots, n\}$. Then it computes a matrix $H' = H_{\mathcal{I}'}^{-1}H$, where $H_{\mathcal{I}'}$ is the block of columns of indices $\mathcal{I}'$ in H, and $y' \leftarrow H_{\mathcal{I}'}^{-1}s$. If e has its w non-zero values concentrated in the entries defined by $\mathcal{I}'$, then the syndrome will reveal the error vector. Figure 2.4 illustrates this idea.

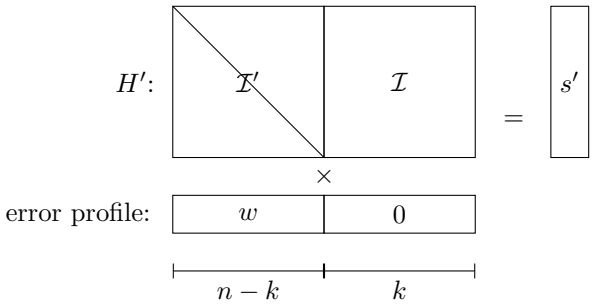

Figure 2.4: Prange Information Set Decoding.

In all ISD variants, the algorithm samples different sets $\mathcal{I}'$, until finding one that satisfies the error profile. We denote the event of finding the sought pattern by E. The algorithm build lists of size L of candidates that might match the error profile. When the algorithm parameters are *optimal*, the cost of these algorithms is $C \approx L/P(E)$, up to a small factor.

In the classical information set decoding, the event of matching the sought error pattern has probability $\Pr(E_{\text{Pra62}}) = \frac{\binom{n-k}{w}}{\binom{n}{w}}$.

Many improvements have been made since then. In [LB88], the authors relaxed the constraint on the positions of the errors, permitting some $p < w$ errors to be out of the positions indexed by $\mathcal{I}'$. Thus, all combinations of p columns are evaluated. Those ones that their sum plus the syndrome vector gives a vector of weight $w - p$ will give a solution. The idea is that other $w - p$ columns from the identity block defined by $\mathcal{I}'$ can be selected to complete the sought error pattern. This changes the probability of finding such an error vector to $\Pr(E_{\text{LB88}}) = \frac{\binom{n-k}{w-p}\binom{k}{p}}{\binom{n}{w}}$.

In [Ste89], Stern proposed a relevant improvement. The first difference consists of considering a 0-window in the last l bits of the error vector. This implies that the sum of the p columns above mentioned must results in a vector with a 0-window in its last l positions. This condition cab be satisfied by considering two sets of $p/2$ columns and finding collisions on these two sets. This approach benefits from the birthday paradox. The leads the probability of success to $\Pr(E_{Ste89}) = \frac{\binom{n-k-l}{w-p}\binom{k/2}{p/2}^2}{\binom{n}{w}}$.

In [FS09], Finiasz and Sendrier improved this attack suppressing the 0-window present in the Stern version. Thus, the probability of success is $\Pr(E_{\text{FS09}}) = \frac{\binom{n-k-l}{w-p}\binom{(k+l)/2}{p/2}^2}{\binom{n}{w}}$.

In summary, the works proposed in [Pra62, LB88, Ste89, FS09] changed the complexity of decoding by considering different error patterns.

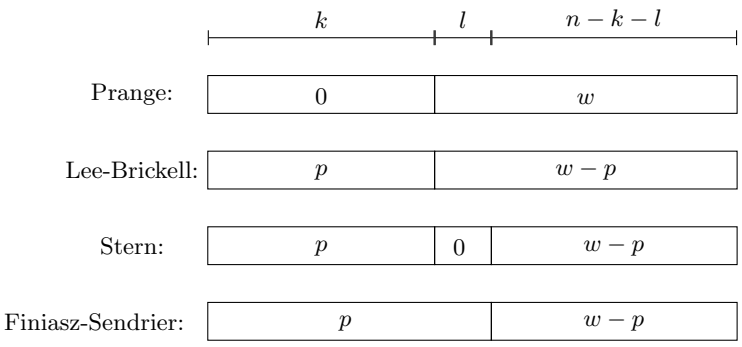

Figure 2.5: Different error-patterns.

The most recent improvements [MMT11, BJMM12] generalize the way to find collisions in the last l bits, of the p columns in $\mathcal{I}$. These modifications change the candidates list size L. They provide slight improvements on the exponential of the cost, but it is worth mentioning that in practice they require an important overhead to process such lists.

In [MMT11] the authors improve ISD by using the representation technique. It exploits the fact that a solution vector can be obtained from the sum of different vectors. For example, a position with value 1 can be obtained by two different ways: $1+0$ and $0+1$. By retaining a single combination for each solution, the authors could demonstrate an asymptotic gain.

Finally in [BJMM12], the authors extends the representation technique. Here, it is also considered that a zero position can be obtained by $0=0+0$ and $0=1+1$, in a finite field of characteristic 2. In [MMT11], it was implicitly restricted to the case $0+0=0$.

We conclude the discussion about ISD techniques citing some *ad-hoc* approaches for particular scenarios. In [Pet10, Pet11], Peters extends the information set decoding technique to codes defined over $\mathbb{F}_q$. In [Sen11], Sendrier studies improvements on solving one instance of the decoding problem, when N instances of the problem are available. It results in a gain by using the birthday-paradox. This variant will be analyzed in the context of MDPC codes, in Section 6.5.2.

Generalized-Birthday Algorithm (GBA).

For some particular parameter sets, another technique is more advantageous than ISD. This is the Generalized-Birthday Algorithm (GBA), which was first proposed in [Wag92] to solve the k-sum problem. The application of the GBA to the context of decoding is due to D. Bleichenbacher appeared in [FS09]. This algorithm has been used in the successful forge valid CFS signatures [FS09].

1. Build 3 lists L_0, L_1, and L_2 of the sum of respectively t_0, t_1 and t_2 columns of H (with $t = t_0 + t_1 + t_2$).
2. Merge the two lists L_0 and L_1 into a list L'_0 of the sum of $t_0 + t_1$ columns of H, keeping only those starting with λ zeros.
3. Repeat the following steps:
   (a) Choose a counter and compute the corresponding document hash,
   (b) Sum this hash with all elements of L_2 matching on the first λ bits,
   (c) Look up each of these sums in L'_0: any complete match gives a valid signature.

The workfactor of this attack is lower bounded in [FS09]. Consider a code of length n, codimension r and error weight w. Then the workfactor to correct w errors using GBA is at least:

$$WF(n,r,w)_{GBA} \geq \frac{r-a}{a} 2^{\frac{r-a}{a}},$$

such that $2^{-a}\binom{n}{\frac{2w}{2^a}} = 2^{\frac{r-a}{a}}$.

Part I

Algebraic Codes

Chapter 3

Introduction

Algebraic codes are the most used linear codes in cryptography. Since 1978, when Robert J. McEliece proposed the first code-based cryptosystem based on binary Goppa codes, they have been studied for cryptographic applications. Posteriorly, many other algebraic codes have been investigated for such a purpose, such as BCH and Alternant codes. In this chapter, we give the definition of some codes that will be useful along the text and describe how some of them have been applied to reduce the keys of code-based cryptosystems. We refer to [MS78] for more details about algebraic codes.

Organization of Chapter 3. In the first part of this chapter, we give the definition of Generalized Reed-Solomon, Alternant and Goppa codes. These codes play a key role in our work. In the second part, we discuss the previous proposals for reducing the key-length of code-based cryptosystems using algebraic codes.

3.1 Some Algebraic Codes

In this section, we give the definition of some codes that will be useful along the text. Some of them are defined in terms of a parity-check matrix, others in terms of a syndrome. These codes are the Generalized Reed-Solomon codes, also known as GRS codes, the Alternant codes and the important family of Goppa codes.

The relationship between these codes is given as follows. Goppa codes form a sub-family of alternant codes, *i.e.* every Goppa code is also an alternant code. Alternant codes are sub-field subcodes of Generalized Reed-Solomon codes. Sub-field subcodes can be produced by different methods. In this section, we describe two constructions for producing sub-field subcodes, namely, the trace and the co-trace construction.

All these codes are decoded in a very similar way. In summary, a polynomial equation is built from the syndrome of the received message and from some particular algebraic structure of the code. When solved, this equation (usually called as the *key-equation* for decoding) provides enough information to find the positions and the values of the errors contained into the received message.

Representing tuples by polynomials over rings. Usually, plaintext, codewords and words are interpreted as vectors with elements in a finite field. An alternative representation uses polynomials over a ring. Let p be a prime number, $q = p^m$ for some positive integer m and $\mathbb{F}_q$ be a finite field of q elements. We denote by $\mathbb{F}_q[x]$ the polynomial ring in $\mathbb{F}_q$, i.e. the ring that consists of polynomials with coefficients in $\mathbb{F}_q$. Consider a vector $f = (f_0, \ldots, f_{k-1}) \in \mathbb{F}_q^k$. Its polynomial representation in the polynomial ring $\mathbb{F}_q[x]$ is

$$f(x) = f_{(k-1)} \cdot x^{(k-1)} + f_{(k-2)} x^{(k-2)} + \cdots + f_1 x + f_0.$$

Polynomials have many features that have been explored in coding theory.

3.1.1 Generalized Reed Solomon Codes

The GRS code family is a remarkable example of codes that uses the polynomial interpretation for achieving efficient decoding.

Definition 3.1 (GRS code)**.** *Let $\mathbb{F}$ be a finite field, $\boldsymbol{v} = \{v_1, \ldots, v_n\} \in \mathbb{F}^n$ a sequence of non-zero elements and $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\} \in \mathbb{F}^n$ a sequence of distinct elements. The (n, k)-generalized Reed-Solomon code defined by $(\boldsymbol{v}, \alpha)$ is*

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

In other words, the code $GRS_{n,k}(\boldsymbol{\alpha}, \mathbf{v})$ is the evaluation of all polynomials $f(x) \in \mathbb{F}_q[x]$ of degree up to k at $\boldsymbol{\alpha}$, scaled by the components of $\mathbf{v}$. This operation can be seen as the product of a vector $\mathbf{f} = (f_1, \ldots, f_k) \in \mathbb{F}_q^k$ by a particular generator matrix for $GRS_{n,k}(\boldsymbol{\alpha}, \mathbf{v})$. This particular generator matrix is called the *canonical generator matrix*, which is the product of a Vandermonde matrix and a diagonal matrix.

Definition 3.2 (Vandermonde matrix)**.** *The Vandermonde matrix of sequence $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ is defined by $V_{ij} = \alpha_i^j$, i.e.*

$$V(\boldsymbol{\alpha}) = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^i & \alpha_2^i & \ldots & \alpha_n^i \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_n^{k-1} \end{pmatrix}$$

Definition 3.3 (Diagonal matrix)**.** *The diagonal matrix of sequence $\boldsymbol{v} = (v_1, \ldots, v_n)$ is defined by $D_{ii} = v_i$ and $D_{ij} = 0$ for $i \neq j$.*

Definition 3.4 (Canonical Generator Matrix of a GRS code)**.** *The* canonical

generator matrix *of a code* $GRS_{n,k}(\boldsymbol{\alpha}, \boldsymbol{v})$ *is:*

$$G = V(\boldsymbol{\alpha})D(\boldsymbol{v}) \tag{3.1}$$

$$= \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^i & \alpha_2^i & \dots & \alpha_n^i \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \dots & \alpha_n^{k-1} \end{pmatrix} \begin{pmatrix} v_1 & 0 & \dots & 0 \\ 0 & v_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & v_i & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & v_n \end{pmatrix}$$

$$= \begin{pmatrix} v_1 & v_2 & \dots & v_n \\ v_1\alpha_1 & v_2\alpha_2 & \dots & v_n\alpha_n \\ \vdots & \vdots & \vdots & \vdots \\ v_1\alpha_1^i & v_2\alpha_2^i & \dots & v_n\alpha_n^i \\ \vdots & \vdots & \vdots & \vdots \\ v_1\alpha_1^{k-1} & v_2\alpha_2^{k-1} & \dots & v_n\alpha_n^{k-1} \end{pmatrix} \tag{3.2}$$

An interesting property of these codes is that it is possible (and convenient) to express the dual of a GRS code as another GRS code of same sequence $\boldsymbol{\alpha}$.

Proposition 3.1. *Let* $\boldsymbol{u}_i = (v_i \prod_{j \neq i}(\alpha_i - \alpha_j))^{-1}$ *and* $r = n - k$. *Then*

$$GRS_{n,k}(\boldsymbol{\alpha}, \boldsymbol{v})^{\perp} = GRS_{n,r}(\boldsymbol{\alpha}, \boldsymbol{u})$$

The main consequence of Proposition 3.1 is that it gives a straightforward way to build the *canonical parity-check matrix* for $GRS_{n,k}(\boldsymbol{\alpha}, \mathbf{v})$.

Definition 3.5 (Canonical Parity-Check Matrix of a GRS code)**.** *The* canonical parity-check matrix *of a code* $GRS_{n,k}(\boldsymbol{\alpha}, \boldsymbol{v})$ *is:*

$$H = V(\boldsymbol{\alpha})D(\boldsymbol{u}) \tag{3.3}$$

$$= \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^i & \alpha_2^i & \dots & \alpha_n^i \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \dots & \alpha_n^{r-1} \end{pmatrix} \begin{pmatrix} u_1 & 0 & \dots & 0 \\ 0 & u_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & u_i & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & u_n \end{pmatrix}$$

$$= \begin{pmatrix} u_1 & u_2 & \dots & u_n \\ u_1\alpha_1 & u_2\alpha_2 & \dots & u_n\alpha_n \\ \vdots & \vdots & \vdots & \vdots \\ u_1\alpha_1^i & u_2\alpha_2^i & \dots & u_n\alpha_n^i \\ \vdots & \vdots & \vdots & \vdots \\ u_1\alpha_1^{r-1} & u_2\alpha_2^{r-1} & \dots & u_n\alpha_n^{r-1} \end{pmatrix} \tag{3.4}$$

where the sequence $\boldsymbol{u}$ *is defined as in Proposition 3.1.*

The canonical parity-check matrix gives a convenient definition for the GRS syndrome vector. More precisely, the syndrome vector corresponding to a vector $\mathbf{c} \in \mathbb{F}_q^n$ is:

$$\mathbf{s}_j = \sum_{i=1}^{n} (c_i u_i \alpha_i^j), \qquad \text{for } 0 \leq j \leq r-1$$

Note that when writing this vector in polynomial form in respect to a variable z, we have:

$$\begin{aligned} S(z) &= \sum_{j=0}^{r-1} \sum_{i=1}^{n} c_i u_i \alpha_i^j z^j \\ &= \sum_{i=1}^{n} c_i u_i \sum_{j=0}^{r-1} (\alpha_i z)^j \end{aligned} \tag{3.5}$$

The sum $\sum_{j=0}^{r-1}(\alpha_i z)^j$ can be replaced by a more convenient expression. This comes from the fact that we are actually interested in $S(z) \bmod z^r$ and that the inverse of $1 - \alpha z$ in $\mathbb{F}_q[z] \bmod z^r$ is:

$$\frac{1}{1-\alpha z} = \sum_{j=0}^{r-1} (\alpha z)^j \bmod z^r$$

Thus, by replacing $\sum_{j=0}^{r-1}(\alpha_i z)^j$ by $\frac{1}{1-\alpha z}$ into 3.5, we obtain a polynomial definition of the GRS syndrome, which is more convenient for decoding.

Definition 3.6 (Syndrome of GRS codes)**.** *Let* $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$*. The syndrome of* $\boldsymbol{c}$ *in respect to the code* $GRS_{n,k}(\boldsymbol{\alpha}, \boldsymbol{v})$ *over* $\mathbb{F}_q$ *is*

$$S_c(z) = \sum_{i=1}^{n} \frac{c_i u_i}{1 - \alpha_i x} \mod z^r$$

where $u_i^{-1} = v_i \prod_{j \neq i}(\alpha_i - \alpha_j)$*.*

3.1.2 Alternant Codes

Informally, an alternant code is a subfield subcode of a GRS code. It consists of all codewords with components in the basefield. Let $q = p^m$ be a prime power.

Definition 3.7 (Alternant code)**.** *Let* $\boldsymbol{v} = \{v_1, \ldots, v_n\} \in \mathbb{F}_q^n$ *be a sequence of non-zero elements and* $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\} \in \mathbb{F}_q^n$ *be a sequence of distinct elements. The alternant code* $\mathcal{A}_{n,k}(\boldsymbol{\alpha}, \boldsymbol{v})$ *consists of all* $c \in \mathbb{F}_p^n$ *such that* $Hc^t = 0$*, where* $H \in \mathbb{F}_q^{r \times n}$ *is given by 3.4.*

It is possible and convenient to use a parity-check matrix defined over $\mathbb{F}_p$, instead of the one defined over $\mathbb{F}_q$ given by 3.4. Below, we discuss two approaches for this construction.

Trace construction. The trace construction defines a parity-check matrix $\tilde{H} \in \mathbb{F}_p^{mr \times n}$ from $H \in \mathbb{F}_q^{r \times n}$ by replacing each element of H by the corresponding column vector of length m from $\mathbb{F}_q$. Therefore each column of H will produce r vectors of length m. This is the simplest construction for subfield subcodes.

Co-trace construction. The co-trace construction defines a parity-check matrix $\tilde{H}' \in \mathbb{F}_p^{mr \times n}$ from $H \in \mathbb{F}_q^{r \times n}$ by writing the coefficients of equal degrees of all components of H onto a column vector of length r. Therefore each column of H will produce m vectors of length r. This construction will present some advantages in comparison to the trace construction for storing particular types of matrices (e.g. Quasi-Dyadic Goppa codes, see Chapter 4).

3.1.3 Goppa Codes

Goppa codes are one of the most important code families for cryptography. They are defined by a Goppa polynomial $g(z)$ and a sequence $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\} \in \mathbb{F}_q^n$ such that $g(\alpha_i) \neq 0$ for $1 \leq i \leq n$. Next, the Goppa syndrome definition.

Definition 3.8 (Goppa syndrome)**.** *Let $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{F}_p^n$. The syndrome of $\boldsymbol{c}$ in respect to the code $\Gamma(\boldsymbol{\alpha}, g)$ over $\mathbb{F}_p$ is*

$$S_c(z) = \sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} \bmod g(z)$$

A straightforward definition for Goppa codes is the set of all vectors $c \in \mathbb{F}_p^n$, such that $S_c(z) = 0 \bmod g(z)$. However, Goppa codes are subfield subcodes of GRS codes and therefore they must admit a similar definition to 3.7. Indeed, a Goppa code $\Gamma(\boldsymbol{\alpha}, g)$ is the restriction to $\mathbb{F}_p$ of a $GRS_{n,k}(\boldsymbol{\alpha}, \mathbf{v})$ code, where $\mathbf{v} = \{g(\alpha_1)^{-1}, \ldots, g(\alpha_n)^{-1}\} \in \mathbb{F}_q^n$.

Definition 3.9 (Goppa code)**.** *Let $q = p^m$ be a prime power. Let $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\} \in \mathbb{F}_q^n$ be a sequence of distinct elements, $g(z) \in \mathbb{F}_q[z]$ be a polynomial such that $g(\alpha_i)_{1 \leq i \leq n} \neq 0$ and $\boldsymbol{v} = \{g(\alpha_1)^{-1}, \ldots, g(\alpha_n)^{-1}\} \in \mathbb{F}_q^n$. The Goppa code $\Gamma(L, g)$ consists of all $c \in \mathbb{F}_p^n$ such that $Hc^t = 0$, where $H \in \mathbb{F}_q^{r \times n}$ is given by 3.4.*

Remark 3. If the Goppa polynomial is irreducible, then it is called an *irreducible* Goppa code.

Remark 4. The minimum distance of a Goppa code $\Gamma(L, g)$ is $d_0 \geq deg(g) + 1$.

Remark 5. If $g(z)$ has no multiple zeros then $\Gamma(L, g) = \Gamma(L, g^2)$. This particularly implies that the minimum distance of $\Gamma(L, g) \geq 2\ deg(g) + 1$.

As discussed for Alternant codes, Goppa codes admit parity-check matrices with elements in the base field $\mathbb{F}_p$, instead of in $\mathbb{F}_q$. Both Trace and Co-trace construction (see 3.1.2) are valid for building such a parity-check matrix.

3.2 Previous Algebraic Proposals for Key Size Reduction

In this section, we discuss about two attempts to reduce the key-length of code-based cryptography using algebraic codes. The first one was proposed by Gaborit using quasi-cyclic BCH codes [Gab05]. The second attempt was proposed

by Berger *et al.* using quasi-cyclic alternant codes [BCGO09]. Finally we show why these proposals failed to achieve security against structural attacks.

Circulant Matrices

Circulant matrices are the central point in many attempts to reduce the key-size of code-based cryptosystems. They provide both efficient representation and algorithmic complexity.

Definition 3.10 (Circulant Matrix)**.** *A matrix $M \in \mathbb{F}^{p\times p}$ is called circulant if its rows are obtained by cyclically shifting its first row.*

$$M = \begin{pmatrix} m_0 & m_1 & \dots & m_{p-1} \\ m_{p-1} & m_0 & \dots & m_{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ m_1 & m_2 & \dots & m_0 \end{pmatrix}$$

Circulant matrices are *closed* under product and sum, i.e. these operations preserve cyclicity. Moreover a circulant matrix is completely defined by a single vector, for example its first row or column. In the previous section, we introduced the polynomial representation of tuples, which can also be used for circulant matrices. Thus the above matrix can be described by the polynomial

$$m(x) = m_0 + m_1 x + m_2 x^2 + ...m_{p-1}x^{p-1} \in \mathbb{F}[x].$$

The i-th row of a circulant matrix can be obtained by $x^i \cdot m(x) \bmod (x^p - 1)$. Moreover the product of a vector $v \in \mathbb{F}^p$, with polynomial representation $v(x) \in \mathbb{F}[x]$, by a circulant matrix can be computed by $v(x) \cdot m(x) \bmod (x^p - 1)$. Finally the product of two circulant matrices $M_1 \in \mathbb{F}^{p\times p}$, $M_2 \in \mathbb{F}^{p\times p}$, with polynomial representation $m_1(x)$, $m_2(x) \in \mathbb{F}[x]$ respectively, can be computed by $m_1(x) \cdot m_2 \bmod (x^p - 1)$. These properties are formally described in Proposition 3.2.

Proposition 3.2. *Circulant matrices of size $p \times p$ with elements in $\mathbb{F}$ are isomorphic to polynomials in $\mathbb{F}[x]/(x^p - 1)$.*

Quasi-Cyclic and Cyclic Codes

Quasi-cyclic and cyclic codes are linear codes that benefit from the features of circulant matrices. Thus linear codes that have efficient decoding procedures and a non-empty intersection with cyclic or quasi-cyclic codes are natural candidates for achieving efficient code-based cryptosystems.

Definition 3.11 (Quasi-Cyclic Codes)**.** *A linear code is quasi-cyclic of order p and index n_0 iff any cyclic shift of a codeword by n_0 coordinates is again a codeword. Moreover, these codes admit both generator and parity-check matrices composed by circulant blocks of size $p \times p$.*

A linear code with $n = n_0 p$, $k = k_0 p$ and $r = n - k = (n_0 - k_0)p = r_0 p$ is quasi-cyclic and admit a $k_0 \times n_0$ generator matrix G such that

$$G = \begin{pmatrix} G_{1,1} & G_{1,2} & \dots & G_{1,n_0} \\ G_{2,1} & G_{2,2} & \dots & G_{2,n_0} \\ \vdots & \vdots & \ddots & \vdots \\ G_{k_0,1} & G_{k_0,2} & \dots & G_{k_0,n_0} \end{pmatrix}$$

and a $r_0 \times n_0$ parity-check matrix H such that

$$H = \begin{pmatrix} H_{1,1} & H_{1,2} & \dots & H_{1,n_0} \\ H_{2,1} & H_{2,2} & \dots & H_{2,n_0} \\ \vdots & \vdots & \ddots & \vdots \\ H_{r_0,1} & H_{r_0,2} & \dots & H_{r_0,n_0} \end{pmatrix}$$

where each block $G_{i,j}$ and $H_{i,j}$ is a $p \times p$ circulant block.

