UNIVERSITÀ POLITECNICA DELLE MARCHE
SCUOLA DI DOTTORATO DI RICERCA IN SCIENZE DELL'INGEGNERIA
CURRICULUM IN INGEGNERIA ELETTRONICA, ELETTROTECNICA E DELLE
TELECOMUNICAZIONI

On the use of structured codes for cryptographic applications

Ph.D. Dissertation of:
Paolo Santini

Advisor:
Prof. Marco Baldi

Coadvisor:
Ing. Giuseppe Gottardi

Curriculum Supervisor:
Prof. Francesco Piazza

XVIII edition - new series

UNIVERSITÀ POLITECNICA DELLE MARCHE
SCUOLA DI DOTTORATO DI RICERCA IN SCIENZE DELL'INGEGNERIA
CURRICULUM IN INGEGNERIA ELETTRONICA, ELETTROTECNICA E DELLE
TELECOMUNICAZIONI

On the use of structured codes for cryptographic applications

Ph.D. Dissertation of:
Paolo Santini

Advisor:
Prof. Marco Baldi

Coadvisor:
Ing. Giuseppe Gottardi

Curriculum Supervisor:
Prof. Francesco Piazza

XVIII edition - new series

UNIVERSITÀ POLITECNICA DELLE MARCHE
SCUOLA DI DOTTORATO DI RICERCA IN SCIENZE DELL'INGEGNERIA
FACOLTÀ DI INGEGNERIA
Via Brecce Bianche – 60131 Ancona (AN), Italy

"How many years must some people exist
before they're allowed to be free?
And how many times can a man turn his head
and pretend that he just doesn't see?
The answer, my friend, is blowin' in the wind,
the answer is blowin' in the wind"

Blowing in the wind, Bob Dylan

Abstract

The upcoming advent of quantum computers poses a serious threat on most of modern public key cryptosystems, which are commonly based on hard mathematical problems such as the integer factorization or the discrete logarithm. It has indeed been proven that quantum algorithms can be used to solve such problems in polynomial time. Thus, we strongly need to move to a class of new cryptographic primitives, constructed upon mathematical problems for which no efficient quantum solver exists. For this reason, the corresponding schemes are normally defined *post-quantum*.

Code-based cryptosystems are among the most promising ones. Security of such schemes is based on hard problems arising from the coding theory, such as that of decoding a random linear code, which is a well known NP-complete problem, for which no efficient quantum solver is known. Despite a largely recognized security, the main drawback of code-based cryptosystems is represented by the large public key sizes. Indeed, in such schemes the public key is the representation of an error correcting code, whose length must be sufficiently large to guarantee correction of a non trivial amount of errors. The most meaningful example is that of the original McEliece cryptosystem, named after its inventor Robert McEiece that, in 1978, de facto initiated the area of code-based cryptography. The original McEliece proposal, based on Goppa codes, is still essentially unbroken, and additionally yields to a very low algorithmic complexity. However, the scheme, due to its large public key size, has experienced very few practical applications.

In this work we investigate solutions to address the public key size problem. A common strategy is that of replace the original choice of Goppa codes with a more convenient family of codes (for instance, codes correcting more errors or admitting a compact representation). A well assessed strategy to face this issue is that of relying on codes having a large and known automorphism group: in such cases, indeed, the whole code can be completely represented by a bunch of its codewords. Thus, when codes of this type are used to derive the public key, compactness in the code representation may be achieved, with the obvious result of reducing the public key size.

However, this choice may represent a security flaw, since it the end it consists in adding some constraints to the employed code which, to enable correct correction, necessarily needs to already have a significant amount of structure. In a few words, this choice somehow reduces the security of the scheme, in a way that depends not only on the chosen family of codes, but also on the strength of the imposed geometric symmetry. While this fact certainly represents a major issue for some families

of algebraic codes, it is not the case for pseudo-random codes, i.e., codes that admit a random-like fashion design. The most significant case is that of Low-Density Parity-Check (LDPC) codes, codes whose only constraint is that of being represented through a sparse parity-check matrix (i.e., with a low number of set entries). This property guarantees the existence of efficient probabilistic decoding techniques, that can correct non trivial amount of errors with an algorithmic complexity that grows linearly with the code length. When such codes are employed, a regular geometrical structure can be safely introduced to obtain very compact keys and, at the same, high algorithmic efficiency, without no significant security reduction. However, with respect to algebraic codes, LDPC codes are characterized by a completely new venue of attacks, which come from the sparse inner structure and from the intrinsic probabilistic nature of the decoder.

In this work we investigate the use of such codes in modern public key cryptosystems. We describe the main properties of LDPC decoding techniques, and provide methodologies to assess their error correction performances. We describe cryptographic primitives based on LDPC codes and analyze both classical and modern cryptanalysis techniques.

Foreword

During my period at Dipartimento di Ingegneria dell'Informazione of Università Politecnica delle Marche as a Ph.D. student, I had the pleasure to work in the research groups leaded by the professors Marco Baldi and Franco Chiaraluce. They have introduced me to the area of coding theory and, in particular, to that of code-based cryptography. During these years I had the possibility to collaborate with other important universities such as the University of Zurich, Switzerland, and the Florida Atlantic University, USA. In particular, I spent almost three months at the Math Department of FAU, where part of this thesis was developed through the collaboration with Prof. Edoardo Persichetti. The contents of this thesis have been partially included in the following publications.

- M. Baldi, F. Chiaraluce, J. Rosenthal, P. Santini and D. Schipani, "Security of generalised Reed–Solomon code-based cryptosystems," in IET Information Security, vol. 13, no. 4, pp. 404-410, 7 2019.

- M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, P. Santini, "A Finite Regime Analysis of Information Set Decoding Algorithms", in MDPI Algorithms 2019, Volume 12, Issue 10.

- M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, "Ledakem: A post-quantum key encapsulation mechanism based on QC-