Definition 3.12 (Cyclic Codes)**.** *Cyclic code is a quasi-cyclic code with* $n_0 = 1$.

Remark 6. A cyclic code of length n is a quasi-cyclic code of any order which divides n.

An equivalent definition for cyclic codes is: a cyclic-code $\mathcal{C}$ of length n is an ideal of the ring $\mathbb{F}[x]/(x^n - 1)$. Every cyclic code is associated to a polynomial $g(x)$ of degree r which divides $(x^n - 1)$ called its *generator polynomial*. Any codeword $c(x) \in \mathcal{C}$ of this code can be obtained by $c(x) = m(x)g(x)$, where $m(x) \in \mathbb{F}[x]$ of degree at most $n - r - 1$. Thus a generator-matrix for this code is built using the coefficients of $g(x)$ in its first-row.

3.2.1 Using Quasi-Cyclic BCH Codes

In [Gab05], Gaborit proposed the use of Bose-Chaudhuri-Hocquenghem (BCH) codes to instantiate the McEliece cryptosystem. Below, we give the definition of these codes, we present the scheme and we explain why it fails at achieving security against structural attacks.

BCH Codes

BCH codes are cyclic codes and therefore can be described in terms of a generator polynomial. Let $q = p^m$ be a prime power and α be a primitive n-th root of unity, i.e. α is a root of the polynomial $(x^n - 1)$ over $\mathbb{F}_q$.

Definition 3.13. *An* (n, k)*-BCH code* $\mathcal{C}$ *over* $\mathbb{F}_q$ *is a cyclic code with generator polynomial* $g(x)$ *such that for some integers* $b \geq 0$, $\delta \geq 1$,

$$g(\alpha^b) = g(\alpha^{b+1}) = \cdots = g(\alpha^{b+\delta-2}) = 0.$$

A primitive BCH code is a BCH code with length $n = q - 1$.

BCH codes, as defined above, have minimum distance at least δ and therefore allow the correction of, at most, $\lfloor \frac{\delta-1}{2} \rfloor$ errors. There are many decoding algorithms for BCH codes and they usually proceed as briefly described next. The syndrome of the received word is computed, which permits the computation of the *error locator polynomial* and the *error evaluator polynomial*. The roots of these polynomials provide the position and the value of the errors, respectively. The decoding algorithm solves an equation to determine such polynomials and then it corrects the errors.

The QC-BCH McEliece Variant [Gab05]

The idea proposed by Gaborit is to pick a primitive BCH code, which is originally cyclic, and then produce a quasi-cylic subcode by using the following method. Let $\mathcal{C}_0$ be a cyclic code of length $n = pn_0$ and dimension $k = pk_0$. Let $\{c_1, c_2, \ldots, c_{k_0-1}\}$ be random codewords of $\mathcal{C}_0$. Consider the code $\mathcal{C} = S_{n_0}(c_1) + \cdots + S_{n_0}(c_{k_0-1})$, where S_{n_0} is the symmetric group of order n_0. We assume that $\mathcal{C}$ is a code of dimension $k - p = p(k_0 - 1)$. Any vector c_i can be seen as vector $(c_{i,0}, \ldots, c_{i,n_0-1})$, where each $c_{i,j}$ can be seen as an element of $\mathbb{F}_p[x]/(x^p - 1)$. Thus $\mathcal{C}$ is a quasi-cyclic code that admits the generator matrix

$$G = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n_0} \\ \vdots & & \vdots \\ c_{k_0-1,1} & \cdots & c_{k_0-1,n_0} \end{pmatrix}.$$

This code can be efficiently decoded and thus represents the private-key of the McEliece cryptosystem. The public-key must allow encoding and therefore should be closely related to the original private code. A usual approach is to have the public code $\mathcal{C}$ and the private code $\mathcal{C}'$ as permutation-equivalent codes. In other words, there exists a permutation that maps codewords from $\mathcal{C}$ into codewords of $\mathcal{C}'$. This is equivalent to the existence of a permutation matrix $\Pi \in \mathbb{F}_q^{k \times k}$ such that $G' = \Pi G$, where $G' \in \mathbb{F}_q^{k \times n}$ is a generator-matrix for $\mathcal{C}'$. Note however that to maintain the quasi-cyclicity in G', some restrictions are required to the permutation Π. For such purpose, Gaborit suggests Π as

$$\Pi = \begin{pmatrix} \pi & & & \\ & \pi & & \\ & & \ddots & \\ & & & \pi \end{pmatrix}, \tag{3.6}$$

where $\pi \in \mathbb{F}_q^{\frac{k}{n_0} \times \frac{k}{n_0}}$ is a permutation block. This leads to a public generator matrix of the form

$$G' = \begin{pmatrix} c_{1,\pi(1)} & \cdots & c_{1,\pi(n_0)} \\ \vdots & & \vdots \\ c_{k_0-1,\pi(1)} & \cdots & c_{k_0-1,\pi(n_0)} \end{pmatrix}.$$

The author suggests parameters for 80 bits of security, with public-key sizes of 40505 bits, and for 90 bits of security, with public-key sizes of 12302 bits.

Claimed security	n	Number of errors	Size of the cyclic block
2^{80}	2047	31	23
2^{90}	4095	26	45

Table 3.1: Parameters of the QC-BCH Variant.

Attacking the QC-BCH McEliece Variant

Two things make the scheme [Gab05] vulnerable against structural attacks, as explained in [OTD10].

1. The code used in the scheme is derived from a primitive BCH code.

For all parameters suggested by the author, the number of possible primitive BCH codes is limited to at most just a few hundreds. Since an adversary is able to enumerate all of them, in practice the private code must be considered as a public information. For this reason, the security of the scheme relies solely on the unknown permutation that maps the private and public codes. Unfortunately, the permutation also has security flaws, leaking private information.

2. The permutation Π as presented in 3.6 is too restrictive.

The permutation Π actually relies only on the smaller permutation block π. This enables adversaries to build and solve an over-constrained linear system, revealing the secret permutation.

3.2.2 Using Quasi-Cyclic Alternant Codes

Rather than considering subcodes of a BCH code, the scheme proposed in [BCGO09] suggests the use of subfield subcodes of generalised Reed-Solomon codes, also known as Alternant codes. We defined Alternant codes in section 3.1.2. The authors suggest the following method to generate the codes.

1. Generate a Reed-Solomon code, which is a cyclic code.

2. Using shortening, scaling and block-permuting transformations, produce a Generalized-Reed-Solomon code which is quasi-cyclic.

3. Take its subfield sub-code, which results in an alternant quasi-cyclic code.

Parameters for different security levels and key sizes are suggested. Let q_0 be the size of the finite field of the original code, q be the size of the subfield where the public code is defined, N_0 the number of original blocks, n_0 is the number of blocks after shortening the code, ℓ the order of the quasi-cyclic code (i.e. the size of each circulant block), t the number of errors and n, k are the usual code parameters. Table 3.2 presents the parameters suggested by the authors. The key-size column refers to the public key-size in bits and the security column to the logarithm in base 2 of the expected number of steps necessary to break the scheme.

q_0	ℓ	N_0	t	n	k	q	n_0	security	key-size
2^{16}	51	1285	25	663	561	2^8	13	80	8980
2^{16}	51	1285	37	663	510	2^8	13	95	12240
2^{16}	51	1285	37	1020	867	2^8	20	116	20800
2^{16}	85	771	42	1105	935	2^4	13	83	14960
2^{20}	93	11275	23	744	651	2^{10}	8	80	6510
2^{20}	93	11275	46	744	558	2^{10}	8	107	11160
2^{20}	75	11275	36	750	675	2^{10}	10	100	12120
2^{20}	165	6355	40	1320	1155	2^5	8	90	11480

Table 3.2: Parameters of the QC-Alternant variant.

Attacking the QC-Alternant McEliece variant

The attack presented in [FOPT10a] succeeded in break all the parameters proposed in [BCGO09]. This attack is explained in Section 4.4.3. Briefly, it consists of setting up and solving the system generated by $GH^T = 0$, where G is the public generator matrix and H is an unknown alternant parity-check matrix. The system can be described as

$$\left\{ g_{i,0} Y_0 X_0^j + \cdots + g_{i,n-1} Y_{n-1} X_{n-1}^j = 0 | i \in \{0, \ldots, k-1\}, j \in \{0, \ldots, r-1\} \right\} \tag{3.7}$$

This system is also valid for the classical McEliece cryptosystem instantiated with binary Goppa codes. However the usual parameters prevent the effectiveness of this approach.

Regarding the QC-Alternant variant, the scenario is much better for adversaries. In short, two features makes possible to solve this system.

1. The system is bihomogeneous and bilinear.

2. The quasi-cyclic structure allows a drastic reduction of the number of unknowns.

This kind of attack is exponential in nature and can be easily prevented by choosing more conservative parameters. Note that this attack is also effective against the preliminary version of the quasi-dyadic Goppa proposal [MB09b], where the public code was defined over an intermediary field (see discussions in Sections 4.1.4 and 4.4.3).

3.2.3 Lessons Learned

The two previous approaches have an important impact on our contributions related to algebraic codes, namely the p-adic Goppa proposal.

The first one, based on QC-BCH codes, presented a new trend on code-based cryptography by using the quasi-cyclicity of some linear codes to achieve compact keys for code-based cryptosystems. On the other hand, this work warned us about two possible menaces against compact-keys code-based variants. The first one refers to the importance of choosing a large code-family. When the code family is not large enough, adversaries have the advantage of knowing the private code, since they can exhaustive examine each possible private code. This notably improves the possibilities of attacks. The second possible threat refers to the mapping between public and private code. Very constrained permutations might permit the setup of system of equations, which can be used to break the scheme.

The second approach, based on QC-alternant codes, presented an interesting and apparently effective way to hide private information into the public code. This is done by using punctured permuted subfield sub-codes. This technique is also used in our construction. On the other hand, this proposal also suffered from using intermediary fields for the public code. This particularity led to the effectiveness of algebraic structural attacks.

As a last remark, none of these proposals have used the reputable family of Goppa codes and both of them were broken. This code-family, especially its binary case, has been used to instantiate the McEliece cryptosystem for more

than thirty years, remaining almost unscathed from practical attacks. Thus it seems prominent to investigate the possibility of achieving compact keys using the lessons learned in previous attempts without leaving the Goppa family, as in the classical McEliece setup.

Chapter 4

p-adic Goppa Codes

The McEliece cryptosystem was originally built upon the class of binary Goppa codes, which remains secure to this date but leads to very large public keys. Many other code-families have been analyzed for this code-based cryptosystem, for different purposes. Unfortunately, the majority of them have been subsequently broken, highlighting the possibly strengthened security of Goppa codes.

One of the main reasons to replace Goppa codes by other code families refers to reduce its key-size. The idea is to use codes that admit compact representation. Natural candidates for this approach are codes that admit generator and parity-check matrices in quasi-cyclic form. For example, [Gab05] and [BCGO09] suggested quasi-cyclic BCH and alternant codes, respectively. As discussed in Chapter 3.2, the algebraic structure of these codes combined with the quasi-cyclicity permitted to successfully attack all these variants.

In this chapter, we describe a simple way to significantly reduce the key size of McEliece and related cryptosystems without leaving the Goppa family. More precisely, we propose a subclass of Goppa codes which admits compact representation.

The initial property that allowed our construction comes from Theorem 4.1, where it is proven that certain Goppa codes admit parity-check matrices in Cauchy form. Matrices in Cauchy form admit very efficient representation (occupying linear space, instead of quadratic), but they are not directly suitable to code-based cryptosystems. For example, the matrix-product operation, required by McEliece-like cryptosystems, do not preserve the Cauchy structure.

An alternative consists of finding a family of matrices that has a non-empty intersection with Cauchy matrices (thus useful for representing Goppa codes), which also enjoys from compact representation and preserves its structure after manipulations. Matrices called *p-adic* matrices have a particular structure and are excellent candidates for this purpose. The index p comes from the fact that they are defined over a finite field of characteristic p. Similarly to cyclic matrices, they are completely defined by their first row/column and closed under matrix products. Moreover, they have a non-empty intersection with Cauchy matrices.

Note that p-adic matrices cannot be directly applied in code-based cryptosystems. The overly constrained algebraic structure would be easily recovered by adversaries. Thus, strategies to hide such structure are required. To address this issue, we followed the techniques proposed in [BCGO09] to disguise codes, exploiting Wieschebrink's theorem on the NP-completeness of distinguishing

punctured codes. The result is the family of *Quasi-p-adic Goppa codes*.

These codes benefit from very compact parity-check or generator matrices leading to McEliece-type keys that are up to a factor $t = \tilde{O}(n)$ smaller than keys produced from generic t-error correcting Goppa codes of length n. Moreover, the complexity of all typical cryptographic operations become $\tilde{O}(n)$. Specifically, under the common cryptographic setting $t = O(n/\lg n)$, code generation, encryption and decryption all have asymptotic complexity $O(n \lg n)$.

This approach first appeared regarding a particular case of p-adic matrices, called dyadic matrices, which corresponds to $p = 2$. The generalization to p-adic matrices for $p > 2$ came afterwards. Since we judge the dyadic case more illustrative, we start the discussion by the dyadic case and then we generalize it. An important remark regarding the dyadic case is that initially we have considered non-binary Goppa codes in dyadic form. This strategy has been shown insecure due to [FOPT10a] (as in [BCGO09]). For this reason, non-binary dyadic codes were not considered for the final, published version of this work.

Organization of Chapter 4. The remainder of this chapter is organized as follows. In Section 4.1, we present the *Quasi-Dyadic Goppa codes* and show how to build them. In Section 4.1.4, we explain the preliminary version of the quasi-dyadic proposal. In Section 4.1.5, we extend this construction to the CFS digital signature scheme. In Section 4.2, we discuss the general case (for codes over $\mathbb{F}_{p\geq 2}$), which leads to *Quasi-p-adic Goppa codes*. In Section 4.3.1, we consider the algorithmic complexity and, in Section 4.3.2, the storage complexity of our proposal. The security issues are discussed in Chapter 4.4. To conclude, we provide guidelines on how to choose parameters in Section 4.5.

Publications presented in Chapter 4. The first version of this work (considering the attacked non-binary Quasi-Dyadic Goppa codes) has been published as a pre-print in:

1. [MB09b]: Rafael Misoczki and Paulo S. L. M. Barreto. Compact McEliece keys from Goppa codes. Cryptology ePrint Archive, Report 2009/187, 2009. http://eprint.iacr.org/2009/187.

The final, published version of the Quasi-Dyadic Goppa codes appeared in:

2. [MB09a]: Rafael Misoczki and Paulo S. L. M. Barreto. Compact McEliece keys from Goppa codes. Proceedings of the Selected Areas in Cryptography, pages 376–392, 2009.

The extension of QD-Goppa codes to CFS signature scheme appeared in:

3. [BCMN10]: Paulo S. L. M. Barreto, Pierre-Louis Cayrel, Rafael Misoczki, and Robert Niebuhr. Quasi-dyadic CFS signatures. Proceedings of the 6th China International Conference on Information Security and Cryptology (Inscrypt 2010), LNCS, pages 336–349. Springer, 2010.

The extension to codes over characteristics greater than 2 appeared in:

4. [BLM11]: Paulo S. L. M. Barreto, Richard Lindner, and Rafael Misoczki. Monoidic codes in cryptography. In the Proceedings of the 4th Post-Quantum Cryptography, volume 7071 of Lecture Notes in Computer Science, pages 179–199. Springer Berlin Heidelberg, 2011.

4.1 The Dyadic Case ($p = 2$)

Dyadic matrices are p-adic matrices defined over finite fields of characteristic $p = 2$. p-adic matrices are defined for $p \geq 2$ and therefore encompass dyadic ones. We start by the dyadic case since it is the most illustrative case.

4.1.1 Preliminaries

In this section, we define the different types of matrices needed by our proposal: dyadic, quasi-dyadic and Cauchy matrices. In what follows, all vector/matrix indices are numbered from zero onwards.

Definition 4.1 (Dyadic Matrices)**.** *Given a commutative ring $\mathcal{R}$ and a vector $h = (h_0, \ldots, h_{n-1}) \in \mathcal{R}^n$, the* dyadic *matrix $\Delta(h) \in \mathcal{R}^{n \times n}$ is the symmetric matrix with components $\Delta_{ij} = h_{i \oplus j}$, where $\oplus$ stands for bitwise exclusive-or on the binary representations of the indices. The sequence h is called its* signature*.*

One can recursively characterize dyadic matrices when n is a power of 2. Any 1×1 matrix is dyadic and for $k > 0$ any $2^k \times 2^k$ dyadic matrix M has the form

$$M = \begin{bmatrix} A & B \\ B & A \end{bmatrix}$$

where A and B are $2^{k-1} \times 2^{k-1}$ dyadic matrices. Figure 4.1 illustrates this behavior.

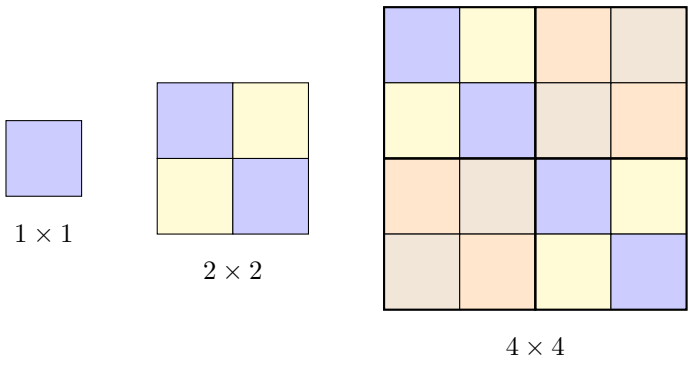

Figure 4.1: Illustration of dyadic matrices of size 1×1, 2×2 and 4×4.

Remark 7. The signature of a dyadic matrix coincides with its first row.

Remark 8. Dyadic matrices form a commutative subring of $\mathcal{R}^{n \times n}$ as long as $\mathcal{R}$ is commutative [Gul73].

We denote by $\Delta(\mathcal{R}^n)$ the set of dyadic $n \times n$ matrices over $\mathcal{R}$ and by $\Delta(t, h)$ the matrix $\Delta(h)$ truncated to its first t rows, for a given integer $t > 0$. Along the text, we will be mainly concerned with the case $\mathcal{R} = \mathbb{F}_q$, the finite field with q (a prime power) elements. A particularly interesting type of dyadic matrices consists of dyadic permutations.

Definition 4.2. *A* dyadic permutation *is a dyadic matrix* $\Pi^i \in \Delta(\{0,1\}^n)$ *whose signature is the* i*-th row of the identity matrix.*

Note that a dyadic permutation is an involution, *i.e.* $(\Pi^i)^2 = I$.

Remark 9. The i-th row of the dyadic matrix defined by a signature h can be written $\Delta(h)_i = h\Pi^i$.

Now we introduce the quasi-dyadic matrices, which will be effectively used in our approach and that are derived from dyadic matrices.

Definition 4.3 (Quasi-Dyadic Matrices)**.** *A* quasi-dyadic *matrix is a (possibly non-dyadic) block matrix whose component blocks are dyadic submatrices.*

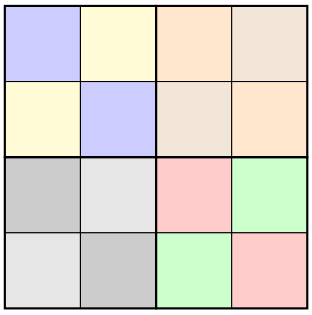

Figure 4.2: A (4×4)-quasi-dyadic matrix composed by (2×2)-dyadic blocks.

These matrices are defined by the first row of their component blocks, and not only by the first row of the whole matrix. In terms of storage, this is no as optimal as dyadic matrices, but still provides an important gain. For example, quasi-dyadic matrices of size $k \times n$ composed by dyadic blocks of size $p \times p$ only need $\frac{k}{p}n$ elements to define it, *i.e.* only k/p rows of length n. Therefore, dyadic and quasi-dyadic matrices have compact representation. However note that they have no direct connection with Goppa codes. To address this issue, we will need Cauchy matrices.

Definition 4.4. *Given two disjoint sequences* $z = (z_0, \ldots, z_{t-1}) \in \mathbb{F}_q^t$ *and* $L = (L_0, \ldots, L_{n-1}) \in \mathbb{F}_q^n$ *of distinct elements, the* Cauchy matrix $C(z, L)$ *is the* $t \times n$ *matrix with elements* $C_{ij} = 1/(z_i - L_j)$, i.e.

$$C(z,L) = \begin{bmatrix} \frac{1}{z_0 - L_0} & \cdots & \frac{1}{z_0 - L_{n-1}} \\ \vdots & \ddots & \vdots \\ \frac{1}{z_{t-1} - L_0} & \cdots & \frac{1}{z_{t-1} - L_{n-1}} \end{bmatrix}$$

Remark 10. Any sub-matrix of a Cauchy matrix is nonsingular [Sch59].

As a side note, Cauchy and dyadic matrices have previously appeared in the cryptology literature in the context of symmetric block ciphers [Yan90, YMT97].

4.1.2 Construction

A property of Goppa codes that is fundamental to our proposal is that they admit a parity-check matrix in Cauchy form.

Theorem 4.1 ([TZ75])**.** *The Goppa code generated by a monic polynomial $g(x) = (x - z_0) \dots (x - z_{t-1})$ without multiple zeros admits a parity-check matrix of the form $H = C(z, L)$,* i.e. *$H_{ij} = 1/(z_i - L_j)$, $0 \leqslant i < t$, $0 \leqslant j < n$.*

This theorem (also appearing in [MS78, Ch. 12, §3, Pr. 5]) is entirely general when one considers the factorization of the Goppa polynomial over its splitting field, in which case a single root of g is enough to completely characterize the code. We will restrict our attention to the case where all roots of that polynomial are in the field $\mathbb{F}_q$ itself.

In general, Cauchy matrices are not dyadic and vice-versa. However, the intersection of these two classes is not always empty. Below we present a method to construct codes on this intersection. This construction is derived from the following Theorem 4.2, which characterizes all Cauchy matrices in dyadic form.

Theorem 4.2. *Let $H \in \mathbb{F}_q^{n \times n}$ with $n > 1$ be simultaneously a dyadic matrix $H = \Delta(h)$ for some $h \in \mathbb{F}_q^n$ and a Cauchy matrix $H = C(z, L)$ for two disjoint sequences $z \in \mathbb{F}_q^n$ and $L \in \mathbb{F}_q^n$ of distinct elements. Then $\mathbb{F}_q$ is a field of characteristic 2, h satisfies*

$$\frac{1}{h_{i \oplus j}} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}, \tag{4.1}$$

and $z_i = 1/h_i + \omega$, $L_j = 1/h_j + 1/h_0 + \omega$ for some $\omega \in \mathbb{F}_q$.

Proof. The proof is done using the following remarks.

1. Dyadic matrices are symmetric. Thus, the sequences that define it must satisfy $1/(z_i - L_j) = 1/(z_j - L_i)$, hence $L_j = z_i + L_i - z_j$ for all i and j.

2. Then $z_i + L_i$ must be a constant α, and taking $i = 0$ in particular this simplifies to $L_j = \alpha - z_j$.

3. Substituting back into the definition $M_{ij} = 1/(z_i - L_j)$ one sees that $H_{ij} = 1/(z_i + z_j + \alpha)$.

4. But dyadic matrices also have constant diagonal, namely, $H_{ii} = 1/(2z_i + \alpha) = h_0$. This is only possible if all z_i are equal (contradicting the definition of a Cauchy matrix), or else if the characteristic of the field is 2.

5. In this case we see that $\alpha = 1/h_0$, and hence $H_{ij} = 1/(z_i + z_j + 1/h_0)$.

6. Plugging in the definition $H_{ij} = h_{i \oplus j}$ we get $1/H_{ij} = 1/h_{i \oplus j} = z_i + z_j + 1/h_0$, and taking $j = 0$ in particular this yields $1/h_i = z_i + z_0 + 1/h_0$, or simply $z_i = 1/h_i + 1/h_0 + z_0$.

7. Substituting back one obtains $1/h_{i \oplus j} = z_i + z_j + 1/h_0 = 1/h_i + 1/h_0 + z_0 + 1/h_j + 1/h_0 + z_0 + 1/h_0 = 1/h_i + 1/h_j + 1/h_0$, as expected.

8. Finally, define $\omega = 1/h_0 + z_0$ and substitute into the derived relations $z_i = 1/h_i + 1/h_0 + z_0$ and $L_j = \alpha - z_j$ to get $z_i = 1/h_i + \omega$ and $L_j = 1/h_j + 1/h_0 + \omega$, as desired.

□

Therefore all we need is a method to solve Equation 4.1. The technique we propose consists of simply choosing distinct nonzero h_0 and h_i at random where i scans all powers of two smaller than n, and setting all other values as

$$h_{i+j} \leftarrow \frac{1}{\frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}}$$

for $0 < j < i$ (so that $i + j = i \oplus j$), as long as this value is well-defined.

For convenience, we introduce the concept of *essence* of the sequence h. It comes from the fact that the values of h_0 and h_i (for i powers of two smaller than n) completely define all others. Then, the essence η of h is:

$$\begin{aligned} \eta_s &= 1/h_{2^s} + 1/h_0, \qquad \text{for } s = 0, \ldots, \lceil \lg n \rceil - 1 \\ \eta_{\lceil \lg n \rceil} &= 1/h_0 \end{aligned}$$

such that

$$1/h_i = \eta_{\lceil \lg n \rceil} + \sum_{k=0}^{\lceil \lg n \rceil - 1} i_k \eta_k, \qquad \text{for } i = \sum_{k=0}^{\lceil \lg n \rceil - 1} i_k 2^k$$

Algorithm 1 captures this idea. The notation $u \xleftarrow{\$} U$ means that variable u is uniformly sampled at random from set U.

From Dyadic to Quasi-Dyadic Goppa Codes

A cryptosystem cannot be securely defined on a Goppa code specified directly by a parity-check matrix in Cauchy form, since this would immediately reveal the Goppa polynomial $g(x)$: it suffices to solve the overdefined linear system $z_i - L_j = 1/H_{ij}$ consisting of tn equations in $t + n$ unknowns. Therefore, we need a way to hide the Goppa structure.

We adopt the same technique as proposed in [BCGO09] to hide cyclic codes, namely, working with a block-shortened subfield subcode of a very large code. This idea is also built upon the work of Wieschebrink [Wie06] who proved that deciding whether a code is equivalent to a shortened code is an NP-complete problem. Below we describe how we implemented this approach.

Initially, we construct a subfield subcode of the original code. As discussed in Chapter 3, a subfield subcode can be obtained from either trace or co-trace construction. In our case, we have to preserve the dyadic symmetry in this basefield construction and the usual trace construction leads to a generator of the dual code that most probably violates it. In contrast, by using the co-trace construction, one can view H as a superposition of $m = [\mathbb{F}_q : \mathbb{F}_2]$ distinct binary dyadic matrices. Each of them can be stored in a separate dyadic signature.

Then we proceed puncturing and block permuting the code. The principle to follow here is to *select and permute* the columns of the original parity-check

Algorithm 1 Constructing a binary Goppa code in dyadic form

INPUT: q (a power of 2), $n \leqslant q/2$, t.
OUTPUT: Support L, generator polynomial g, dyadic parity-check matrix H for a binary Goppa code $\Gamma(L, g)$ of length n and design distance $2t + 1$ over $\mathbb{F}_q$, and the essence η of the signature of H.

$U \leftarrow \mathbb{F}_q \setminus \{0\}$
▷ Choose the dyadic signature $(h_0, \ldots, h_{n-1})$. N.B. Whenever h_j with $j > 0$ is taken from U, so is $1/(1/h_j + 1/h_0)$ to prevent a potential spurious intersection between z and L.
$h_0 \xleftarrow{\$} U$
$\eta_{\lceil \lg n \rceil} \leftarrow 1/h_0$
$U \leftarrow U \setminus \{h_0\}$
for $s \leftarrow 0$ **to** $\lceil \lg n \rceil - 1$ **do**
    $i \leftarrow 2^s$
    $h_i \xleftarrow{\$} U$
    $\eta_s \leftarrow 1/h_i + 1/h_0$
    $U \leftarrow U \setminus \{h_i,\ 1/(1/h_i + 1/h_0)\}$
    for $j \leftarrow 1$ **to** $i - 1$ **do**
        $h_{i+j} \leftarrow 1/(1/h_i + 1/h_j + 1/h_0)$
        $U \leftarrow U \setminus \{h_{i+j},\ 1/(1/h_{i+j} + 1/h_0)\}$
    end for
end for
$\omega \xleftarrow{\$} \mathbb{F}_q$
▷ Assemble the Goppa generator polynomial:
for $i \leftarrow 0$ **to** $t - 1$ **do**
    $z_i \leftarrow 1/h_i + \omega$
end for
$g(x) \leftarrow \prod_{i=0}^{t-1} (x - z_i)$
▷ Compute the support:
for $j \leftarrow 0$ **to** $n - 1$ **do**
    $L_j \leftarrow 1/h_j + 1/h_0 + \omega$
end for
$h \leftarrow (h_0, \ldots, h_{n-1})$
$H \leftarrow \Delta(t, h)$
return L, g, H, η

matrix so as to preserve quasi-dyadicity in the target subfield subcode and the distribution of introduced errors in cryptosystems. A similar process yields a generator matrix in convenient quasi-dyadic, systematic form. Here, we describe the technique step-by-step.

1. For the desired security level, choose $q = 2^m$ for some m, a code length n and a design number of correctable errors t such that $n = \ell t$ for some $\ell > m$. For simplicity we assume that t is a power of 2, but the following construction method can be modified to work with other values.

2. Run Algorithm 1 to produce a code over $\mathbb{F}_q$ whose length $N \gg n$ is a large multiple of t not exceeding the largest possible length $q/2$, so that the constructed $t \times N$ parity-check matrix $\hat{H}$ can be viewed as a sequence of N/t dyadic blocks $[B_0 \mid \cdots \mid B_{N/t-1}]$ of size $t \times t$ each.

3. Select uniformly at random ℓ distinct blocks $B_{i_0}, \ldots, B_{i_{\ell-1}}$ in any order from $\hat{H}$.

4. Select uniformly at random ℓ dyadic permutations $\Pi^{j_0}, \ldots, \Pi^{j_{\ell-1}}$ of size $t \times t$.

5. Compute $\hat{H}' = [B_{i_0}\Pi^{j_0} \mid \cdots \mid B_{i_{\ell-1}}\Pi^{j_{\ell-1}}] \in (\mathbb{F}_q^{t \times t})^\ell$.

6. Compute the co-trace matrix $H' = T'_m(\hat{H}') \in (\mathbb{F}_2^{t\times t})^{m\times \ell}$.

7. Finally, compute the systematic form H from H'.

The resulting parity-check matrix defines a binary code of length n and dimension $k = n - mt$, and since all block operations performed during the Gaussian elimination are carried out in the ring $\Delta(\mathbb{F}_2^t)$, the result still consists of dyadic submatrices which can be represented by a signature of length t. Hence the whole matrix can be stored in space a factor t smaller than a general matrix. However, the dyadic submatrices that appear in this process are not necessarily nonsingular, as they are not associated to a Cauchy matrix anymore; should all the submatrices on a column be found to be singular (above or below the diagonal, according to the direction of this process) so that no pivot is possible, the whole block containing that column may be replaced by another block $B_{j'}$ chosen at random from $\hat{H}$ as above.

The trapdoor information consists of the sequence $(i_0, \ldots, i_{\ell-1})$ of blocks, the essence η of h and the sequence $(j_0, \ldots, j_{\ell-1})$ of dyadic permutation identifiers. It relates the public code defined by H with the private code defined by $\hat{H}$.

4.1.3 A toy example

Let $\mathbb{F}_{2^5} = \mathbb{F}_2[u]/(u^5 + u^2 + 1)$. The dyadic signature

$$h = (u^{20}, u^3, u^6, u^{28}, u^9, u^{29}, u^4, u^{22}, u^{12}, u^5, u^{10}, u^2, u^{24}, u^{26}, u^{25}, u^{15})$$

and the offset $\omega = u^{21}$ define a 2-error correcting binary Goppa code of length $N = 16$ with $g(x) = (x - u^{12})(x - u^{15})$ and support

$$L = (u^{21}, u^{29}, u^{19}, u^{26}, u^6, u^{16}, u^7, u^5, u^{25}, u^3, u^{11}, u^{28}, u^{27}, u^9, u^{22}, u^2).$$

The associated parity-check matrix built according to Theorem 4.1 is

$$\hat{H} = \left[\begin{array}{cc|cc|cc|c|cc|cc|cc} u^{20} & u^3 & u^6 & u^{28} & u^9 & u^{29} & \ldots & u^{10} & u^2 & u^{24} & u^{26} & u^{25} & u^{15} \\ u^3 & u^{20} & u^{28} & u^6 & u^{29} & u^9 & \ldots & u^2 & u^{10} & u^{26} & u^{24} & u^{15} & u^{25} \end{array}\right],$$

with eight 2×2 blocks $B_0, \ldots, B_7$ as indicated. From this we extract the shortened, rearranged and permuted sequence $\hat{H}' = [B_7\Pi^0 \mid B_5\Pi^1 \mid B_1\Pi^0 \mid B_2\Pi^1 \mid B_3\Pi^0 \mid B_6\Pi^1 \mid B_4\Pi^0]$ (because in this example the subfield is the base field itself, all scale factors have to be 1), *i.e.* $\hat{H} =$

$$\left[\begin{array}{cc|cc|cc|cc|cc|cc|cc} u^{25} & u^{15} & u^2 & u^{10} & u^6 & u^{28} & u^{29} & u^9 & u^4 & u^{22} & u^{26} & u^{24} & u^{12} & u^5 \\ u^{15} & u^{25} & u^{10} & u^2 & u^{28} & u^6 & u^9 & u^{29} & u^{22} & u^4 & u^{24} & u^{26} & u^5 & u^{12} \end{array}\right],$$

whose co-trace matrix over $\mathbb{F}_2$ has the systematic form:

$$H = \left[\begin{array}{cccc|cccccccccc} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] = [M^{\mathsf{T}} \mid I_{n-k}],$$

from which one readily obtains the $k \times n = 4 \times 14$ generator matrix in systematic form:

$$G = \left[\begin{array}{cccc|cccccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}\right] = [I_k \mid M],$$

where both G and H share the essential part M:

$$M = \left[\begin{array}{cccccccccc} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{array}\right],$$

which is entirely specified by the elements in boldface and can thus be stored in 20 bits instead of, respectively, $4 \cdot 14 = 56$ and $10 \cdot 14 = 140$ bits.

4.1.4 The Preliminary Version

The preliminary version of the quasi-dyadic work has appeared in [MB09b]. The only difference between the presented version and this preliminary work is the finite field where the final subfield sub-code is defined.

In the construction discussed in this thesis, we begin with a code defined over a finite field $\mathbb{F}_q$, for some prime power $q = p^m$. Then, in the co-trace construction, we define a subfield sub-code defined over the base field $\mathbb{F}_p$.

In contrast, in the preliminary version, we begin with a code defined over a finite field $\mathbb{F}_{q^m}$, where q is a prime power, *i.e.* $q = p^c$. However, in this case, the co-trace construction would produce a subfield sub-code defined over the intermediary field $\mathbb{F}_{p^c}$, not over the base field $\mathbb{F}_p$. Working on larger finite fields seemed to enable to reduce even more the code description and consequently the key-sizes. Unfortunately, as we will discuss in Section 4.4.3, the choice regarding the finite field where the public code is defined is a critical point in the security assessment of the scheme. In practice, this difference was enough to make algebraic structural attacks effective [FOPT10a, UL09].

4.1.5 Extending the Construction to CFS Signatures

Actually, the construction provided by Algorithm 1 is not optimal in terms of maximal code-length, which is restricted to $n \leqslant 2^{m-1}$. In this section, we

explain how to improve it to $n \leqslant 2^m - t$. The increased maximal code-length will allow the application of our codes to the CFS digital signature scheme. This improvement appeared in [BCMN10].

As discussed in Section 2.3.1 of Chapter 1, the CFS scheme is only practical when instantiated with codes whose density of decodable syndromes is high. For example, consider a t-error correcting $\mathbb{F}_p$-alternant code of length n derived from a code over $\mathbb{F}_{p^m}$. The syndrome space has size p^{mt}, but the decodable syndromes are only those that correspond to error vectors of weight not exceeding t. In other words, only $\sum_{w=1}^{t} \binom{n}{w}(p-1)^w$ nonzero syndromes are decodable, and hence its density is

$$\delta = \frac{1}{p^{mt}} \sum_{w=1}^{t} \binom{n}{w}(p-1)^w.$$

If the code length is a fraction $1/p^c$ for some $c \geqslant 0$ of the full length, *i.e.* $n = p^{m-c}$, the density can be approximated as

$$\delta \approx (n^t/t!)(p-1)^t/p^{mt} = (p^{m-c})^t(p-1)^t/(p^{mt}t!) = (p-1)^t/(p^{ct}t!).$$

A particularly good case is therefore $\delta \geqslant 1/t!$, which occurs when $(p^c/(p-1))^t \leqslant 1$, *i.e.* $c \leqslant \log_p(p-1)$, or $n \geqslant p^m/(p-1)$. For binary codes, this implies that the highest densities are attained only by full or nearly full length codes, otherwise the density is reduced by a factor 2^{ct}. For full length binary codes ($p = 2$, $n = 2^m$) the density simplifies to

$$\delta \approx \frac{1}{2^{mt}} \frac{n^t}{t!} = \frac{1}{t!}.$$

Algorithm 1 produces Goppa codes of fairly low density of decodable syndromes. Since the sequences z and L are disjoint and composed by distinct elements, the length of the codes Algorithm 1 produces are upper bounded by $n \leqslant 2^{m-1}$, and hence the syndrome density is bound by $1/(2^t t!)$. Clearly, if z and L were not disjoint at least one element $H_{ij} = 1/(z_i - L_j)$ of matrix H would be undefined due to division by zero.

However, the CFS signature scheme only needs a very small t (say, $t \lesssim m$), meaning that most elements of the sequence z, and hence the corresponding rows of the largest possible matrix $\Delta(h)$, are left unused anyway when defining the actual code. It is therefore possible to allow matrix $\Delta(h)$ to contain undefined entries, as long as the rows and columns containing those entries are removed afterwards, and that $\Delta(t, h)$ itself contains only well-defined entries. This means the code length can be naturally extended all the way up to $2^m - t$, corresponding to an exact partition of the field elements from $\mathbb{F}_{2^m}$ into two disjoint sequences z and L.

In principle, this strategy can fail and the first t rows could contain an undefined element. This can be handled by either choosing a different code, or else by carefully rearranging the dyadic signature h into some h' in order to permute the rows of $\Delta(h)$ and eliminate undefined elements from $\Delta(t, h')$. Empirically, we noticed that the best strategy for when an improper element appear on the first t rows of $\Delta(h)$ is the simplest one: just try another code.

This idea is captured in Algorithm 2, which in practice is as simple to implement and as efficient as Algorithm 1. In a sense it is actually somewhat simpler, since less field elements have to be computed and discarded from the remaining

allowed set U. Notice that improper array elements, whose evaluation would cause division by zero, are represented by a zero value, since this cannot ever occur on a proper array entry.

Algorithm 2 Constructing a CFS-friendly binary Goppa code in dyadic form

INPUT: m, n, t.

OUTPUT: A dyadic signature h from which a CFS-friendly t-error correcting binary Goppa code of length n can be constructed from a code over $\mathbb{F}_{2^m}$, and the sequence b of all consistent blocks of columns (*i.e.* those that can be used to define the code support).

1: $q \leftarrow 2^m$
2: **repeat**
3: $U \leftarrow \mathbb{F}_q \setminus \{0\}$
4: $h_0 \xleftarrow{\$} U$, $U \leftarrow U \setminus \{h_0\}$
5: **for** $s \leftarrow 0$ **to** $m - 1$ **do**
6: $i \leftarrow 2^s$
7: $h_i \xleftarrow{\$} U$, $U \leftarrow U \setminus \{h_i\}$
8: **for** $j \leftarrow 1$ **to** $i - 1$ **do**
9: **if** $h_i \neq 0$ **and** $h_j \neq 0$ **and** $1/h_i + 1/h_j + 1/h_0 \neq 0$ **then**
10: $h_{i+j} \leftarrow 1/(1/h_i + 1/h_j + 1/h_0)$
11: **else**
12: $h_{i+j} \leftarrow 0$ ▷ undefined entry
13: **end if**
14: $U \leftarrow U \setminus \{h_{i+j}\}$
15: **end for**
16: **end for**
17: $c \leftarrow 0$ ▷ also: $U \leftarrow \mathbb{F}_q$
18: **if** $0 \notin \{h_0, \ldots, h_{t-1}\}$ **then** ▷ consistent root set
19: $b_0 \leftarrow 0$, $c \leftarrow 1$ ▷ also: $U \leftarrow U \setminus \{1/h_i, 1/h_i + 1/h_0 \mid i = 0, \ldots, t-1\}$
20: **for** $j \leftarrow 1$ **to** $\lfloor q/t \rfloor - 1$ **do**
21: **if** $0 \notin \{h_{jt}, \ldots, h_{(j+1)t-1}\}$ **then** ▷ consistent support block
22: $b_c \leftarrow j$, $c \leftarrow c + 1$ ▷ also: $U \leftarrow U \setminus \{1/h_i + 1/h_0 \mid i = jt, \ldots, (j+1)t - 1\}$
23: **end if**
24: **end for**
25: **end if**
26: **until** $ct \geqslant n$ ▷ consistent roots and support
27: $h \leftarrow (h_0, \ldots, h_{q-1})$, $b \leftarrow (b_0, \ldots, h_{c-1})$ ▷ also: $\omega \xleftarrow{\$} U$
28: **return** h, b ▷ also: ω

Algorithm 2 produces a code that is amenable to the same treatment as described for the codes produced by Algorithm 1, *i.e.* 1) select a sub-set of blocks 2) permute them, 3) dyadic-permute each block individually, 4) apply the co-trace construction to get a description of a binary quasi-dyadic alternant code, and 5) compute its systematic form. This has to be done carefully so as to fully hide the code structure. The obvious approach is to delete more blocks and/or to replace them (and also the blocks that contain improper columns) by random dyadic blocks (the latter case corresponds to Wieschebrink's technique). One has to be careful here as well, since if only a fraction $1/2^c$ of the columns remain, the syndrome density effectively decreases by a factor 2^{ct} as seen above. A sensible choice, which we will usually adopt, is to take a fraction $2^{-1/t}$ of full code length (*i.e.* $c = 1/t$), since this only increases the average signing time by a factor of 2.

In an independent work, Kobara proposed a construction [Kob09], called "Flexible Quasi-Dyadic", to increase the maximum code-length from what is obtained by Algorithm 1, as well. This approach is based on solving linear restrictions that prevent the existence of non-defined entries in the final quasi-

dyadic Goppa code. Although both approaches achieve similar results, the "flexible" construction demands an extra cost for solving and manipulating these linear equations.

4.2 The General Case ($p \geq 2$)

Most attempts at decreasing key sizes deal with codes in characteristic 2, in spite of evidence that odd characteristics may offer security advantages [Pet10]. In this section we explain how the approach started with quasi-dyadic codes can be generalized to codes defined over finite fields of characteristic greater than 2. We start by introducing p-adic matrices, which generalize dyadic matrices.

Definition 4.5 (p-adic Matrices)**.** *Let $A = \{a_0, \ldots, a_{N-1}\}$ be a finite abelian group of size $|A| = p^d$ with neutral element $a_0 = 0$, R a commutative ring and $h\colon A \longrightarrow R$ a sequence indexed by A. The p-adic matrix $M(h)$ associated with this sequence is one for which $M_{i,j} = h(a_i - a_j)$ holds,* i.e.

$$M = \begin{pmatrix} h(0) & h(-a_1) & \cdots & h(-a_{N-1}) \\ h(a_1) & h(0) & \cdots & h(a_1 - a_{N-1}) \\ \vdots & \vdots & \ddots & \vdots \\ h(a_{N-1}) & h(a_{N-1} - a_1) & \cdots & h(0) \end{pmatrix}.$$

Remark 11. p-adic matrices form a commutative subring of $\mathcal{R}^{n\times n}$ as long as $\mathcal{R}$ is commutative.

In Definition 4.5, we use the additive notation for the finite abelian group A. However, the definition can be generalized to all groups, in which case one might prefer the multiplicative notation.

Some A-adic matrices have special names, for example the p-adic matrices for $p = 2$ are dyadic for $p = 3$ are triadic matrices. If we do not want to specify the group A explicitly, we will say the matrix is p-adic. We continue by giving necessary and sufficient conditions for Cauchy matrices to be p-adic, thus identifying all p-adic Goppa codes.

Theorem 4.3. *Let $M(h)$ be p-adic matrix for a sequence h of length p^d over $\mathbb{F}_{p^m}$. Then M is Cauchy iff*

(1) $h(a_i)$ *are distinct and invertible in* $\mathbb{F}_{p^m}$ *for all* $0 \leq i < p^d$, *and*

(2) $\dfrac{1}{h(a_i - a_j)} = \dfrac{1}{h(a_i)} + \dfrac{1}{h(-a_j)} - \dfrac{1}{h(0)}$ *for all* $0 \leq i, j < p^d$.

In this case $M(h) = C(\beta, \gamma)$, where

$$\begin{aligned} \beta(a_i) &= \frac{1}{h(a_i)} \\ \gamma(a_i) &= \frac{1}{h(0)} - \frac{1}{h(-a_i)}. \end{aligned}$$

Proof. We start by showing that our conditions indeed imply that M is Cauchy. For the disjointness, assume that there are indices i and j, such that $\beta(a_i) =$

$\gamma(a_j)$. In this case we get $0 = \beta(a_i) - \gamma(a_j) = 1/h(a_i - a_j)$, which is a contradiction. Finally we compare the matrices $M(h)$ and $C(\beta, \gamma)$ resulting in the equality

$$M_{i,j} = h(a_i - a_j) = 1/(1/h(a_i) + 1/h(-a_j) - 1/h(0)) = 1/(\beta(a_i) - \gamma(a_j)) = C_{i,j}.$$

We continue by showing that if M is Cauchy, *i.e.* , $M(h) = C(\beta', \gamma')$, then indeed our conditions must hold. Since $C(\beta', \gamma') = C(\beta' + \omega, \gamma' + \omega)$ for any $\omega \in \mathbb{F}$, we can choose the sequences in such a way that $\gamma'(0) = 0$. Now, $M_{i,0} = C_{i,0}$ for all i, which means $h(a_i) = 1/\beta'(a_i)$. By the properties of β' this gives us condition (1), *i.e.* , that all $h(a_i)$ are distinct and invertible, as well as $\beta' = \beta$. We use similarly that $M_{0,i} = C_{0,i}$ which implies $h(-a_i) = 1/(\beta(0) - \gamma'(a_i))$. Solving for γ' reveals that it equals γ. Since $\beta = \beta'$ and $\gamma = \gamma'$, we get that $M(h) = C(\beta, \gamma)$ implying condition (2). □

Now, we will show how to construct random p-adic Cauchy matrices.

Corollary 4.1. *Let A be a finite abelian group of order $|A| = p^d$ with generators $b_1, \ldots, b_d$ and $M(h)$ be p-adic and Cauchy for a sequence h over $\mathbb{F}$. Then for all $c_1, \ldots, c_d \in \mathbb{Z}$,*

$$(h(c_1 b_1 + \cdots + c_d b_d))^{-1} = c_1(h(b_1))^{-1} + \cdots + c_d(h(b_d))^{-1} - (c_1 + \cdots + c_d - 1)(h(0))^{-1}.$$

Moreover, the field characteristic of $\mathbb{F}$ divides the order of any element in $A \setminus \{0\}$.

Proof. By Theorem 4.3, we know that for all $a, a' \in A$ the following holds

$$(h(a + a'))^{-1} = (h(a))^{-1} + (h(a'))^{-1} - (h(0))^{-1}.$$

By repeatedly using this equation, we prove the first claim.

$$\begin{aligned}
(h(c_1 b_1 + \cdots + c_d b_d))^{-1} &= (h(\underbrace{b_1 + \cdots + b_1}_{c_1 \text{ times}} + \cdots + \underbrace{b_d + \cdots + b_d}_{c_d \text{ times}}))^{-1} \\
&= (h(b_1))^{-1} + (h(\underbrace{b_1 + \cdots + b_1}_{(c_1 - 1) \text{ times}} + \cdots + \underbrace{b_d + \cdots + b_d}_{c_d \text{ times}}))^{-1} - (h(0))^{-1} \\
&= c_1(h(b_1))^{-1} + (h(\underbrace{b_2 + \cdots + b_2}_{(c_2) \text{ times}} + \cdots + \underbrace{b_d + \cdots + b_d}_{c_d \text{ times}}))^{-1} - c_1(h(0))^{-1} \\
&= c_1(h(b_1))^{-1} + \cdots + c_d(h(b_d))^{-1} - (c_1 + \cdots + c_d - 1)(h(0))^{-1}.
\end{aligned}$$

For the second claim, let $a \in A \setminus \{0\}$ be a non-neutral group element and $k = \mathrm{ord}(a)$, *i.e.* , $ka = 0$. By the equation we have just shown, we know that

$$\begin{aligned}
h(0)^{-1} = h(ka)^{-1} &= kh(a)^{-1} - (k-1)h(0)^{-1} \\
k(h(0)^{-1} - h(a)^{-1}) &= 0
\end{aligned}$$

Since a is not the neutral element, all elements of h are distinct, and the field characteristic is prime, the second claim follows. □

Note that since the field characteristic p divides the order of any element, only groups of size p^d can be used. Conversely, let $b_1, \ldots, b_d$ be group elements that form an $\mathbb{F}_p$ set of generators, then the sequence elements $h(0), h(b_1), \ldots, h(b_d)$ completely determine the sequence. We call these values the essence of the sequence h, analogously to the essence defined for dyadic matrices. For example, if $A = \mathbb{F}_p^d$, then such a set of generators $b_1, \ldots, b_d$ is given by the generators of the d distinct copies of $\mathbb{F}_p$ in A. For a given set of generators, we can sample a p-adic sequence uniformly at random with the algorithm in Algorithm 3. Figure 4.3 summarizes the restriction over the parameters.

Description	Parameter	Restriction
Field char	p	prime
Extension field	p^m	–
Group order	p^d	$d \leq m-1$
Goppa roots	t	$< n/m$
Goppa multiplicity	r	$< n/(tm)$
Blocksize	$\mathtt{b}$	$\gcd(t, N)$
Code length	n	$\mathtt{b}\ell < N$

Figure 4.3: Parameters for quasi-p-adic Goppa codes.

Algorithm 3 Constructing a Goppa code in p-adic form. Choosing A-adic Cauchy sequences, where $A = \{0, a_1, \ldots, a_{p^d-1}\}$ has set of generators $b_1, \ldots, b_d$.

INPUT: p, m, d.
OUTPUT: .
1: $U \leftarrow \mathbb{F}_{p^m} \setminus \{0\}$
2: $h(0) \xleftarrow{\$} U$
3: **for** $i = 1, \ldots, d$ **do**
4: $\quad U \leftarrow \mathbb{F}_{p^m} \setminus \{\mathbb{F}_p h(0) + \mathbb{F}_p h(b_1) + \cdots + \mathbb{F}_p h(b_{i-1})\}$
5: $\quad h(b_i) \xleftarrow{\$} U$
6: **end for**
7: **for** $c_1, \ldots, c_d \in \mathbb{F}_p$ **do**
8: $\quad h(c_1 b_1 + \cdots + c_d b_d) \leftarrow c_1 h(b_1) + \cdots + c_d h(b_d) - (c_1 + \cdots + c_d - 1)h(0)$
9: **end for**
10: **return** $\left(h(0)^{-1}, h(a_1)^{-1}, \ldots, h(a_{p^d-1})^{-1}\right)$

We will briefly argue why Algorithm 3 is correct. Assume it is not. The only situation resulting in an error is in line 7, if the computed quantity is not invertible, so let us assume this to be the case. Since only zero is not invertible, we have

$$0 = c_1 h(b_1) + \cdots + c_d h(b_d) - (c_1 + \cdots + c_d - 1)h(0).$$

Now, not all coefficients of $h(0), h(b_1), \ldots, h(b_d)$ can be zero simultaneously, so there is an $\mathbb{F}_p$-linear dependency among them. However, by our choice of F in line 4, from which all $h(b_i)$ are chosen, no such dependency can exist.

Algorithm 3 produces a code that is amenable to the same treatment as described for the codes produced by Algorithm 1, producing then a quasi-p-adic Goppa code:

1. Construct a random fully p-adic Goppa code of length p^d with Algorithm 3.

2. Split the support in blocks of length $\mathtt{b}$ and select ℓ such blocks at random to comprise the support of the quasi-p-adic code.

3. To each chosen support block, apply a random p-adic permutation, *i.e.* , multiply with the matrix $M(\chi_{a_\pi})$, where χ_{a_π} is the characteristic function of the group element a_π, for a randomly chosen π. Since χ is a characteristic function, this matrix will have a single non-zero coefficient being 1 per row and column, so it is a permutation matrix. Furthermore, this transformation preserves the p-adic structure of the block and indeed all p-adic permutations have this form.

4. Generate a parity-check matrix for the corresponding subfield subcode over the base field $\mathbb{F}_q$. Recall that $\mathbb{F}_{p^m} = \mathbb{F}_q[x]/\langle f \rangle$ for some irreducible polynomial f of degree m. We can identify each matrix coefficient $h_{i,j}$ with its representative polynomial $h_{i,j,0} + h_{i,j,1}x + \cdots + h_{i,j,m-1}x^{m-1}$ of smallest degree. We expand the matrix rows by a factor of m and distribute the entries as follows $h^{\text{new}}_{kt+i,j} \longleftarrow h_{i,j,k}$,*i.e.* , in order to keep the block structure intact, we first take all constant terms of coefficients in a block then all linear terms and so on.

5. Compute the quasi-p-adic systematic form of the parity-check matrix. It does so by identifying each p-adic block with an element of the corresponding ring of p-adic matrices and performing the usual Gauss algorithm on those elements. Since this ring is not necessarily an integral domain, the algorithm may find that a pivot element is not invertible. In this case, the systematic form we seek does not exist and the algorithm has to loop back to the "Block permutation" step. Fortunately, the chance of this is small. The probability that the matrix is nonsingular is $\prod_{j=0}^{k-1} 1 - 1/p^{k-j}$, which approaches a constant (to be determined numerically) for large k. This constant is different for each p but tends to 1 for large p.

Decoding p-adic Goppa codes. p-adic Goppa codes can be efficiently decoded by the decoder presented in [BML13]. This work is a generalization of Patterson's decoding algorithm for Goppa codes with square-free Goppa polynomial defined over finite fields of characteristics greater than 2. Note that p-adic Goppa codes also have square-free Goppa polynomials since their roots are distinct by construction.

A priori, this decoding algorithm can correct $(2/p)t$ errors with high probability. Nonetheless, the correction capability is higher than $(2/p)t$ if the distribution of error magnitudes is not uniform, approaching or reaching t errors when any particular error value occurs much more often than others or exclusively. This is good for cryptography, where we could be allowed to arbitrarily choose all-equal error magnitude.

Generalized-Srivastava codes in p-adic form. In [BLM11], it is also discussed the possibility of extending the p-adic construction to define Generalized-Srivastava codes. This remark is related to the work of Persichetti [Per11], where the dyadic construction ($p = 2$) was used to define Quasi-Dyadic Generalized-Srivastava codes. In this thesis, we preferred to focus on obtaining compact-keys either from the reputable family of Goppa codes or from codes free of algebraic structure.

4.3 Efficiency

In this section, we discuss the efficiency of our approach in terms of both algorithmic and storage complexity.

4.3.1 Algorithmic Complexity

Below, the algorithmic complexity for key-generation, encryption and decryption using p-adic Goppa codes.

Key-Generation: The predominant cost in the key generation is the Gaussian elimination needed to convert a quasi-dyadic matrix to systematic (echelon) incurring about $m^2\ell$ products of dyadic $t \times t$ submatrices, implying a complexity $O(m^2\ell t \lg t) = O(m^2 n \lg n)$, which simplifies to $O(n \lg^3 n)$ in the typical cryptographic setting $m = O(\lg n)$. The other steps have complexity $O(n)$ and therefore do not influence the overall complexity. More precisely, Algorithm 1 has complexity $O(n)$ because each element of the signature h is assigned a value exactly once. Also, the inversion has complexity $O(n)$, since a binary dyadic matrix $\Delta(h)$ of dimension n satisfies $\Delta^2 = (\sum_i h_i)^2 I$. This implies that its inverse, when it exists, is $\Delta^{-1} = (\sum_i h_i)^{-2}\Delta$, which can be computed in $O(n)$ steps since it is entirely determined by its first row.

Encryption: The operation of multiplying a vector by a (quasi-)dyadic matrix is at the core of McEliece encryption. The fast Walsh-Hadamard transform (FWHT) [Gul73] approach for dyadic convolution via lifting[1] to characteristic 0 leads to the asymptotic complexity $O(n \lg n)$ for this operation and hence also for encoding.

Decryption: The decryption in McEliece-like schemes boils down to decoding. Sarwate's decoding method [Sar77] sets the asymptotic cost of that operation at roughly $O(n \lg n)$ as well for the typical cryptographic setting $t = O(n/\lg n)$. For p-adic Goppa codes, the efficient decoding algorithm for square-free (irreducible or otherwise) Goppa codes over $\mathbb{F}_p$ for any prime p presented [BML13] can be used.

4.3.2 Storage Complexity

In this section, we discuss the cost of storing both private and public information regarding our codes. Note that due to their simple structure, the matrices can be held on a simple vector not only for long-term storage or transmission, but for processing as well.

Private-key: The space occupied by the trapdoor information, *i.e.* the private McEliece key, is thus $m^2 + \ell \lg N$ bits. If one starts with the largest possible $N = 2^{m-1}$, this simplifies to the maximal size of $m^2 + \ell(m-1)$ bits.

Public-key: The total space occupied by the essential part of the resulting binary generator (or parity-check) matrix, *i.e.* the public McEliece key, is

[1] We are grateful to Dan Bernstein for suggesting the lifting technique to emulate the FWHT in characteristic 2.

$mt \times (n - mt)/t = mk$ bits, a factor t better than generic Goppa codes, which occupy $k(n-k) = mkt$ bits. Had t not been chosen to be a power of 2, say, $t = 2^u v$ where $v > 1$ is odd, the final parity-check matrix would be compressed by only a factor 2^u.

Implementation. In [Hey11], a complete implementation of the McEliece cryptosystem instantiated with quasi-dyadic Goppa codes is presented. This implementation takes into account a CCA2-secure conversion for achieving semantic security. Due to the extremely compact key-size of our proposal, the authors were able to make such implementation on a 8-bits AVR microcontroller, without any external memory. Moreover, according to the authors, this work outperforms the previous available implementations of the original McEliece in encryption and storage complexity.

4.4 Security Assessment

The key security of the p-adic Goppa proposal deserves much attention. As an example, the pre-print version of the dyadic Goppa proposal had some parameters successfully attacked. These parameters had to be discarded, but they were useful to better understand the trade-offs present in the security assessment of our proposal. In this section, we analyze possible attacks aimed at recoveirng the private p-adic Goppa codes in McEliece-like cryptosystems.

4.4.1 Exhaustive Search in the Key Space

The first type of attacks that must be considered is the exhaustive search in the key space.

Theorem 4.4. *Algorithm 1 produces up to* $\prod_{i=0}^{\lceil \lg n \rceil} (q - 2^i)$ *distinct essences of dyadic signatures corresponding to Cauchy matrices.*

Proof. Each dyadic signature produced by Algorithm 1 is entirely determined by the values h_0 and h_{2^s} for $s = 0, \ldots, \lceil \lg n \rceil - 1$ chosen at steps 2 and 7 (ω only produces equivalent codes). Along the loop at line 5, exactly $2i = 2^{s+1}$ elements are erased from U, corresponding to the choices of $h_{2^s} \ldots h_{2^{s+1}-1}$. At the end of that loop, $2 + 2\sum_{\ell=0}^{s} 2^\ell = 2^{s+2}$ elements have been erased in total. Hence at the beginning of each step of the loop only 2^{s+1} elements had been erased from U, *i.e.* there are $q - 2^{s+1}$ elements in U to choose h_{2^s} from, and $q - 1$ possibilities for h_0. Therefore this construction potentially yields up to $(q-1)\prod_{s=0}^{\lceil \lg n \rceil - 1} (q - 2^{s+1}) = \prod_{i=0}^{\lceil \lg n \rceil} (q - 2^i)$ possible codes. □

Note that Algorithm 2, used to build binary quasi-dyadic Goppa codes suitable for CFS signatures, is a relaxed version of Algorithm 1 and thus generates more possible outputs. Therefore we can use the analysis presented to Algorithm 1 as a lower bound to Algorithm 2, which has a non-trivial estimation.

In Theorem 4.5, we prove the bound for p-adic Goppa codes generated by Algorithm 3, which coincides with Theorem 4.4 when $p = 2$.

Theorem 4.5. *Algorithm 3 produces up to* $\prod_{i=0}^{\lceil \lg_p n \rceil} (q - p^i)$ *distinct essences of p-adic signatures corresponding to Cauchy matrices.*

Proof. Each dyadic signature produced by Algorithm 3 is entirely determined by the values h_i, $i \in (1, \ldots, d)$. For each value $(h_i)_{1 \leq i \leq d}$, all possible combinations of the previous $h_{i'<i}$ elements are erased from U. Then for each $(h_i)_{1 \leq i \leq d}$ it is possible to choose h_i from $q - p^{i-1}$. Thus it yields up to $\prod_{i=0}^{\lceil \lg_p n \rceil} (q - p^i)$ possible codes. □

For each p-adic code produced by either Algorithm 1 and Algorithm 3, the number of codes generated by this construction is $\binom{N/t}{\ell} \times \ell! \times t^\ell$, hence $\binom{N/t}{\ell} \times \ell! \times t^\ell \times \prod_{i=0}^{\lceil \lg_p N \rceil} (q - p^i)$ p-adic Goppa codes are possible in principle. This estimation is a lower bound for when using Algorithm 2.

4.4.2 OTD Attack

The attack presented in [OTD10] was successful against the scheme proposed by Gaborit using quasi-cyclic BCH codes [Gab05]. This attack relies on the fact that public and private code are permutation-equivalent. For this reason, it is not effective against our construction, since we use the same technique to hide the private code structure as presented in [BCGO09]. It consists of using as the public code a subfield subcode of a punctured block-permuted version of the private code. This approach relies on the work of Wieschebrink [Wie06], who proved that deciding whether a code is equivalent or not to a shortened code is an NP-complete problem.

4.4.3 FOPT Attack

The attack proposed in [FOPT10a] by Faugère, Otmani, Perret and Tillich is based on the relationship between generator and parity-check matrices, namely

$$GH^T = 0. \tag{4.2}$$

The generator matrix $G \in \mathbb{F}_q^{k \times n}$ is public. The authors use the fact that Goppa codes are also alternant codes, an thus they propose to solve the system defined by 4.2 for an unknown alternant parity-check matrix H of the form:

$$H = \begin{pmatrix} Y_0 & Y_1 & \ldots & Y_{n-1} \\ Y_0 X_0 & Y_1 X_1 & \ldots & Y_{n-1} X_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ Y_0 X_0^i & Y_1 X_1^i & \ldots & Y_{n-1} X_{n-1}^i \\ \vdots & \vdots & \vdots & \vdots \\ Y_0 X_0^{r-1} & Y_1 X_1^{r-1} & \ldots & Y_{n-1} X_{n-1}^{r-1} \end{pmatrix} \tag{4.3}$$

This leads to a system of rn non-linear equations and $2n$ unknowns:

$$\{g_{i,0} Y_0 X_0^j + \cdots + g_{i,n-1} Y_{n-1} X_{n-1}^j = 0 | i \in \{0, \ldots, k-1\}, j \in \{0, \ldots, r-1\}\} \tag{4.4}$$

This system cannot be solved for usual parameters of the original McEliece scheme using binary Goppa codes. However, the redundancy added by the dyadic (or cyclic) structure strongly decreases the number of unknowns of this system. Below we analyze the scenario for quasi-dyadic Goppa codes.

Consider a quasi-dyadic Goppa code of length $n = n_0 t$, with dyadic blocks of size $t \times t$. Thus, for any $0 \leq j \leq n_0 - 1$ and $0 \leq i, i' \leq t - 1$, it holds that:

$$Y_{jt+1} = Y_{jt} \tag{4.5}$$
$$X_{jt+i} + X_{jt} = X_i + X_0 \tag{4.6}$$
$$X_{jt+(i \oplus i')} = X_{jt+i} + X_{jt+i'} + X_{jt}. \tag{4.7}$$

Relation 4.5 states that all Y's in the same $t \times t$ dyadic block are equal. Thus, the number of unknowns Y's is actually $n/t - 1 = n_0 - 1$. The minus one comes from the fact that we can arbitrarily choose one of Y's. Since all Y's in the same block are equal, they define redundant equations. Thus, the number of relevant equations involving Y's is $k/t = n_0 - m$. The number of unknowns X's is $n_0 + \log_2 t - 2$ unknowns, and it comes from 4.6 and 4.7. Finally, there are $tk - k$ non-linear equations and, since $k = t(n_0 - m)$, it corresponds to $t(t-1)(n_0 - m)$.

	Normal Goppa Code	Quasi-Dyadic Goppa Code
Unknowns Y's	n	$n_0 - 1$
Unknowns X's	n	$n_0 + \log_2(t) - 2$
Equations involving Y's	k	$n_0 - m$
Non-Linear Equations	rn	$t(t-1)(n_0 - m)$

Table 4.1: Differences between attacking normal and quasi-dyadic Goppa codes.

Algorithm 4 illustrates the steps of the attack. The input is a code defined over $\mathbb{F}_{p^c}$ built from a code defined over $\mathbb{F}_q^m$, where q is a prime power $q = p^c$, which coincides with the preliminary version of p-adic Goppa codes [MB09b].

Algorithm 4 FOPT Attack.

INPUT: A generator-matrix $G \in \mathbb{F}_{p^c}^{k \times n}$, with entries $(g_{i,j})_{0 \leq i \leq k-1, 0 \leq j \leq n}$ of the code defined over $\mathbb{F}_p^c$, composed by blocks of size $t \times t$. Then $n = n_0 t$, $k = k_0 t$ and $r = r_0 t$.
OUTPUT: A parity-check matrix H defined over $\mathbb{F}_{q^m} = \mathbb{F}_{p^{mc}}$.

1: Build the system of equations:

$$\{g_{i,0} Y_0 X_0^j + \cdots + g_{i,n_1} Y_{n-1} X_{n-1}^j = 0 | i \in \{0, \ldots, k-1\}, j \in \{0, \ldots, t-1\}\}$$

2: Choose $n_{Y'} \geq n - k$ variables Y' from Y.
3: Express all other variables in $Y \setminus Y'$ in terms of Y'.
4: Compute the project of the solutions regarding the variables Y'.
5: Once having determined Y', the system can be expressed only in terms of X:

$$\{g'_{i,0} X_0^j + \cdots + g'_{i,n_1} X_{n-1}^j = 0 | i \in \{0, \ldots, k-1\}, j \in \{0, \ldots, t-1\}\}$$

6: Consider now the subset of the equations having degree equal to a power of two, *i.e.* $j = 2^l$, for $l = \{1, \ldots, \log_2(t-1)\}$.
7: Use the Frobenius automorphism to produce a system over $\mathbb{F}_2$, consisting of mcn unknowns and $mc \log_2(t-1)k$ equations.
8: Solve the system for X_i and **return** H built from (X, Y) as presented in 4.3.

In step 3, the variables in Y depend on the variables of Y'. For this reason, variables in Y are called *dependent* and variables in Y' are called *free-variables*. Step 4 is done by computing a Gröbner basis. This is the most expensive step of this attack. Note that the system in step 6 can be solved when $\log_2(t-1)k > n$.

This technique was effective in attacking p-adic Goppa codes defined over intermediate fields (as presented in 4.1.4). However, the attack failed in breaking the parameters that use public-codes defined over the binary field. In this case, the computation of the Gröner basis cited in the previous paragraph is easy, but trivial. It results in only one equation, not providing enough information to proceed with the attack. This is mostly due to the fact that only a subset of equations can be selected (see step 5 of Algorithm 4).

The algorithmic complexity of this attack is exponential, but a precise estimation of its cost remains an open problem. Recently the authors presented more accurate formulas in [FOPT10b], which justifies the reasoning that its complexity increases steeply as the codes are defined over smaller extension fields, making it unfeasible for public codes defined over the base field. For this reason, we select only p-adic Goppa codes defined over the base field $\mathbb{F}_p$, for p a prime number.

4.5 Suggested Parameters

In this section, we present parameters for dyadic and p-adic Goppa codes to instantiate encryption (McEliece and Niederreiter schemes) and for signatures (Parallel CFS scheme). For dyadic Goppa codes, the security levels are according to the non-asymptotic analysis of [BJMM12], the most recent variant of the information set decoding technique. For p-adic Goppa codes, the security levels are according to [Pet11, Remark 6.9] and also [BLP11]. In all tables, 'level' refers to the security level.

4.5.1 Quasi-Dyadic Goppa Codes for Encryption

Table 4.2 contains suggested parameters for different practical security levels to instantiate the McEliece cryptosystem. The number of errors is always a power of 2 to enable maximum size reduction, and the original code from which the binary Goppa code is extracted is always defined over $\mathbb{F}_{2^{16}}$.

The 'Size' column contains the amount of bits effectively needed to store a quasi-dyadic generator or parity-check matrix in systematic form. The size of a corresponding systematic matrix for a generic Goppa code at roughly the same security level as suggested in [BLP08] is given on column 'Generic'. In both cases we take into account only the redundant part of the key in systematic form. The 'Shrink' column contains the size ratio between such a generic matrix and a matching quasi-dyadic matrix.

Level	m	n	k	t	WF	Size	Generic	Shrink
80	11	1792	1088	64	82.518	11968	661122	55.2
80	12	3840	768	256	81.499	9216	661122	71.7
112	12	2944	1408	128	116.735	16896	1524936	90.3
128	12	3200	1664	128	131.235	19968	1991880	99.8
192	13	5888	2560	256	205.804	33280	5215496	156.7
256	15	11264	3584	512	279.002	53760	9276241	172.5

Table 4.2: Sample parameters for a $[n, k, 2t + 1]$ binary Goppa code generated by Algorithm 2.

The second parameter set for 80-bits of security achieves very compact key-sizes, at the price of a reduced code rate.

4.5.2 Quasi-Dyadic Goppa codes for Signatures

Table 4.3 presents the suggested parameter for the Parallel CFS signature scheme with quasi-dyadic Goppa codes. The 'Size' column contains its key size in systematic form. The 'Generic' column contains the key size in systematic form of generic Goppa codes. We assume codes of maximum length, namely $n = \lfloor (2^m - t)/t \rfloor t$. The column σ presents the number of signatures and the column δ the number of additional errors to perform the complete decoding approach as described in Section 2.3.1.

Level	t	m	σ	δ	Generic	Keys
80	8	20	3	2	20480 KiB	2560 KiB
112	8	27	4	1	3456 MiB	432 MiB
128	12	22	3	3	132 MiB	33 MiB

Table 4.3: Parallel-CFS signature with quasi-dyadic Goppa codes.

4.5.3 Quasi-p-adic Goppa Codes for Encryption

In this section, we present parameters for quasi p-adic Goppa codes. Table 4.4 suggests parameters for encryption schemes, aiming at achieving compact keys. It is assumed the ability to correct t errors of equal magnitude, using *e.g.* the decoding method for square-free Goppa codes proposed in [BML13].

Level	p	m	n	k	t	Keys (bits)	Syndrome (bits)
80	2	12	3840	768	256	9216	3072
80	3	8	2430	486	243	6163	3082
80	5	5	1000	375	125	4354	1452
112	2	12	2944	1408	128	16896	1536
112	3	8	2673	729	243	9244	3082
112	11	5	1089	484	121	8372	2093
128	2	12	3200	1664	128	19968	1536
128	3	9	3159	972	243	13866	3467
128	5	5	5000	625	625	10159	10159
192	2	14	6144	2560	256	35840	3584
192	3	10	4131	1701	243	26961	3852
192	29	6	5887	841	841	24514	24514
256	2	15	11264	3584	512	53760	7680
256	7	9	5145	2058	343	51998	8667
256	37	6	9583	1369	1369	42791	42791

Table 4.4: Encryption quasi-p-adic Goppa codes.

Table 4.5 presents parameters also for encryption schemes, but focusing on achieving compact syndromes. One can argue that minimizing keys may not be the best way to reduce bandwidth occupation. We might expect to exchange encrypted messages considerably more often than certified keys. Therefore, they

are suitable for Niederreiter scheme. Table 4.5 suggests codes that satisfy these requirements, without incurring unduly long keys. One sees that the choice for short syndromes often implies longer codes for larger characteristics.

Level	p	m	n	k	t	Key (bits)	Syndrome (bits)
80	2	11	1792	1088	64	11968	704
80	7	5	735	490	49	6879	688
80	41	3	451	328	41	5272	659
128	2	12	3200	1664	128	19968	1536
128	3	9	2106	1377	81	19643	1156
128	7	6	1813	1519	49	25587	826
192	2	14	5376	3584	128	50176	1792
192	3	11	4536	3645	81	63550	1413

Table 4.5: Encryption quasi-p-adic Goppa codes yielding short syndromes.

4.5.4 Quasi-p-adic Goppa Codes for Signatures

In Table 4.6, we present parameters to instantiate the Parallel-CFS signature scheme. The signature size presented in 'Sign. (bits)' column is slightly smaller than the product of the the syndrome size by the number of parallel signatures. The signing times are $O(t!)$. Quasi-p-adic codes in larger characteristics yield either shorter keys and signatures than in the binary case, or else considerably shorter signing times due to smaller values of t. Recall that σ denotes the number of signatures and δ the additional number of errors necessary to use the complete decoding approach.

Level	p	m	n	k	t	Key(KiB)	σ	δ	Sign.(bits)
80	2	15	32580	32400	12	178	2	4	326
80	3	11	177048	176949	9	377	3	2	375
80	13	4	28509	28457	13	52	2	4	342
112	2	20	1048332	1048092	12	7677	3	3	636
112	11	6	1771495	1771429	11	4489	3	2	558
112	13	5	371228	371163	13	839	3	3	624
128	2	23	8388324	8388048	12	70652	3	2	684
128	5	8	390495	390375	15	2656	3	4	759
128	13	6	4826731	4826653	13	13082	2	3	514

Table 4.6: Parallel CFS with quasi-p-adic Goppa codes.

Part II

Graph-Based Codes

Chapter 5

Introduction

The code-based cryptography community has been mainly focused on algebraic codes for more than thirty years. The theory behind these codes is widely understood and they present features such as efficient decoding algorithms and easy minimum distance estimation. Recently, a new trend started in [MRS00] has attracted much attention. It consists of instantiating code-based cryptosystems with graph-based codes. This new approach presents many security advantages.

As discussed in Chapter 1, the security of code-based cryptosystems is based on two hardness assumptions: the indistinguishability of the code family and generic decoding. The decoding problem is a well studied NP-complete problem [BMvT78], believed to be hard after decades of research. On the other hand, the indistinguishability problem is usually the weakest one, strongly depending on the choice of the code family.

Even Goppa codes, very reputable as a secure choice for code-based cryptosystems, have recently been affected. In [FGO+11], a distinguisher for high rate Goppa codes was presented. Thus the indistinguishability problem for Goppa codes might not be always sufficiently hard. Although it does not necessarily lead to a practical attack, it suggests that Goppa (and more generally, algebraic) codes may not be the optimal choice for code-based cryptography.

Algebraic codes have also been suggested for reducing the code-based cryptosystem keys. They were combined with some redundant structure, such as cyclic or dyadic structure. However, structural algebraic attacks [FOPT10a] succeeded in breaking many of them (this does not include p-adic Goppa codes, as presented in Chapter 4). The effectiveness of this attack is due to the strong algebraic structure of the suggested code-families, which allows the adversary to set up an algebraic equations system and solve it with Gröbner bases techniques. This kind of attack is exponential in nature and can be easily prevented by choosing more conservative parameters. However, codes that do not have any algebraic structure would completely prevent this practical threat.

Low-Density Parity Check (LDPC) codes [Gal63] are good candidates for this purpose. They are codes with no algebraic structure which meet a very simple combinatorial property: they admit a sparse parity-check matrix. Their use in cryptography has been originally analyzed in [MRS00]. Subsequently, it was followed by a sequence of works [BCG06, BCGM07, BC07] until the most recent ones [BBC08, BBC12]. In this chapter, we give the basic concepts of this type of codes and describe how they have been applied in cryptography.

Organization of Chapter 5. This chapter is organized as follows: Section 5.1 presents the definition of LDPC codes. Section 5.2 explains how to decode these codes. Section 5.3 explains the previous proposals using LDPC codes to reduce the key size of code-based cryptosystems.

5.1 LDPC codes

LDPC codes were created in the 1960's by Robert G. Gallager [Gal63]. Over 35 years, they remained almost forgotten by the coding-theory community. An exception refers to the work of Michael Tanner [Tan06], which introduced the graph interpretation for those codes. But only in 1996, David J. C. McKay (re)discovered these codes (in an apparently independent result from Gallager's work). This work has been followed by several others, attesting their excellent error correcting features and promoting them to the select group of good linear codes for telecommunication applications. This section is based on [Rya03].

LDPC codes admit two different representations.

1. **Matrix representation**. Like all linear codes, LDPC codes can be represented by a parity-check or generator matrix. Here, we will prefer the parity-check matrix representation.

2. **Graph representation**. Differently from other linear codes (for example, algebraic ones), LDPC codes admit a graph representation. This graph, also know as Tanner graph, is bipartite, composed by *check nodes* and *variable nodes*. These nodes are connected following the rule: a check node j is connected to a variable node i if the element $H_{j,i} \neq 0$, where H is the sparse parity-check matrix that defines the code.

For example, a binary linear code $(10, 4)$-$\mathcal{C}$ with sparse parity-check matrix

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

has the following associated Tanner graph

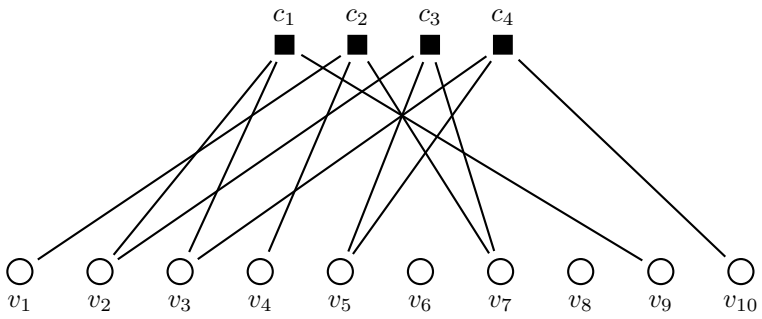

Figure 5.1: Tanner graph associated to a $(10, 4)$-LDPC code.

Tanner graphs have interesting properties that help to describe (and evaluate the quality of) LDPC codes. The *degree* of a node is the number of edges that connect this node to any other. A *cycle* of length ν in a Tanner graph is a path containing ν edges which ends on itself. The *girth* ν of a Tanner graph is the minimum cycle length of the graph. The concept of girth is important because short cycles tend to degrade the error-correction performance of LDPC codes.

Below, we give the convenient definition for this class of linear codes in terms of the sparse parity-check matrix.

Definition 5.1 (LDPC codes)**.** *An LDPC code is a linear code which admits a sparse parity-check matrix.*

- *We denote by w_r its row weight and by w_c its column weight.*
- *A* Regular *LDPC code has constant row weight.*
- *An* Irregular *LDPC code does not require constant row weight.*
- *We denote by (n, r, w)-LDPC codes the ensemble of LDPC codes of length n, co-dimension r and constant row weight w.*

For LDPC codes employed in practice, *sparse* usually means linear codes that admit a parity-check matrix of row weight less than 10. This sparsity is used by iterative decoding algorithms.

5.2 Decoding LDPC codes

The first decoding algorithm for LDPC codes was introduced in the seminal work of Gallager [Gal63]. The general idea is to compute the *a posteriori* probability that a given bit c_i of the transmitted codeword $c = [c_1, \ldots, c_n]$ is equal to 1, given the received word $y = [y_1, \ldots, y_n]$:

$$\Pr(c_i = 1 \mid y), \tag{5.1}$$

equivalently, we can consider its likelihood ratio:

$$l(c_i) = \frac{\Pr(c_i = 0 \mid y)}{\Pr(c_i = 1 \mid y)}. \tag{5.2}$$

This information is propagated across the Tanner graph from variable nodes to check nodes, and vice-versa, by the edges connecting these two types of nodes. This is done repeatedly until some stop condition is reached. Basically, each iteration has two steps, described next. To illustrate this technique, we consider a code with variable nodes of degree $w_c = 3$ and check nodes of degree $w_r = 4$.

The first step of an iteration consists of each variable node sending its most likely value for each connected check-node. To have a correct propagation effect, a variable node v_i that needs to send its most likely value to a check node c_j, will not consider the previous value contained in c_j. Figure 5.2 illustrates this step for a particular pair of variable-check nodes (v_1, c_3), but the reasoning is valid for all (v_i, c_j), when v_i is a variable node connected to the check node c_j.

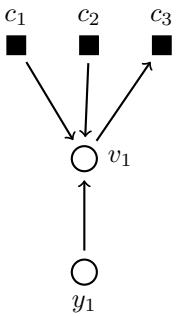

Figure 5.2: First step of an iteration for a given check and variable node.

In the second step of each iteration, the messages are passed in the opposite sense, i.e. the variable nodes send messages to the check nodes. The value a check node c_j sends to a variable node v_i consists of the value v_i should have to the check c_j sums to 0. Figure 5.3 illustrates this step for a given pair of check-variable node (c_1, v_4).

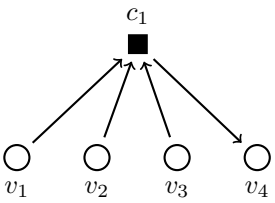

Figure 5.3: Second step of an iteration for a given check and variable node.

The idea is that these two steps are repeated until $Hc^T = 0$ be satisfied. However, sometimes the number of errors is too high (or the given code has a non sufficient error correction capability to correct). Then this process either takes too long or $Hc^T = 0$ is never accomplished. To deal with these cases, a maximum number of iterations I_{max} is previously stipulated. Then if the decoder iterates more than I_{max} times without success, the decoding algorithm stops and returns a failure symbol.

This decoding algorithm has a simple guideline implementation as described by Gallager [Gal63] for what became known as the *Bit-Flipping Algorithm*. At each iteration, the number of unsatisfied parity-check equations associated to each bit of the message is computed. Every bit associated to more than b unsatisfied equations is flipped and the syndrome is recomputed. This process is repeated until either the syndrome becomes zero or after a maximum number of iteration. It is easy to see that this algorithm has complexity $O(nwI)$, where I stands for the average number of iterations.

5.2.1 Estimating LDPC Error-Correction Capability

A weak estimation for the error correction capability of LDPC codes is provided in [Gal63]. It is based on the probability that a bit is in error after a given number of decoding algorithm iterations. When such a probability converges to zero, assuming that each bit is independent from the others, reliable error correction can be achieved.

Consider an LDPC code of length n, column weight w_c and row weight w_r. Let also P_i be the probability that a bit $\mathcal{X}$ is in error after i iterations of the decoding algorithm. Suppose that we are verifying the convergence of P_i to 0 when messages containing t errors are received. Such convergence depends on the initial probability of errors $P_0 = t/n$. When P_0 is small enough, P_i converges to zero for increasing i. Also, it depends on the required number of unsatisfied parity-checks to flip a bit, here denoted by b. The choice of b is discussed later.

Given P_0 and b, we define the probability P_i. A bit is in error after iteration i in two cases: when there was an error before i-th iteration and the algorithm did not correct it and when there was no error before iteration i but the algorithm wrongly flipped it. Without loss of generality, we denote an error by $\mathcal{X} = 1$ and no error by $\mathcal{X} = 0$. Also we denote by S_{i+1} the event when the bit is flipped at iteration $i+1$ and by $\overline{S}_{i+1}$ when the bit is not flipped at iteration $i+1$.

$$P_{i+1} = P_0 \cdot P[\overline{S}_{i+1}|\mathcal{X} = 1] + (1 - P_0) \cdot P[S_{i+1}|\mathcal{X} = 0] \tag{5.3}$$

In order to estimate the probability of the event $\overline{S}_{i+1}$, we need to estimate the probability of a parity-check be unsatisfied. This occurs when an odd number of bits is in error. Such a probability can be estimated using the following lemma.

Lemma 1 ([Gal63]). *Consider a sequence of m independent bits in each l-th bit is a 1 with probability P_l. Then the probability that an odd number of digits are 1 is:*

$$\frac{1 - \prod_{l=1}^{m}(1 + 2P_l)}{2}$$

From Lemma 1 and assuming the bits are independent, the probability r_i that an odd number of errors among $w_r - 1$ digits occur (i.e. a parity-check be unsatisfied) after iteration i is:

$$r_i = \frac{1 - (1 - 2P_i)^{w_r - 1}}{2} \tag{5.4}$$

Then we can define the inner probabilities of Equation 5.3:

$$P[\overline{S}_{i+1}|\mathcal{X} = 1] = \sum_{l=0}^{b-1} \binom{w_c - 1}{l} (1 - r_i)^l r_i^{w_c - 1 - l} \tag{5.5}$$

$$P[S_{i+1}|\mathcal{X} = 0] = \sum_{l=b}^{w_c - 1} \binom{w_c - 1}{l} r_i^l (1 - r_i)^{w_c - 1 - l} \tag{5.6}$$

Using (5.5) and (5.6) in (5.3):

$$P_{i+1} = P_0 \cdot \sum_{l=0}^{b-1} \binom{w_c - 1}{l} (1-r_i)^l r_i^{w_c - 1 - l} + (1-P_0) \cdot \sum_{l=b}^{w_c - 1} \binom{w_c - 1}{l} r_i^l (1-r_i)^{w_c - 1 - l} \tag{5.7}$$

According to [Gal63], the integer b that minimizes the probability P_{i+1} is the smallest integer b for which

$$\frac{1-P_0}{P_0} \leq \left[\frac{1+(1-2P_i)^{w_r-1}}{1-(1-2P_i)^{w_r-1}}\right]^{2b-w_c+1}. \tag{5.8}$$

Therefore, the maximum value t such that $P_i \to 0$, where $P_0 = t/n$, will provide the *threshold* from where reliable error correction can be achieved.

5.3 Previous Graph-Based Proposals for Key Size Reduction

LDPC codes have an interesting feature for code-based cryptosystems: they are free of algebraic structure. This potentially strengthens the security of code-based schemes against distinguishers and key-recovery attacks. In this section, we discuss how these codes have been used to instantiate code-based cryptosystems. We denote by Ψ_H the error correcting procedure able to correct t errors using a sparse parity-check matrix H of a linear code $\mathcal{C}$.

5.3.1 Using LDPC codes

The first work proposing LDPC code to instantiate McEliece cryptosystem was presented in [MRS00]. Sparse matrices have can be represented only by the indices of the non-zero entries. The scheme is presented in Table 5.2.

KeyGen:

1. Generate a random $r \times n$ sparse parity-check matrix H of a binary LDPC code $\mathcal{C}$ able to correct up to t.
2. Generate a sparse invertible $r \times r$ matrix T.
3. Generate a sparse invertible $k \times k$ matrix S.
4. Compute a systematic generator-matrix G obtained from $\tilde{H}$.
5. Compute $\tilde{G} \leftarrow S^{-1}G$.

– Public key: $(\tilde{G}, S, t)$, where $\tilde{H} = TH$. Private key: (H, T).

Enc: The encryption of $m \in \mathbb{F}_q^k$ is:

– Generate an error vector e of length n and weight t.
– Compute $y \leftarrow m\tilde{G} + e$.

Dec: The decryption of $y \in \mathbb{F}_q^n$ is:

– $mS^{-1}G \leftarrow \Psi_{\mathcal{C}}(y)$.
– $mS^{-1} \leftarrow$ first k entries of $mS^{-1}G$.
– $m = mS^{-1}S$.

Table 5.1: LDPC-McEliece Variant.

As explained by the authors, this scheme is insecure since the low weight parity-check equations of the private parity-check matrix H can be easily recovered. This can be done by searching for low-weight codeword in the dual of the public code. Note that this is equivalent to solve the computational decoding problem for a syndrome which is a zero-vector. Therefore, general decoding algorithms can be employed to search these low-weight codewords (see 2.3.3). The authors concluded that sparsity is not secure approach to reduce key-sizes.

5.3.2 Using Quasi-Cyclic LDPC codes

QC-LDPC McEliece #1. In [BCG06], the first use of QC-LDPC codes to instantiate the McEliece cryptosystem was presented. Actually the cryptosystem proposed is essentially the same as described above, but using a LDPC code in quasi-cyclic form. The idea was to use the quasi-cyclicity to achieve compact keys, as first suggested in [Gab05]. As expected, the cryptosystem suffers from the same weaknesses as in [MRS00].

QC-LDPC McEliece #2. In [BCGM07], the authors suggest a slightly changed version of the previous variant. They recommend the use of dense matrices T and "sufficiently dense" S, instead. However, as claimed in [BC07], this strategy is not sufficient to avoid effectiveness of the dual low weight codeword finding attack.

QC-LDPC McEliece #3. In [BC07], a new variant is proposed.

KeyGen:

1. Generate a random $r \times n$ sparse parity-check matrix with column weight d_v of a binary LDPC code $\mathcal{C}$ able to correct t errors.
2. Generate its corresponding generator matrix G in reduced echelon form.
3. Generate a sparse invertible $n \times n$ quasi-cyclic matrix Q of row weight m, as discussed below.
4. Generate a sparse invertible $k \times k$ quasi-cyclic matrix S of row weight s.

– Public key: $G' = S^{-1} \cdot G \cdot Q^{-1}$
– Private key: (S, H, Q).

Enc: The encryption of $m \in \mathbb{F}_q^k$ is:

– Generate an error vector e of length n and weight t'.
– Compute $y \leftarrow mG' + e$.

Dec: The decryption of $y \in \mathbb{F}_q^n$ is:

– $y' \leftarrow yQ = mS^{-1}G + eQ$.
– $mS^{-1}G \leftarrow \Psi_{\mathcal{C}}(mS^{-1}G + eQ)$.
– $mS^{-1} \leftarrow$ extract first k entries from $mS^{-1}G$.
– $m = mS^{-1}S$.

Table 5.2: QC-LDPC-McEliece Variant # 3.

Note that the error vector e is affected by the matrix Q leading to a propagation error of weight up to $t = t' \cdot m$. Therefore the parameters t, t' and m should be scaled in such a way that allows the decoder to efficient correct $t = t' \cdot m$ errors.

Two important details of this variant have to be stressed. The former is that the public code is not a permutation-equivalent of the private one, as seen in the McEliece cryptosystem original proposal. The second one refers to the construction of the matrix Q and the sparsity of both matrices Q and S, leading to a successful attack. The authors suggested to have Q with diagonal blocks with row/column weight m and the rest of the matrix to be zero weight blocks. Since G is in row reduced echelon form, from the public matrix $G' = S^{-1} \cdot G \cdot Q^{-1}$, an eavesdropper easily derives the following matrix:

$$G'_{\leq k} = S^{-1} \cdot \begin{bmatrix} Q_0^{-1} & 0 & \dots & 0 \\ 0 & Q_1^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q_{n_0-2}^{-1} \end{bmatrix}$$

Using the isomorphism of the polynomials mod $x - 1$ and knowing that both Q and S are sparses, it is possible to successfully attack this variant as claimed in [OTD10].

QC-LDPC McEliece #4. In [BBC08], the authors propose a dense, instead of sparse, *scrambling* matrix S. This choice prevents the eavesdropper to obtain the blocks in Q and in S, even knowing the product of these blocks. Note that a dense matrix S implies an increased cost for decoding. Regarding the matrix Q, it is proposed to avoid the block diagonal-form, since it also facilitates the work of the attacker.

This proposal has (an overestimated) key size of 49152 bits. In Section 6.4.3, we present a discussion about the actual minimum key size. Both of those variants use the same encryption and decryption processes described above in the third QC-LDPC McEliece proposal.

It is important to stress that this proposal has never been successfully attacked, but as we will see in the next Section, there are several improvements that can be adopted in order to obtain a more secure, efficient and simpler cryptosystem.

5.3.3 Lessons Learned

The first general remark on using LDPC codes in code-based cryptography refers to the key-recovery attacks which consist in searching for low weight codewords in the dual of the public code. This is a valid procedure since LDPC codes only require a sufficiently sparse parity-check matrix to perform efficient decoding. This means that any sufficiently large set of low-weight codewords can be used as a private-key.

This was the first problem reported in this research topic [MRS00] and still remains an important concern. Different approaches to hide such dual low weight codewords have been proposed [BCG06, BCGM07, BC07, BBC08].

Many of them use an auxiliary matrix of row weight m that when multiplied by the sparse parity-check matrix results in an increased dual weight codeword by a factor m. Note however this matrix equally affects the error weight, which has its weight also increased by a factor m. This obliges the private code to correct not only the errors introduced by the legitimate user, but this number increased by a factor m. This significantly restricts the chooses for selecting parameters.

Moreover, many of these attempts (except [BBC08]), were successfully attacked due to the particular structure imposed on these auxiliary matrices. Thus, it seems to be valuable to find ways to discard these auxiliary matrices, originating a scheme more efficient and, more importantly: with a simpler and secure security assessment. Ideally, we would like to have a scheme which still relies its security on the low weight codeword finding problem (a well known coding-theory problem polynomial-equivalent to decoding), but not in the degenerate case of LDPC codes, which is composed by easy instances.

Chapter 6

Moderate Density Parity-Check (MDPC) Codes

LDPC codes have an interesting feature: they are free of algebraic structure. This tends to strengthen the security of code-based cryptosystems regarding the distinguishing problem. Pure LDPC codes are not recommended to instantiate code-based cryptosystems, as discussed in Chapter 5, since their low weight parity-check equations can be easily recovered by adversaries. A possible solution consists of disguising LDPC codes with an auxiliary matrix of fixed row weight m. Thus, the dual low weight codewords would be increased by a factor m, which might be scaled to avoid this sort of attack. However, the structure imposed by this auxiliary matrix may also lead to structural weaknesses.

In this chapter, we propose a simple and powerful alternative: the use of *Moderate Density Parity-Check* (MDPC) codes. MDPC codes are LDPC codes of higher density than what is usually adopted for telecommunication applications. In general, this leads to a worse error-correction capability. However, in code-based cryptography, we are not necessarily interested in correcting many errors, but only a number of errors which ensures an adequate security level, a condition satisfied by MDPC codes.

The benefits of employing MDPC codes are many. At first, we analyze the security implications. MDPC codes are only defined by some low weight codewords existent in their dual code. Therefore it is natural to assume that the only way to distinguish such codes is by finding (or attesting the existence of) these low weight dual codewords. This is a remarkable advantage of our variant not only in comparison with compact-keys McEliece proposals, but also regarding the classical McEliece scheme instantiated with binary Goppa codes.

MDPC codes are decoded by using belief propagation techniques. This fact leads to two issues. The first one is its poor error correction performance when working with codes of higher density. The second one is regarding its probabilistic nature, *i.e.* the decoding process is susceptible to fail.

Regarding its poor error correction capability, we had to use MDPC codes of very long code-length to correct a number of errors necessary to thwart generic decoding attacks. Unfortunately, this particularity led to key-sizes larger than

the classical McEliece scheme instantiated with binary Goppa codes. To circumvent this problem, we considered the addition of a quasi-cyclic structure, leading to the family of QC-MDPC codes. This approach, on the other hand, provides extremely compact-keys: public-keys of only 4801 bits for 80-bits of security. Note that the state of the art indicates that a quasi-cyclic structure, by itself, does not imply a significant improvement for adversaries. All previous attacks on compact-key McEliece variants are based on the combination of a quasi-cyclic/dyadic structure with some *algebraic* code information, a characteristic absent in MDPC codes by design.

As for its probability of decoding failure, a non-desired feature for cryptographic applications, we managed to propose three different approaches to address this issue. The first one consists in scaling the parameters in order to make such probability negligible. The second one consists in switching to more elaborate decoding algorithms that achieve a better error-correction performance. The last one consists in requiring a CCA2-security conversion, permitting one to request a new encryption in case of failure without leaking private information.

A side note: the terminology *MDPC* has appeared before in the communications literature for the same concept [OB09], but applied in different scenarios and for different purposes. In summary, the authors showed that certain quasi-cyclic MDPC codes may perform well at moderate lengths for correcting a rather large number of errors by using a variation of the belief propagation decoding technique, taking advantage of the quasi-cyclic structure.

Organization of Chapter 6. The remainder of this chapter is organized as follows. Section 6.1 defines Moderate Density Parity-Check codes. Section 6.2 explains the construction of MDPC codes and the setup for the new MDPC/QC-MDPC McEliece variants. Section 6.3 encompasses the decoding aspects of our proposal. Section 6.3.1 discusses how we estimate the error correction capability of MDPC codes. Section 6.3.2 explains a variant of bit-flipping decoding algorithm that is suitable for MDPC codes. Section 6.3.3 gives three different approaches to deal with decoding failures. Section 6.4 discusses the algorithmic and storage complexity of our proposal. Section 6.5.2 assesses the security of our proposal. Section 6.5.1 present a security-reduction proof that, under a reasonable assumption, shows that our proposal relies its security only on the decoding problem. Section 6.5.2 assesses the practical security of our proposal. Section 6.6 present our suggested parameters.

Publications presented in Chapter 6. The results concerning MDPC codes led to a long, pre-print paper and a short, published paper.

1. [MTSB12]: Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo L.S.M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. http://eprint.iacr.org/2012/409. Submitted preprint, 2012.

2. [MTSB13]: Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo L.S.M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In IEEE International Symposium on Information Theory – ISIT'2013, pages 2069–2073, Istambul, Turkey, 2013.

6.1 Preliminaries

We start by defining MDPC codes, which differ from LDPC codes simply by the magnitude of the row weight of the sparse parity-check matrix. Recall that LDPC codes have small constant row weights (usually less than 10).

Definition 6.1 (MDPC codes)**.** *An (n, r, w)-MDPC code is a linear code of length n, codimension r which admits a parity-check matrix of a constant row weight $w = \tilde{\mathcal{O}}(\sqrt{n})$.*

- *When quasi-cyclic, we call it an (n, r, w)-QC-MDPC code.*

6.2 Explaining the Construction

The construction of both MDPC and QC-MDPC codes is quite simple and basically boils down to the selection of random vectors of a given weight.

MDPC code construction.

A random (n, r, w)-MDPC code is easily generated by picking a random parity-check matrix $H \in \mathbb{F}_2^{r \times n}$ of row weight w. With overwhelming probability this matrix is of full rank and the rightmost $r \times r$ block is always invertible after possibly swapping a few columns.

QC-MDPC code construction.

We are specially interested in (n, r, w)-QC-MDPC codes where $n = n_0 p$ and $r = p$. This means that the parity-check matrix has the form

$$H = [H_0 | H_1 | \dots | H_{n_0-1}],$$

where H_i is a $p \times p$ circulant block.

We define the first row of H picking a random vector of length $n = n_0 p$ and weight w. The other $r - 1$ rows are obtained from the $r - 1$ quasi-cyclic shifts of this first row. Each block H_i will have a row weight w_i, such that $w = \sum_{i=0}^{n_0-1} w_i$. In general, a smooth distribution is expected for the sequence of w_i's.

A generator matrix G in row reduced echelon form can be easily derived from the H_i's blocks. Assuming the rightmost block H_{n_0-1} is non-singular (which particularly implies w_{n_0-1} odd, otherwise the rows of H_{n_0-1} would sum up to 0), we construct a generator-matrix as follows.

$$G = \left[\begin{array}{c|c} \mathrm{I} & \begin{array}{c} (H_{n_0-1}^{-1} \cdot H_0)^T \\ (H_{n_0-1}^{-1} \cdot H_1)^T \\ \vdots \\ (H_{n_0-1}^{-1} \cdot H_{n_0-2})^T \end{array} \end{array}\right]$$

MDPC/QC-MDPC McEliece variant

As described in Chapter 1, McEliece-like cryptosystems do not need any auxiliary permutation and scrambling matrices. Moreover, due to the use of CCA-2 security conversion, it is possible to have the public generator matrix in systematic form. Thus our variants have as secret key the sparse parity-check matrix $H \in \mathbb{F}_2^{r\times n}$ and as public key the generator matrix $G \in \mathbb{F}_2^{k\times n}$ in systematic form.

KeyGen:

1. Generate a sparse parity-check matrix $H \in \mathbb{F}_2^{r\times n}$ of a (n, r, w)-MDPC code $\mathcal{C}$ equipped with a t-error-correcting procedure $\Psi_{\mathcal{C}}$, as described above.
2. Generate its corresponding generator matrix $G \in \mathbb{F}_2^{k\times n}$ in systematic form.

– Secret-Key: H.
– Public-Key: G.

Enc: The encryption of $m \in \mathbb{F}_q^k$ is:

1. Select at random a vector $e \in \mathbb{F}_q^n$ of weight t.
2. $x \leftarrow mG + e$.

Dec:The decryption of $x \in \mathbb{F}_q^n$ is:

1. $m \leftarrow \Psi_{\mathcal{C}}(x)G^{-1}$.

Table 6.1: MDPC-McEliece Variant.

6.3 Decoding MDPC codes

For decoding MDPC codes, we naturally suggest the use of LDPC decoding techniques. This brings two main challenges. The first one regards the error-correction performance of MDPC codes. LDPC decoding techniques strongly depends on the very sparse parity-check matrix, and thus a degradation should be noticed when moving from LDPC to MDPC codes. However, in cryptography, we are not necessarily interested in correcting a large number of errors, but only a number which ensures an adequate security level. The second challenge refers to the inherent property of probabilistic decoding, namely the chance of having decoding failures, and this situation must be addressed for applications in cryptography. Next, we discuss how to deal with these two challenges.

To star our discussion, we precise that we will use a variant of Gallager's bit flipping decoding algorithm [Gal63]. This iterative decoding algorithm provides an error-correction capability which increases linearly with the code-length and decreases nearly linearly when the weight of the parity-checks increases, thus explaining the degradation in the error-correcting capability when moving from LDPC, where $w = \mathcal{O}(1)$, to MDPC codes, where $w = \tilde{\mathcal{O}}(\sqrt{n})$ using a soft notation that omits logarithmic factors.

6.3.1 Error-Correction Capability

In Section 5.2.1, we discussed the Gallager's analysis for the error-correction capability of the bit flipping algorithm [Gal63] applied on LDPC codes. This analysis is valid for increasing code-length and when each variable node is considered independent from the others. In practice, it provides a satisfactory performance description for long codes without short cycles in the Tanner graph.

MDPC codes are randomly constructed and therefore do not prevent the presence of short cycles in the associated Tanner graph. Actually, since the density is reasonably increased, these codes most likely have such short cycles, and thus the aforementioned analysis is not as precise as for LDPC codes. So far, codes having such short cycles have never attracted much attention due to their restrictive or none practical applicability, thus limiting the literature focused on estimating their performance.

For this reason, we propose a simple method to estimate the performance of bit-flipping algorithm applied on (n, r, w)-MDPC codes. It is divided in two steps. The first one consists of using the Gallager's analysis to estimate an *upper-bound* on the error correction capability. In Section 5.2.1, this upper-bound is called the *threshold* for achieving reliable decoding. The second step consists of estimating the quality of these codes in correcting a given number of errors. This is done by performing exhaustive simulation, providing the *decoding failure rate* (DFR), *i.e.* the fraction of decoding failures among a reasonably large number of decoding tests and different random (n, r, w)-MDPC codes.

In fact, we will use a slightly changed version of bit-flipping algorithm. It is inspired on the bit-flipping algorithm as described in [HP03], which slightly improves the error correction capability by relaxing the rule to flip a bit.

In summary, for choosing the number of errors our codes must correct, we start by considering the upper bound provided by Gallager's analysis and then we perform exhaustive simulation for computing its DFR. We proceed by decreasing the number of errors until reaching an adequate DFR. Using this approach, we validate that the parameters of Section 6.6 reach a DFR below 10^{-7}.

6.3.2 Bit-Flipping Variant Suitable for MDPC Codes

We recall the Gallager's bit flipping algorithm. At each iteration, the number of unsatisfied parity-check equations associated to each bit of the message is computed. Each bit associated to more than b unsatisfied equations is flipped and the syndrome is recomputed. This process is repeated until either the syndrome becomes zero or after a maximum number of iteration. This algorithm has complexity $O(nwI)$, where I stands for the average number of iterations.

Due to the increased row weight and to the existence of short-cycles in the corresponding Tanner graph, MDPC codes may lead to an increased number of iterations. To minimize this problem, we suggest a modification for choosing b. Below a few possibilities for this choice, followed by our approach:

I. Given $P_0 = t/n$ and using Equation (5.7), precompute b as the smallest integer b for which:

$$\frac{1 - P_0}{P_0} \leq \left[\frac{1 + (1 - 2P_i)^{w-1}}{1 - (1 - 2P_i)^{w-1}}\right]^{2b-w+1}$$

This is the method proposed by Gallger, as discussed in Section 5.2.1.

II. In [HP03], at each iteration, b is chosen as the maximum number of unsatisfied parity-check equations, here denoted by Max_{upc}.

III. Our approach is: $b = \text{Max}_{\text{upc}} - \delta$, for a small integer δ. In the case of a decoding failure, we decrease the value of δ by 1 and restart the process.

The main feature of each approach is: Approach I uses an estimation for b and therefore avoids its computation at each iteration. Approach II is more general than I, leading to a better error-correcting capability at the price of an increased number of iterations. Finally, Approach III combines the benefits from I and II:

- It reduces the overall number of iterations obtained by Approach II, since much more bits are flipped at each iteration.
- In the case of a decoding failure, we suggest to decrease the value of δ by 1 and restart the process. Thus, after δ decoding failures, we have $\delta = 0$ and we are back to Approach II ensuring at least its error-correcting capability.

The optimal initial value for δ is determined empirically. For the parameters suggested in Section 6.6, a good choice is $\delta \approx 5$, reducing the number of iterations from ~ 65 to less than 10.

6.3.3 Failure Decoding

As discussed above, MDPC codes (like any other code that use probabilistic decoding techniques) suffer from a non-zero decoding failure probability. In cryptography, this must be treated. Next we present three approaches to deal with this problem.

Conservatively choosing the parameters: A straightforward approach consists of conservatively choosing the number of errors so that the decoding failure rate becomes negligible. For example, a common approach in error-correcting systems consists of using codes whose DFR is smaller than the machine failure rate where the system is deployed.

Switching to more sophisticated decoding algorithms: A second approach deals with these unlikely events on the fly. In the case of a decoding failure, more sophisticated decoding algorithms with better error correction capability can be used, e.g. [HOP96]. Note however that this comes at the price of a significantly increased decoding complexity.

Requesting a new encryption: When the application allows, a third approach consists of using a CCA-2 security conversion, e.g. [KI01]. In short, a CCA2-security conversion uses hash functions and random sequences to ensure the indistinguishability of the encrypted messages. Thus, in the case of a decoding failure, new encryptions can be requested. Since the encrypted messages behave like random sequences, the adversary cannot extract information from this redundancy.

6.4 Efficiency

In this section, we discuss the efficiency of our approach in terms of both algorithmic and storage complexity.

6.4.1 Algorithmic Complexity

In this section, we discuss about the algorithmic complexity for code-generation, encoding and decoding using our codes.

Key-Generation: The key-generation process is mainly based on the selection of random vectors of a given weight. therefore, the main cost in the key-generation process is reduced to the cost of obtaining the systematic generator matrix from the sparse parity-check matrix. The quasi-cyclic variant has an advantage since its cyclic blocks can be manipulated as polynomials in the ring $\mathbb{F}_2[x] \bmod (x^p - 1)$.

Encryption: The encryption depends on a matrix-vector product and an vector addition. For the quasi-cyclic variant, these operations have linear complexity, whilst for the standard MDPC variant, the matrix-vector product has quadratic complexity.

Decryption: As explained before, the Gallager's bit-flipping algorithm has complexity $O(nwI)$, where I stands for the average number of iterations. Note that due to the approach presented in 6.3, we managed to reduce the average number of iterations to less than 10.

6.4.2 Storage Complexity

In this section, we discuss about the cost of storing both private and public information regarding our codes. In summary, the private key consists of the sparse parity check matrix of the code, and the public key consists of the generator matrix of the code. The cost of storage will depend on each variant: QC-MDPC or standard MDPC codes.

Private-key: For the standard variant, the private key will need $r \times n$ bits. For the quasi-cyclic variant, the private key will need n bits.

Public-key: For the standard variant, the public key will need $k \times r$ bits. For the quasi-cyclic variant, the public key will need k bits.

	MDPC variant	QC-MDPC variant
Private key	$r \times n$	n
Public key	$k \times r$	k

Table 6.2: Key-sizes of MDPC and QC-MDPC variant.

As we can see in Table 6.2, the MDPC variant will provide huge keys, whilst the QC-MDPC variant provides very compact keys. Also note that the private description, here described in terms of matrices can be improved depending on how sparse the matrices are. For example, a binary vector of length n with only $w \ll n$ non-zero positions could be more efficiently stored by the indices of this non-zero positions, rather than the whole vector n. This will depend on the storage required for each index. The same holds for a sparse matrix.

6.4.3 Comparison with the QC-LDPC McEliece Variant

In this section, we show that our variant permits the legal user to introduce more errors than the QC-LDPC McEliece proposal presented in [BBC08], using codes of same size. At first, we recall this QC-LDPC McEliece variant.

- Private-key:
  - Parity-check matrix $H \in \mathbb{F}_2^{r\times n}$ of a code $\mathcal{C}$, of column weight d_c and row weight d_r.
  - Sparse matrix $Q \in \mathbb{F}_2^{n\times n}$ of row weight m.
  - Dense matrix $S \in \mathbb{F}_2^{r\times r}$.
- Public-key: Generator matrix $G' \leftarrow S^{-1}GQ^{-1}$, where G is a generator matrix for $\mathcal{C}$.

Thus, the dual of the public-code contains codewords of weight md_r. An user that wants to encrypt a message $m \in \mathbb{F}_2^{(n-r)}$ must compute $y \leftarrow mG' + e$, where $e \in \mathbb{F}_2^n$ is a vector of weight t. The decryption of y starts with a multiplication of y by Q, *i.e.* : $y' \leftarrow yQ = mS^{-1}G + eQ$. Then the legal receiver user must correct all errors present in y'. Note that the errors in y' are not limited to t anymore, they were increased by a factor m, i.e. $\mathtt{wt}(y') = mt$. Therefore the user must correct m times more errors than the number of errors purposely introduced by the sender. Apart the fact that this situation represents a restrictive requirement for selecting parameters, it also introduces an important disadvantage when compared to our proposal.

- Using the same public code, our scheme permits to introduce more errors than in [BBC08].

This comes from the fact that an LDPC code with parity-check matrix of row weight increased by a linear factor m has its error correction capability degraded by less than a factor m. In this sense, if the scheme was based solely on the capability of its public code, the users will be able to introduce more errors than in the aforementioned scheme.

We can illustrate this discussion by using their own parameters. Let $n = 12288$, $d_v = 13$, $d_c = 39$ and $m = 7$, as proposed in [BBC08] for 80-bits of security. Then the scheme uses a private code whose dual $\mathcal{C}$ contains codewords of weight 39, whilst the dual public code $\mathcal{C}'$ contains codewords of weight $39 \cdot 7 = 273$. We estimate the threshold for reliable decoding as explained in Section 5.2.1 for both codes $\mathcal{C}$ and $\mathcal{C}'$. For $\mathcal{C}$, the threshold obtained with this technique is 207. However, in principle, we are restricted by the design of the scheme to correct only $1/7$ of this, which corresponds to 30. On the other hand, the threshold for H' is 50 errors, which represents an improvement of 66% more errors. This was the initial motivation to use LDPC codes of increased density.

6.4.4 Implementing QC-MDPC McEliece

In [HMG13], the authors present an implementation of the QC-MDPC McEliece variant for embedded devices. More precisely, the platform used was a reconfigurable hardware Xilinx Virtex-6 FPGA and an embbeded microcontroller Atmel

AVR ATxmega (which is a popular low-cost 8-bit microcontroller). Their implementation used the parameters set for achieving 80-bits of security, with key-sizes under 5000 bits of length. This implementation demonstrated the excellent practical applicability of our proposal, achieving much smaller storage requirement than previous McEliece implementations. Moreover, it is faster and it requires less storage requirement than RSA.

6.4.5 Recent Construction Related to Our Proposal

In [GMRZ13], the authors introduce the family of Low-Rank Parity-Check (LRPC) codes with an efficient decoding algorithm. It is also analyzed their use to instantiate code-based cryptosystems. According to the authors, their scheme is the equivalent to the MDPC proposal in the rank-metric. This proposal achieves very compact public keys of only 1517 bits for a security of 2^{80}. Moreover, they can made the decryption failure rate arbitrarily small.

Note that rank-metric codes require more complex operations and they have been much less studied than codes in Hamming metric for cryptography. Before the aforementioned variant, they have only appeared in [GPT91] to instantiate the McEliece scheme, but being subsequently broken due to the strong algebraic structure present in this particular variant.

6.5 Security Assessment

The security assessment of code-based schemes instantiated with MDPC/QC-MDPC codes is divided in two parts. In the former, we adapt the generic security-reduction proof presented in Chapter 2 for schemes instantiated with our codes. In the second part, we estimate the practical security of our proposal.

6.5.1 Security-Reduction Proof

In this section, we adapt the security reduction of Chapter 2 to a Niederreiter scheme instantiated with our codes. All the statements in this section are valid in both MDPC and QC-MDPC cases.

The main result here is: under the reasonable assumption that MDPC codes can only be distinguished by finding their dual low weight codewords, the distinguishing problem is reduced to the problem of finding low weight codewords, which is problem polynomially equivalent to the decoding problem. We will use the following problems.

Problem 1 (Code Distinguishing Problem).
Parameters: $\mathcal{K}_{n,k}$, $\mathcal{H}_{n,k}$.
Instance: a matrix $H \in \mathcal{H}_{n,k}$.
Question: is $H \in \mathcal{K}_{n,k}$*?*

Problem 2 (Computational Syndrome Decoding Problem).
Parameters: $\mathcal{H}_{n,k}$, *an integer* $t > 0$.
Instance: a matrix $H \in \mathcal{H}_{n,k}$ *and a vector* $s \in \mathbb{F}_2^r$.
Problem: find a vector $e \in \mathcal{S}_n(0,t)$ *such that* $eH^T = s$.

Problem 3 (Codeword Existence Problem)**.**
Parameters: $\mathcal{H}_{n,k}$, *an integer* $w > 0$.
Instance: a matrix $H \in \mathcal{H}_{n,k}$.
Question: is there a codeword of weight w *in the code of generator matrix* H*?*

Problem 4 (Codeword Finding Problem)**.**
Parameters: $\mathcal{H}_{n,k}$, *an integer* $w > 0$.
Instance: a matrix $H \in \mathcal{H}_{n,k}$.
Problem: find a codeword of weight w *in the code of generator-matrix* H.

Problems 1 and 2 were already used in Chapter 2. We introduced Problem 3, which consists of deciding the existence of words of given weight in a given linear code, and Problem 4, which is its computational version. Note that in these two last problems, we consider a code that has a *generator matrix* $H \in \mathcal{H}_{n,k}$.

We adapt the previous notation to our codes.

- $\mathcal{F}_{n,k}$: Family of $(n, n-k, w)$-(QC)-MDPC codes able to correct t errors.
- $\mathcal{K}_{n,k}$: Public-key space of $\mathcal{F}_{n,k}$ composed by parity-check matrices.
- $\mathcal{H}_{n,k}$: The apparent public-key space of $\mathcal{F}_{n,k}$. More precisely:

Note that $\mathcal{K}_{n,k} \subset \mathcal{H}_{n,k}$. Moreover, the apparent public-key space depends on which code-family is used.

- MDPC codes:
  - $\mathcal{H}_{n,k}$: Set of all full rank matrices in $\mathbb{F}_2^{r \times n}$.
- QC-MDPC codes:
  - $\mathcal{H}_{n,k}$: Set of all full rank block circulant matrices in $\mathbb{F}_2^{r \times n}$.

Ideally, we would like to replace the distinguishing problem by Problem 3 in Proposition 2.1. Unfortunately, one would need to replace the distinguisher advantage by the quantity:

$$Adv(\mathcal{E}, \mathcal{K}_{n,k}) = |\Pr_{\Omega}[\mathcal{E}(H) = 1 \mid H \in \mathcal{K}_{n,k}] - \Pr_{\Omega}[\mathcal{E}(H) = 1]|,$$

where $\mathcal{E}$ denotes a program deciding the existence of a word of weight w in a given code. However this quantity is not directly related to the hardness of Problem 3 and therefore cannot be considered. Nevertheless we reach our purpose if we assume the following assumption.

Assumption 1. *Solving Problem 1 for parameters* $(\mathcal{H}_{n,k}, \mathcal{K}_{n,k})$ *is not easier than solving Problem 3 for the parameters* $(\mathcal{H}_{n,k}, w)$.

Within this assumption we could modify the reduction to a claim that the $\mathcal{K}_{n,k}$-McEliece scheme is at least as hard as either Problem 2 and Problem 3.

Proposition 6.1. *Given Assumption 1:*

- *Breaking the MDPC variant of McEliece or Niederreiter is not easier than solving the easiest problem between the syndrome decoding problem and the codeword existence problem for a random linear code.*
- *Breaking the QC-MDPC variant of McEliece or Niederreiter is not easier than solving the easiest problem between the syndrome decoding problem and the codeword existence problem for a random quasi-cyclic linear code.*

Proof. This follows directly from Proposition 2.1. □

However, we can do much better when the parameter w of Problem 3 is smaller than the minimum distance of the code of parity check matrix H of Problem 2. In this scenario, we can prove that these two problems era polynomially equivalent, *i.e.* they can be reduced in polynomial time to each other.

To prove this equivalence, we start by showing that the codeword existence problem is polynomially equivalent to the codeword finding problem. Then we sow that the codeword finding problem is polynomially equivalent to the computational syndrome decoding problem.

Lemma 2. *Problem 3 is polynomially equivalent to Problem 4.*

Proof.

I - Reducing Problem 3 to Problem 4:

An algorithm that solves Problem 4 obviously provides an answer to Problem 3.

II - Reducing Problem 4 to Problem 3:

Let $\mathcal{G}_{n,k}$ denote a subset of $\mathbb{F}_2^{k\times n}$ composed by full rank matrices.

A matrix $G \in \mathcal{G}_{n,k}$ is the generator matrix of a (n,k)-linear code $\mathcal{C}$. For any $1 \leq i \leq n$, we denote $\mathcal{C}_i$ the code shortened at i, that is

$$\mathcal{C}_i = \{c = (c_1, \ldots, c_n) \in \mathcal{C} \mid c_i = 0\}.$$

We will denote by G_i a generator matrix of $\mathcal{C}_i$. We assume we have a program $\mathcal{E}$ that solves Problem 3, that is $\mathcal{E} : \mathcal{G}_{n,k} \to \{0,1\}$ such that $\mathcal{E}(G) = 1$ if and only if there exists a word of weight w in the code spanned by G. The following program called on input G such that $\mathcal{E}(G) = 1$ will return a word of weight at most w in the code spanned by G.

$\mathcal{A}$: input $G \in \mathcal{G}_{n,k}$
  `for` i `from` 1 `to` n `while` G has a rank > 1
    `if` $\mathcal{E}(G_i) = 1$ `then` // *false at most* w *times*
      $G \leftarrow G_i$
  `return` the first row of G of weight at most w

Note that Program $\mathcal{E}$ is called at most n times.

□

Lemma 3. *Problem 4 is polynomially equivalent to Problem 2.*

Proof.

I - Reducing Problem 4 to Problem 2:

Assume that we have a program $\mathcal{A}$ which solves the computational syndrome decoding problem for parameters $(\mathcal{H}_{n,k+1}, w)$

$\mathcal{B}$: input $H \in \mathcal{H}_{n,k}$
  $(g_1, \ldots, g_k) \leftarrow$ a basis of $\mathcal{C}$ // *where $\mathcal{C}$ is the code of parity check matrix H*
  `for` j `from` 1 `to` n
    $H' \leftarrow$ parity check matrix of $\bigoplus_{i \neq j} \langle g_i \rangle$ // *subcode of $\mathcal{C}$ without g_j*
    `if` $\mathcal{A}(H', g_j H'^{\mathrm{T}}) \neq$ FAIL then
      $z \leftarrow \mathcal{A}(H', g_j H'^{\mathrm{T}})$
      `return` $z + g_j$
  FAIL // *$\mathcal{A}$ fails to decode for all j*

If exists a codeword of weight w, the decoder $\mathcal{A}$ will succeed for at least one value of j. $\mathcal{B}$ provides a solution to Problem 4 for parameters $(\mathcal{H}_{n,k}, w)$.

II - Reducing Problem 2 to Problem 4:

Assume that we have a program $\mathcal{B}$ which solves the Problem 4 for parameters $(\mathcal{H}_{n,k+1}, w)$, we define the following program

$\mathcal{A}$: input $H \in \mathcal{H}_{n,k}$, $s \in \mathbb{F}_2^r$
  $y \leftarrow$ solve $Hy = s$. // *No restriction over the weight of y.*
  $G \leftarrow$ a generator matrix of the code of parity-check H.
  $G' \leftarrow \begin{bmatrix} G \\ y \end{bmatrix}$ // *y is the $(k+1)$-th row of G'.*
  `return` $\mathcal{B}(G')$

Note that w should be smaller than the minimum distance of the code of parity check matrix H in order to the call $\mathcal{A}(H, s)$ never fail. This provides a solution to Problem 2 with parameters $(\mathcal{H}_{n,k}, w)$.

□

Within Assumption 1, Lemma 2 and Lemma 3, and assuming that the number of errors is smaller than the minimum distance of the code, we are able to produce strong security statements.

Proposition 6.2. *Given Assumption 1:*

- *Breaking the MDPC variant of McEliece or Niederreiter is not easier than solving the syndrome decoding problem for a random code.*
- *Breaking the QC-MDPC variant of McEliece or Niederreiter is not easier than solving the syndrome decoding problem for a random quasi-cyclic linear code.*

Proof. This follows directly from Proposition 2.1 and the polynomial equivalence of problems 3–4 (Lemma 2) and 4–2 (Lemma 3).

□

6.5.2 Practical Security

In this section, we analyze the practical attacks against the proposed scheme. Consider the McEliece (or Niederreiter) scheme with an (n, r, w)-MDPC code $\mathcal{C}$, possibly quasy-cyclic, correcting t errors. The public-key is a generator matrix of $\mathcal{C}$ for McEliece or a parity-check matrix of $\mathcal{C}$ for Niederreiter. We claim that the best attacks for each context are:

- *Key distinguishing attack:* Exhibit one codeword of $\mathcal{C}^\perp$ of weight w.
- *Key recovery attack:* Exhibit r codewords of $\mathcal{C}^\perp$ of weight w.
- *Decoding attack:* Decode t errors in a $(n, n-r)$-linear code.

For all those attacks we have to solve either the codeword finding problem or the computational syndrome decoding problem. For both problems and for the considered parameters the best technique is information set decoding (ISD). ISD workfactors are commonly used to the security assessment of code-based schemes. However our proposal requires an extra caution.

The problem of finding a *single* low weight codeword in an MDPC code may admit *many* solutions.

We denote by $\mathrm{WF}_{\mathrm{isd}}(n, r, t)$:

- Cost of decoding t errors in an (n, r)-binary linear code;
- Cost of finding a codeword of weight t in an (n, r)-binary linear code when there is a single solution of the problem.

As discussed in Chapter 2, ISD algorithms assume a pattern for the error vector. It analyzes a certain set of candidates until a solution be found. This set of candidates is usually stored in lists of a certain size L and each candidate has a probability P to produce the solution. When the algorithm parameters are *optimal*, up to a small factor, it holds that

$$\mathrm{WF}_{\mathrm{isd}}(n, r, t) \approx L/P.$$

In [Sen11], the *Decoding One Out of Many* (*DOOM*) setting is presented. In this work, the author analyzes the gains when the decoding problem have multiple solutions and the adversary is satisfied in finding a single one.

In short, when the problem has N_s solutions, the probability of success P increases by a factor N_s (as long as $N_s P \ll 1$) and when N_i instances are treated simultaneously the list size L increases at most by a factor $\sqrt{N_i}$. Therefore:

The *DOOM* technique provides a gain[1] of $N_s/\sqrt{N_i}$.

[1] In fact, the real gain might be fact slightly smaller, since these algorithms depend on optimal parameters which might not be the same for multiple instances.

Key Distinguishing Attack. We want to estimate the cost $\mathrm{WF}_{\mathrm{dist}}(n, r, w)$ of distinguishing our codes, *i.e.* of producing one word of weight w in the dual code $\mathcal{C}^{\perp}$. In this context, an adversary applying ISD to the all-zero syndrome will face a problem with r solutions (the r rows of the sparse parity-check matrix). Then $N_s = r$ and $N_i = 1$ and the distinguishing attack cost drops by a factor of r:

$$\mathrm{WF}_{\mathrm{dist}}(n, r, w) = \frac{\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)}{r}.$$

In the quasi-cyclic case, there is no obvious speedup and the distinguishing attack has the same cost as above.

Key Recovery Attack. We want to estimate the cost $\mathrm{WF}_{\mathrm{reco}}(n, r, w)$ of recovering the private code, *i.e.* of producing r codewords of weight w in the dual code $\mathcal{C}^{\perp}$. All ISD variants are randomized and thus we can make r independent calls to a codeword finding algorithm. Each call costs on average $\frac{\mathrm{WF}_{\mathrm{isd}}(n,n-r,w)}{r}$ since there are r codewords of weight w. Therefore on average, recovering all equations will cost:

$$\mathrm{WF}_{\mathrm{reco}}(n, r, w) = r \cdot \frac{\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)}{r} = \mathrm{WF}_{\mathrm{isd}}(n, n-r, w).$$

In the quasi-cyclic case, any word of low weight will provide the sparse matrix and thus the key recovery attack is no more expensive than the key distinguishing attack.

$$\mathrm{WF}^{\mathrm{QC}}_{\mathrm{reco}}(n, r, w) = \mathrm{WF}^{\mathrm{QC}}_{\mathrm{dist}}(n, r, w) = \frac{\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)}{r}.$$

Decoding Attack. We want to estimate the cost $\mathrm{WF}_{\mathrm{dec}}(n, r, t)$ of decoding t errors. In the MDPC case, it holds that:

$$\mathrm{WF}_{\mathrm{dec}}(n, r, t) = \mathrm{WF}_{\mathrm{isd}}(n, r, t).$$

In the quasi-cyclic case, any cyclic shift of the target syndrome $s \in \mathbb{F}_2^r$ provides a new instance whose solution is equal to the original, up to a block-wise cyclic shift. Thus the number of instances and solutions are $N_i = N_s = r$. Therefore a factor $\sqrt{r}$ is gained:

$$\mathrm{WF}^{\mathrm{QC}}_{\mathrm{dec}}(n, r, t) \geq \frac{\mathrm{WF}_{\mathrm{isd}}(n, r, t)}{\sqrt{r}}.$$

	MDPC	QC-MDPC
Key distinguishing	$\frac{1}{r}\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)$	$\frac{1}{r}\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)$
Key recovery	$\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)$	$\frac{1}{r}\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)$
Decoding	$\mathrm{WF}_{\mathrm{isd}}(n, r, t)$	$\frac{1}{\sqrt{r}}\mathrm{WF}_{\mathrm{isd}}(n, r, t)$

Table 6.3: Best attacks against MDPC (or QC-MDPC) codes

Choosing p as a prime number. We recommend to select blocks of prime length as a precaution to prevent possible attacks that exploit the symmetry of codewords in a quasi-cyclic code [FL08, Loi13].

6.6 Suggested Parameters

In this section, we explain how to choose secure and efficient parameters. The idea is to produce a code from a given code-rate R and decoding failure rate. Initially we start with a code-length n and dimension $k = Rn$. At first, we compute the minimum integer t such that $\mathrm{WF}_{\mathrm{isd}}(n, k, t) > 2^{80}$. Then, we compute the minimum integer w such that $\mathrm{WF}_{\mathrm{isd}}(n, n-k, w) > 2^{80}$. Finally, we verify if t errors is under the threshold from where reliable error correction can be achieved (see discussion in 6.3.1 and 5.2.1). If so, we verify by exhaustive simulation if random codes of same parameters attain the desired decoding failure rate. If these codes cannot correct t errors or they do not attain the desired decoding failure rate, the process is restarted with an increased code-length n. In practice, this process converges very quickly for code-lengths of a few thousands.

In Table 6.4, we present some parameters obtained from this method for our quasi-cyclic variant, the most relevant for practical applications. For each security level, we propose three parameter sets ($n_0 = 2$, $n_0 = 3$ and $n_0 = 4$), leading to different code rates (1/2, 2/3, 3/4, respectively). The column r also gives the syndrome size in bits.

Security. The security assessment is based on the workfactor of the ISD variant [BJMM12] decreased by the possible gains of the *DOOM* setting [Sen11].

Example: Let $n_0 = 2$, $n = 9602$, $r = 4801$, $w = 90$, $t = 84$. The ISD cost for key-recovery is $\mathrm{WF}_{\mathrm{isd}}(n, n-r, w) = 2^{92.70}$ and for message-recovery is $\mathrm{WF}_{\mathrm{isd}}(n, r, t) = 2^{87.16}$. Decreasing it by the gains of *DOOM*, a factor of 4801 and $\sqrt{4801}$, respectively, the final workfactors are $2^{80.47}$ and $2^{81.04}$.

Decoding. These QC-MDPC codes attain decoding failure rates below 10^{-7}, using the bit-flipping variant presented in Section 6.3.2.

Security	n_0	n	r	w	t	Key-size
80	2	9602	4801	90	84	4801
80	3	10779	3593	153	53	7186
80	4	12316	3079	220	42	9237
128	2	19714	9857	142	134	9857
128	3	22299	7433	243	85	14866
128	4	27212	6803	340	68	20409
256	2	65542	32771	274	264	32771
256	3	67593	22531	465	167	45062
256	4	81932	20483	644	137	61449

Table 6.4: Suggested parameters. Syndrome and key-size given in bits.

Part III

Synopsis

Chapter 7

Conclusion

Code-based cryptosystems have been neglected to a second-class of cryptographic solutions for more than thirty years. Although efficient and secure against quantum attacks, its drawback concerning its huge keys never permitted to be considered for practical applications. In this thesis, we presented two different approaches that significantly overcome this problem.

The first one uses algebraic codes. We presented the p-adic Goppa codes, which are Goppa codes that admit compact representation. These codes can be used to instantiate public-key encryption schemes, like McEliece or Niederreiter schemes. With a simple modification in their generator algorithm, we showed that these codes are also suitable to instantiate the Parallel-CFS digital signature scheme. Finally, we demonstrated that these codes can be generalized to codes defined over any characteristic $p \geq 2$. In summary, we managed to produce very compact keys based on the reputable family of Goppa codes.

Although never attacked in practice, generic Goppa codes have recently inspired some distrust. In [FGO+11], a distinguisher for high-rate Goppa codes was presented. A distinguisher does not lead necessarily to a practical attack, but it might be seen as an alert. Distinguishing the presence of private information (in this case, its algebraic structure) is the first step on obtaining such information.

Although efficient, our p-adic Goppa proposal suffers from an significant undesired feature: strong algebraic structure. This fact motivated our second approach, using LDPC codes of increased density, or simply MDPC codes. These are graph-based codes, which are free of algebraic structure. The only relevant way to distinguish such codes is by finding low weight codewords in their dual code. Note that for certain parameters this problem is polynomially equivalent to decoding. This is an important advantage of an MDPC-McEliece variant not only in comparison to the previous attempts to reduce code-based keys but also regarding the classical McEliece based on binary Goppa codes. Moreover, adding a quasi-cyclic structure, it was possible to achieve very compact keys. Note that the state of the art indicates that a quasi-cyclic structure, by itself, does not imply a significant improvement for adversaries. All previous attacks on compact-key McEliece variants are based on the combination of a quasi-cyclic/dyadic structure with some *algebraic* code information, property absent in MDPC codes by design.

7.1 Algebraic *vs.* Graph-based approach

As discussed above, the MDPC proposal has an important advantage in comparison to the p-adic proposal: they are free of algebraic structure. Actually, this feature is an advantage even compared to the classical McEliece setup using generic binary Goppa codes.

On the other hand, MDPC codes impose important restrictions for their practical implementation. The first issue refers to the decoding step. MDPC codes are decoded by using probabilistic algorithms. This means that there is a probability of failure for the decoding process. By conservatively selecting parameters, we can decrease the decoding failure probability, yet never completely vanish it. To address this issue, we propose three different approaches for dealing with this problem. In short, these countermeasures impose either less flexibility for choosing parameters, an extra cost for decoding or additional requirements for the protocol. In this context, algebraic codes which benefit from deterministic decoding present an important practical advantage.

Still regarding practical applicability, we must compare their storage complexity. p-adic Goppa codes requires mk bits for storing the public-key, which is a factor t better than generic Goppa codes but not enough to overcome our graph-based variant. Using QC-MDPC codes, the public-key size is reduced to only k bits. Nevertheless, when considering the syndrome sizes, the p-adic proposal achieves a better performance. This is notably advantageous for situations where messages are exchanged more often then public-keys.

Regarding the flexibility of the approaches, the p-adic proposal has another advantage. It can be used to efficiently instantiate Parallel-CFS digital signature scheme. MDPC codes have an extremely low decodable syndrome density and therefore would require a prohibitive algorithmic or storage cost.

In summary, both approaches have advantages and drawbacks. Choosing one solution over the other will strongly depend on the target application.

Impact of Our Contributions

Our contributions have already generated related investigations.

p-adic Goppa proposal:

- In [Per11], the author extends our construction to define Generalized-Srivastava codes, another sub-family of alternant codes.
- In [Kob09], the author extends our construction to define Goppa codes of increased maximum length. The author used a different approach to the one presented in Section 4.1.5.
- In [Hey11] an implementation of the p-adic Goppa proposal was presented.

MDPC proposal:

- In [GMRZ13], the authors present a similar technique than our MDPC approach but considering the rank-metric.
- In [HMG13], the authors present an implementation for QC-MDPC parameters for embbeded devices.

All these works demonstrate the interest of the community in our contributions and in the goal of achieving efficient code-based cryptosystems in general.

7.2 Future Works

Our contributions lead to several possible future works. Regarding algebraic codes, the recent attacks against the preliminary quasi-dyadic proposal together with the distinguisher for high-rate Goppa codes definitely motivate a better understanding on the security of using Goppa codes for cryptography.

As for the graph-based approach, there is room for some improvements. For example, the MDPC proposal does not achieve arbitrarily small decoding failure rate, as demonstrated for the rank-metric variant [GMRZ13]. It would be interesting to investigate if MDPC codes can also benefit from similar analysis. This would probably come along with more precise estimations for the error correction capability.

A particularly interesting investigation consists of extending the MDPC approach to generate nearly sparse parity-check matrices of random codes. Random codes have minimum weight close to what is known as the Gilbert-Varshamov (GV) distance. In short, the idea is to construct codes with parity-check equations of weight slightly above the GV distance of a code of same size. In this sense, this construction would produce nothing more than a convenient description of random codes.These codes should be able to correct a quite small number of errors using belief propagation techniques, but still good enough to provide an advantage in comparison to adversaries. In Appendix A, we present a discussion about a preliminary investigation on this direction. It is important to emphasize that, so far, we have not been able to prove that this construction can attain an uniform distribution, much less admit practical parameters. However, these preliminary thoughts may encourage further investigation towards achieving public-key cryptography based on random codes.

Bibliography

[BBC08] M. Baldi, M. Bodrato, and F. Chiaraluce. A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *Proceedings of the 6th international conference on Security and Cryptography for Networks (SCN 2008)*, pages 246–262, Berlin, Heidelberg, 2008. Springer-Verlag. 1.2, 1.3.3, 5, 5.3.2, 5.3.3, 6.4.3

[BBC12] M. Baldi, M. Bianchi, and F. Chiaraluce. Security and complexity of the McEliece cryptosystem based on QC-LDPC codes. Preprint, 2012. http://arxiv.org/abs/1109.5827. 5

[BC07] M. Baldi and F. Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *IEEE International Symposium on Information Theory (ISIT 2007)*, pages 2591–2595, June 2007. 1.2, 5, 5.3.2, 5.3.3

[BCG06] M. Baldi, F. Chiaraluce, and R. Garello. On the usage of quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In *Proceedings of the First International Conference on Communication and Electronics (ICEE'06)*, pages 305–310, October 2006. 1.2, 5, 5.3.2, 5.3.3

[BCGM07] M. Baldi, F. Chiaraluce, R. Garello, and F. Mininni. Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In *IEEE International Conference on Communications (ICC 2007)*, pages 951–956, june 2007. 1.2, 5, 5.3.2, 5.3.3

[BCGO09] T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing key length of the McEliece cryptosystem. In Bart Preneel, editor, *Progress in Cryptology (AFRICACRYPT 2009)*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97. Springer, 2009. 1.2, 1.3.3, 3.2, 3.2.2, 3.2.2, 4, 4.1.2, 4.4.2

[BCMN10] P. S. L. M. Barreto, P.-L. Cayrel, R. Misoczki, and R. Niebuhr. Quasi-dyadic CFS signatures. In *The 6th China International Conference on Information Security and Cryptology (Inscrypt 2010)*, volume 6584 of *Lecture Notes in Computer Science*, pages 336–349. Springer, 2010. 1.3.3, 3, 4.1.5

[BJMM12] A. Becker, A. Joux, A. May, and A. Meurer. Decoding random binary linear codes in $2^{n/20}$: How 1+1=0 improves information set decoding. In *Advances in Cryptology (EUROCRYPT 2012)*,

volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2012. 1.1, 2.3.3, 4.5, 6.6

[BLM11] P. S. L. M. Barreto, R. Lindner, and R. Misoczki. Monoidic codes in cryptography. In *Post-Quantum Cryptography (PQCrypto 2011)*, volume 7071 of *Lecture Notes in Computer Science*, pages 179–199. Springer, 2011. 1.3.3, 4, 4.2

[BLP08] D. J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *Proceedings of the 2nd International Workshop on Post-Quantum Cryptography (PQCrypto 2008)*, Lecture Notes in Computer Science, pages 31–46, Berlin, Heidelberg, 2008. Springer-Verlag. 1.1, 4.5.1

[BLP11] D. Bernstein, T. Lange, and C. Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *Advances in Cryptology (CRYPTO 2011)*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer, 2011. 10.1007/978-3-642-22792-942. 4.5

[BML13] P. S.L.M. Barreto, R. Misoczki, and R. Lindner. Decoding square-free goppa codes over $\mathbb{F}_p$. *IEEE Transactions on Information Theory*, 59(10):6851–6858, 2013. 4.2, 4.3.1, 4.5.3

[BMvT78] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on*, 24(3):384 – 386, may 1978. 5

[CFS01] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology (Asiacrypt 2001)*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174, Gold Coast, Australia, 2001. Springer. 1.1, 2.3.1

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1.1

[FGO+11] J.-C. Faugère, V. Gauthier, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high rate McEliece cryptosystems. In *IEEE Information Theory Workshop (ITW 2011)*, pages 282–286, Paraty, Brazil, October 2011. 1.1, 5, 7

[Fin10] M. Finiasz. Parallel-CFS. In *Selected Areas in Cryptography (SAC 2010)*, volume 6544 of *Lecture Notes in Computer Science*, pages 161–172. Springer, 2010. 2.3.1

[FL08] P.-A. Fouque and G. Leurent. Cryptanalysis of a hash function based on quasi-cyclic codes. In *RSA Conference, Cryptographers' Track (CT-RSA 2008)*, volume 4964 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2008. 6.5.2

[FM08] C. Faure and L. Minder. Cryptanalysis of the mceliece cryptosystem over hyperelliptic curves. In *International Workshop on Algebraic and Combinatorial Coding Theory*, pages 99 – 107, 2008. 1.1

[FOPT10a] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *Advances in Cryptology (EUROCRYPT 2010)*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2010. 3.2.2, 4, 4.1.4, 4.4.3, 5

[FOPT10b] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of mceliece variants with compact keys – towards a complexity analysis. In *2nd International Conference on Symbolic Computation and Cryptography (SCC 2010)*, pages 45–55. RHUL, 2010. 4.4.3

[FS09] M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In *Advances in Cryptology (Asiacrypt 2009)*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2009. 1.1, 2.3.1, 2.3.3, 2.3.3, 2.3.3

[Gab05] P. Gaborit. Shorter keys for code based cryptography. In *International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, 2005. ACM Press. 1.2, 3.2, 3.2.1, 3.2.1, 3.2.1, 4, 4.4.2, 5.3.2

[Gal63] R. G. Gallager. *Low-Density Parity-Check Codes*. PhD thesis, M.I.T., 1963. 2.1.3, 5, 5.1, 5.2, 5.2, 5.2.1, 1, 5.2.1, 6.3, 6.3.1

[Gib95] J. K. Gibson. Severely denting the Gabidulin version of the McEliece public key cryptosystem. *Designs, Codes and Cryptography*, 6(1):37–45, 1995. 1.1

[Gib96] J. K. Gibson. The security of the Gabidulin public key cryptosystem. In *Advances in Cryptology (EUROCRYPT 1996)*, volume 1070 of *Lecture Notes in Computer Science*, pages 212–223, Zaragoza, Spain, 1996. Springer. 1.1

[GMRZ13] P. Gaborit, G. Murat, O. Ruatta, and G. Zémor. Low rank parity check codes and their application to cryptography. In *International Workshop on Coding and Cryptography (WCC 2013)*, pages 168–180, Bergen, Norway, 2013. 6.4.5, 7.1, 7.2

[GPT91] E.M. Gabidulin, A.V. Paramonov, and O.V. Tretjakov. Ideals over a non-commutative ring and their application in cryptology. In DonaldW. Davies, editor, *Advances in Cryptology (EUROCRYPT'91)*, volume 547 of *Lecture Notes in Computer Science*, pages 482–489. Springer Berlin Heidelberg, 1991. 1.1, 6.4.5

[Gul73] M. N. Gulamhusein. Simple matrix-theory proof of the discrete dyadic convolution theorem. *Electronics Letters*, 9(10):238–239, 1973. 8, 4.3.1

[Hey11] S. Heyse. Implementation of McEliece based on quasi-dyadic goppa codes for embedded devices. In Bo-Yin Yang, editor, *Post-Quantum Cryptography (PQCrypto 2011)*, volume 7071 of *Lecture Notes in Computer Science*, pages 143–162. Springer Berlin Heidelberg, 2011. 4.3.2, 7.1

[HMG13] S. Heyse, I. Maurich, and T. Güneysu. Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices. In *Cryptographic Hardware and Embedded Systems (CHES 2013)*, volume 8086 of *Lecture Notes in Computer Science*, pages 273–292. Springer, 2013. 6.4.4, 7.1

[HOP96] J. Hagenauer, E. Offer, and L. Papke. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 42(2):429 – 445, March 1996. 6.3.3

[HP03] W. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003. 6.3.1, 6.3.2

[JM96] H. Janwa and O. Moreno. Mceliece public key cryptosystems using algebraic-geometric codes. *Designs, Codes and Cryptography*, 8(3):293–307, August 1996. 1.1

[KI01] K. Kobara and H. Imai. Semantically secure mceliece public-key cryptosystems – conversions for McEliece PKC. In *Public Key Cryptography (PKC 2001)*, volume 1992 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2001. 10.1007/3-540-44586-2-2. 2.3.1, 6.3.3

[Kob09] K. Kobara. Flexible quasi-dyadic code-based public-key encryption and signature. Cryptology ePrint Archive, Report 2009/635, 2009. 4.1.5, 7.1

[LB88] P.J. Lee and E.F. Brickell. An observation on the security of mceliece's public-key cryptosystem. In *Advances in Cryptology (EUROCRYPT'88)*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988. 2.3.3

[LDW94] Y. X. Li, R. H. D., and X. M. W. On the equivalence of McEliece's and niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273, January 1994. 2.3.1, 2.3.2

[Loi13] P. Loidreau, 2013. Personal Communication. 6.5.2

[MB09a] R. Misoczki and P. S. L. M. Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography*, pages 376–392, 2009. 1.3.3, 2

[MB09b] R. Misoczki and P. S. L. M. Barreto. Compact McEliece keys from Goppa codes. Submitted preprint, 2009. `http://eprint.iacr.org/2009/187`. 3.2.2, 1, 4.1.4, 4.4.3

[McE78] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, January 1978. 1, 1.1, 2.3.1

[Mer79] Ralph Merkle. *Secrecy, authentication and public key systems – A certified digital signature*. PhD thesis, Stanford University, 1979. 1

[Mil86] V. S. Miller. Use of elliptic curves in cryptography. In *Advances in cryptology (CRYPTO 85)*, pages 417–426, New York, USA, 1986. Springer-Verlag. 1, 1.1

[MMT11] A. May, A. Meurer, and E. Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In *Advances in Cryptology (ASIACRYPT 2011)*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011. 1.1, 2.3.3

[MN96] D. J.C. MacKay and R. M. Neal. Near shannon limit performance of Low Density Parity Check codes. *Electronics Letters*, 32:1645–1646, 1996. 2.1.3

[MRS00] C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *IEEE International Symposium on Information Theory (ISIT'2000)*, page 215, Sorrento, Italy, 2000. IEEE. 1.2, 5, 5.3.1, 5.3.2, 5.3.3

[MS78] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. Elsevier Science B. V., 1978. 2.1, 3, 4.1.2

[MS07] L. Minder and A. Shokrollahi. Cryptanalysis of the sidelnikov cryptosystem. In *Advances in Cryptology (EUROCRYPT 2007)*, volume 4515 of *Lecture Notes in Computer Science*, pages 347–360. Springer, 2007. 1.1

[MTSB12] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. L.S.M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. Submitted preprint, 2012. `http://eprint.iacr.org/2012/409`. 1

[MTSB13] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. L.S.M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *IEEE International Symposium on Information Theory – ISIT'2013*, pages 2069–2073, Istambul, Turkey, 2013. IEEE. 1.3.3, 2

[Nie86] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986. 1.1, 2.3.1, 2.3.2

[OB09] S. Ouzan and Y. Be'ery. Moderate-density parity-check codes. Preprint, 2009. `http://arxiv.org/abs/0911.3262`. 6

[OTD10] A. Otmani, J.P. Tillich, and L. Dallot. Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Special Issues of Mathematics in Computer Science*, 3(2):129–140, January 2010. 3.2.1, 4.4.2, 5.3.2

[Per11] E. Persichetti. Compact McEliece keys based on quasi-dyadic Srivastava codes. Cryptology ePrint Archive, Report 2011/179, 2011. `http://eprint.iacr.org/`. 4.2, 7.1

[Pet10] C. Peters. Information-set decoding for linear codes over $\mathbb{F}_q$. In *Third International Workshop on Post-Quantum Cryptography (PQCrypto 2010)*, volume 6061 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2010. 2.3.3, 4.2

[Pet11] C. Peters. *Curves, Codes, and Cryptography*. PhD thesis, Technische Universiteit Eindhoven, the Netherlands, 2011. http://alexandria.tue.nl/extra2/711052.pdf. 2.3.3, 4.5

[Pra62] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, September 1962. 1.1, 2.3.3, 2.3.3, 2.3.3, A.3.2, A.4

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 1, 2.2

[Rya03] W. E. Ryan. An introduction to LDPC codes. In *CRC Handbook for Coding and Signal Processing for Recording Systems*. CRC Press, 2003. 5.1

[Sar77] D. V. Sarwate. On the complexity of decoding Goppa codes. *IEEE Transactions on Information Theory*, 23(4):515–516, 1977. 4.3.1

[Sch59] S. Schechter. On the inversion of certain matrices. *Mathematical Tables and Other Aids to Computation*, 13(66):73–77, 1959. http://www.jstor.org/stable/2001955. 10

[Sen05] N. Sendrier. Encoding information into constant weight words. In *Proceedings of the International Symposium o Information Theory (ISIT 2005)*, pages 435–438, 2005. 2.3.1

[Sen09] N. Sendrier. On the use of structured codes in code based cryptography. In *Coding Theory and Cryptography III*, Contactforum, pages 59–68. Koninklijke Vlaamse Academie van België voor Wetenschaeppen en Kunsten, 2009. 2.3.2, 2.1

[Sen11] N. Sendrier. Decoding one out of many. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-25405-5-4. 2.3.3, 6.5.2, 6.6

[Sho97] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. 1

[Sid94] V.M. Sidelnikov. A public-key cryptosystem based on binary Reed-Muller codes. *Discrete Mathematics and Applications*, 4(3):191–207, 1994. 1.1

[SS92] V.M. Sidelnikov and S. Shestakov. On cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics*, 4(3):57–63, 1992. 1.1

[SS96] M. Sipser and D. A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42:1710–1722, 1996. A.3.1, A.3.1

[Ste89] J. Stern. A method for finding codewords of small weight. In G. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1989. 1.1, 2.3.3

[Tan06] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 2006. 5.1

[TZ75] K. K. Tzeng and K. Zimmermann. On extending Goppa codes to cyclic codes. *IEEE Transactions on Information Theory*, 21:721–716, 1975. 4.1

[UL09] V. G. Umana and G. Leander. Practical key recovery attacks on two McEliece variants. Cryptology ePrint Archive, Report 2009/509, 2009. `http://eprint.iacr.org/`. 4.1.4

[Wag92] D. Wagner. A generalized birthday problem. In *Advances in Cryptology (CRYPTO'97)*, pages 288–303. Springer-Verlag, 1992. 2.3.3

[Wie06] C. Wieschebrink. Two NP-complete problems in coding theory with an application in code based cryptography. In *IEEE International Symposium on Information Theory (ISIT 2006)*, pages 1733–1737, Seattle, USA, 2006. IEEE. 4.1.2, 4.4.2

[Yan90] Y. X. Yang. Dyadic matrices and their potential significance in cryptography. In *Advances in Cryptology (Auscrypt'90)*, volume 453 of *Lecture Notes in Computer Science*, pages 308–310. Springer, 1990. 4.1.1

[YMT97] A. M. Youssef, S. Mister, and S. E. Tavares. On the design of linear transformations for substitution permutation encryption networks. In *Selected Areas in Cryptography (SAC'97)*, pages 40–48, 1997. 4.1.1

Appendices

Appendix A

Random Codes

> **Note.** *We discuss an attempt to extend the MDPC approach to construct random codes. So far, we have not been able to develop techniques that prove the validity of this construction.This chapter intends to encourage further studies on achieving public-key cryptography based on random codes.*

MDPC codes present an interesting advantage for code-based cryptography in comparison to algebraic proposals, namely the absence of algebraic structure. This feature led to the security reduction proof presented in Chapter 6. This security reduction is not tight and relies on a (reasonable) assumption: the only way to distinguish MDPC codes consists of finding dual low-weight codewords. Although better than the scenario for algebraic codes, the security of this McEliece variant still relies on the distinguishing assumption.

In this chapter, we analyze the possibility of extending the MDPC construction to define *random* codes. Random codes have minimum weight close to the Gilbert-Varshamov distance d_{GV}. Therefore, our approach is to construct codes from parity-check matrices with equations of weight slightly above d_{GV} of a code of same size. Using belief propagation decoding, these codes should be able to correct an increasing, albeit small, number of errors. These codes will suffer from extremely long code-lengths and low code-rate. Nevertheless, they might provide a sufficiently good advantage for who has the sparse description, permitting one to build a secure cryptosystem upon them. We believe that a McEliece-like scheme satisfying these conditions would have parameters as in Table A.1.

Parameter	Magnitude
Code-length n:	$n \to +\infty$
Code-rate R:	$R \to 0$
Code dimension k:	$\tilde{\mathcal{O}}(\sqrt{n})$
Code co-dimension r:	$\mathcal{O}(n)$
Parity-check equations weight w:	$\tilde{\mathcal{O}}(d_{GV}(n,k))$
Number of errors t:	$\tilde{\mathcal{O}}(\sqrt{n})$

Table A.1: Magnitude of Parameters.

A.1 Preliminaries

Definition A.1 (Gilbert-Varshamov distance)**.** *We denote by d_{GV} the smallest integer such that*

$$\binom{n}{d_{GV}} > 2^{n-k}$$

Definition A.2 (Number of codewords of a given weight)**.** *We denote by $\mathcal{A}_w(\mathcal{C})$ the number of distinct codewords in a code $\mathcal{C}$ of weight exactly w, i.e.*

$$\mathcal{A}_w(\mathcal{C}) = |\{v \in \mathcal{C} \mid \mathsf{wt}(v) = w\}|$$

The expectation of the random variable $\mathcal{A}_w$ is:

$$\mathbb{E}[\mathcal{A}_w] = \frac{\binom{n}{w}}{2^r}.$$

A.2 Construction

We begin our discussion by introducing the simple Algorithm 5 that would generate the private-key of our proposal.

Algorithm 5 Private-Key Generation.

PARAMETER: n, k, w.
OUTPUT: $k \times n$ generator-matrix G.

1: $G \leftarrow [0]^{k \times n}$
2: **for** $i \in [1 \dots k]$ **do**
3: $\quad G_i \xleftarrow{\$} \mathcal{S}_{n,w}$
4: **end for**
5: **if** G is not full-rank **then**
6: $\quad$ **go to** Step 1.
7: **end if**
8: **return** G

In practice, the idea is to use the output of Algorithm 5 as a parity-check matrix. The algorithm is described in terms of a generator-matrix because in the majority part of this chapter, we will be analyzing the properties of the code-family generated by such outputs.

Algorithm 5 can be used to define different family of codes. The difference refers of the order of parameter w in comparison with the code-length n. LDPC codes require $w = \mathcal{O}(1)$, for example.

In this chapter we want to prove the following informal statement:

Algorithm 5 generates almost all (n, k)-linear codes almost uniformly, when n is large enough and $w = d_{GV} + \alpha \log_2(n)$, for some $\alpha > 0$.

A.3 Analysis

Let $\log(x)$ denotes the logarithmic function in base 2. From now on, we denote

$$\mathsf{w} = d_{GV} + \alpha \log(n).$$

Initially, we want to prove that the expected number of codewords of weight w is polynomial in n, for well chosen values of α. This is necessary to attest that a random linear code would contain at least k codewords of weight w, thus it is a valid output for Algorithm 5.

Proposition A.1. *For some positive integer a and for any $\alpha \geq \frac{a}{\log\left(\frac{n-d_{GV}}{d_{GV}}\right)}$ that defines $\mathsf{w} = d_{GV} + \alpha \log_2(n)$, it holds that*

$$\mathbb{E}[\mathcal{A}_{\mathsf{w}}] \geq n^a.$$

Proof. We want to find α such that:

$$\mathbb{E}[\mathcal{A}_{\mathsf{w}}] = \frac{\binom{n}{\mathsf{w}}}{2^{n-k}} \geq n^a \tag{A.1}$$

Since $\binom{n}{d_{GV}} > 2^{n-k}$, it is enough to prove that:

$$\frac{\binom{n}{\mathsf{w}}}{\binom{n}{d_{GV}}} \geq n^a \tag{A.2}$$

The simplified Stirling's approximation for binomial coefficients says that:

$$\binom{n}{w} \approx 2^{nh(w/n)}, \tag{A.3}$$

where h is the entropy function $h(x) = -x\log(x) - (1-x)\log(1-x)$. Thus

$$\begin{aligned} \mathbb{E}(\mathcal{A}_{\mathsf{w}}) &= \frac{\binom{n}{\mathsf{w}}}{\binom{n}{d_{GV}}} \\ &\approx \frac{2^{nh(\mathsf{w}/n)}}{2^{nh(d_{GV}/n)}} \\ &= 2^{n(h(\mathsf{w}/n) - h(d_{GV}/n))} \geq n^a. \end{aligned} \tag{A.4}$$

Taking the logarithm in both sides, it holds that:

$$n\left(h\left(\frac{\mathsf{w}}{n}\right) - h\left(\frac{d_{GV}}{n}\right)\right) \geq a\log(n). \tag{A.5}$$

The entropy is concave function with maximum value $h(1/2) = 1$. Since both w/n and d_{GV}/n are smaller than $1/2$ and $\mathsf{w} > d_{GV}$, it holds that

$$\frac{h(\mathsf{w}/n) - h(d_{GV}/n)}{\mathsf{w}/n - d_{GV}/n} \leq h'(d_{GV}/n)$$

The first derivative of the entropy function is $h'(x) = \log\left(\frac{1-x}{x}\right)$. Thus:

$$\begin{aligned} \frac{h(\mathsf{w}/n) - h(d_{GV}/n)}{(\mathsf{w}/n - d_{GV}/n)} &\leq \log\left(\frac{n - d_{GV}}{d_{GV}}\right). \\ h(\mathsf{w}/n) - h(d_{GV}/n) &\leq (\mathsf{w}/n - d_{GV}/n)\log\left(\frac{n - d_{GV}}{d_{GV}}\right). \end{aligned} \tag{A.6}$$

Then we can use the upper bound for $h(\mathsf{w}/n) - h(d_{GV}/n)$ from A.6 into A.5, to obtain the α that ensures $\mathbb{E}[\mathcal{A}_{\mathsf{w}}] \geq n^a$:

$$\begin{aligned} & n(\mathsf{w}/n - d_{GV}/n)\log\left(\frac{n - d_{GV}}{d_{GV}}\right) \geq a\log(n) \\ & (\mathsf{w} - d_{GV})\log\left(\frac{n - d_{GV}}{d_{GV}}\right) \geq a\log(n) \\ & \alpha\log(n)\log\left(\frac{n - d_{GV}}{d_{GV}}\right) \geq a\log(n) \\ & \alpha\log\left(\frac{n - d_{GV}}{d_{GV}}\right) \geq a \\ & \alpha \geq \frac{a}{\log\left(\frac{n-d_{GV}}{d_{GV}}\right)} \end{aligned} \tag{A.7}$$

□

With Proposition A.1, we could verify that w can be chosen such that nearly all linear codes can be produced by Algorithm 5.

The other requirement to attest that Algorithm 5 indeed produces random linear codes refers to the distribution of its outputs. Although reasonable, prove that the output of Algorithm 5 follows an almost uniform distribution seems to be a hard problem. This is due to the fact that some few codes may contain an abnormal large number of codewords of weight w, thus leading to a predominant probability to be generated by our algorithm. Unfortunately, we were not able to prove that this condition does not affect our construction significantly.

A.3.1 Error-Correction

The codes generated by Algorithm 5 should correct an increasing, albeit small, number of errors, when the code-length increases. Apparently, there would be different approaches to prove this intuition, but so far we have not been able to develop a satisfactory one. Next we describe two different attempts.

Using Gallager's Analysis

An idea is to use the weak estimation for error correction capability of LDPC codes as presented by Gallager and explained in Chapter 5. It is based on the probability P_i that a bit is in error after i iterations of decoding algorithm. When such a probability converges to zero, reliable error correction could be achieved. Next we describe some remarks on this analysis.

We recall the definition of P_i from Chapter 5, for a code of length n, dimension k and columns weight w_c.

$$P_{i+1} = P_0 \cdot \sum_{l=0}^{b-1} \binom{w_c - 1}{l} (1-r_i)^l r_i^{w_c-1-l} + (1-P_0) \cdot \sum_{l=b}^{w_c-1} \binom{w_c - 1}{l} r_i^l (1-r_i)^{w_c-1-l},$$

where $r_i = \frac{1-(1-2P_i)^{k-1}}{2}$.

In order to $P_i \to 0$ for increasing i, we need to satisfy $P_{i+1} < P_i$. If we consider the simplest case when $b = \lceil w_c/2 \rceil$, then the probability $P_0 \cdot \sum_{l=0}^{b-1} \binom{w_c-1}{l} (1-r_i)^l r_i^{w_c-1-l}$ becomes negligible and therefore

$$\begin{aligned} P_1 &\approx (1-P_0) \sum_{l=b}^{w_c-1} \binom{j-1}{l} r_0^l (1-r_0)^{w_c-1-l} \\ &< P_0 \end{aligned}$$

Thus it would suffice to show that

$$\sum_{l=b}^{w_c-1} \binom{w_c - 1}{l} r_0^l (1-r_0)^{w_c-1-l} < \frac{P_0}{1-P_0},$$

for k and w_c as defined in Table A.1.

However, this analysis is prejudiced by the fact that we cannot assume the message bits are independent from each other. We believe that by adopting very long code-lengths this undesired effect might become negligible, a remark to be verified.

Using Expander Codes

The other approach is based on a technique developed for expander codes [SS96], which are codes based on graphs that can be randomly generated. This is similar to our construction, considering sparse matrices as the adjacency matrices of graphs. In this work, the authors present decoding algorithms and prove their effectiveness to correct a constant number of errors at each decoding iteration. Expander codes use expander graphs, which have the expander property.

Definition A.3 (Expansion Property)**.** *Let $G = (V, E)$ be a graph on n vertices. Every set of at most m vertices* expand *by a factor of δ if, for all sets $S \subset V$:*

$$|S| \leq m \Rightarrow |\{y : \exists x \in S \text{ such that } (x, y) \in E\}| > \delta |S|$$

More precisely, the graphs here discussed are *unbalanced bipartite* graphs, *i.e.* the vertices of the graph are divided into two sets in such a way that no edges between vertices in the same set will exist. Note that this type of graph is also used to represent LDPC and MDPC codes.

Definition A.4 ((c, d)-regular graph)**.** *A graph is (c, d)-regular if is bipartite and all nodes in one set have degree c and all nodes in other have degree d.*

Definition A.5 ((c, d, ϵ, δ)-expander graph)**.** *A graph is (c, d, ϵ, δ)-expander if it is a (c, d)-regular graph in which every subset of at most an ϵ fraction on the c-regular vertices expands by a factor of at least δ.*

Remark 12. The values c and d can be seen respectively as the column and the row weight of the adjacency matrix of the graph.

In the [SS96], the values c and d remain constant, when n increases. In our analysis, these values should increase along with n, since we define d slightly above the Gilbert-Varshamov distance, which increases with n. Thus, using a similar approach to what is obtained for expander codes, we tried to prove the following informal statements:

1. Algorithm 5 generates adjacency matrices of good expander graphs.
2. The decoding algorithm for expander codes can be used to correct an increasing, albeit small, number of errors when applied to our codes.

However, the fact that c and d are not constant seems to strongly prejudice this analysis, and so far, we have not been able to circumvent this problem.

A.3.2 Security Assessment

To obtain a secure cryptosystem we have to prove that our scheme would provide a super-polynomial advantage for legal users (who are allowed to perform efficient decoding) in comparison to adversaries (who can only perform general decoding).

The best algorithms to perform general decoding are derived from the information set decoding technique [Pra62]. These recent improvements have introduced a collision search as a part of the algorithm. It is easy to see that, as long as the parameters increases, the gain provided by this collision step vanishes by the cost of the Gaussian elimination. In this sense, the original information set decoding algorithm becomes more efficient than its recent variants.

Consider a code of length n and dimension k. Moreover, suppose we want to estimate the cost of the original information set decoding for decoding t errors in this code. As stated in Chapter 2, this cost is proportional to the inverse of the probability of finding an error vector with the following profile: t errors into $n-k$ positions and 0 errors into the other k positions. Thus:

$$\begin{aligned} WF_{Pra62}(n,k,t) &\approx \frac{\binom{n}{t}}{\binom{n-k}{t}} \\ &\approx \frac{n^t}{(n-k)^t} \\ &\approx \left(1-\frac{k}{n}\right)^{-t} \end{aligned} \tag{A.8}$$

Suppose that we want to correct $t = \sqrt{n}\,(\ln n)^{\gamma}$ errors and that the dimension of the code is of the form $k = \sqrt{n}\,(\ln n)^{\beta}$. Then (A.8) becomes:

$$WF_{Pra62}(n,k,t) \approx \left(1-\frac{(\ln n)^{\gamma}}{\sqrt{n}}\right)^{-\sqrt{n}(\ln n)^{\beta}}, \tag{A.9}$$

which is approximately $\exp((\ln n)^{\gamma+\beta})$ and that can be rewritten as:

$$WF_{Pra62}(n,k,t) \approx n^{(\ln n)^{\gamma+\beta-1}} \tag{A.10}$$

Therefore, a super-polynomial advantage is ensured for $k = \sqrt{n}\,(\ln n)^{\beta}$ and $t = \sqrt{n}\,(\ln n)^{\gamma}$ when $\gamma + \beta > 1$.

A.4 Conclusion

In this chapter, we discussed a few remarks on extending the MDPC approach to construct random codes for instantiating code-based cryptosystems. We believe that this construction is possible for long code-lengths and small code rates.

These codes should be able to correct a small number of errors. We have seen that if the number of errors is of the form $t = \sqrt{n}\,(\ln n)^{\gamma}$ and the dimension is $k = \sqrt{n}\,(\ln n)^{\beta}$, then it is possible to ensure a super-polynomial advantage for the legal users when compared to adversaries using information set decoding techniques. Note that, for such parameters, the original information set decoding algorithm [Pra62] is not worse than its most recent variants.

Another barrier to such a construction is the proof that Algorithm 5 generates codes uniformly distributed. This is due to the fact that very few codes might have an abnormal large number of codewords of weight w. In this context, some codes would have an abnormal high probability to be generated over other codes, reducing the set of private-keys in practice.

List of Figures

List of Algorithms

List of Tables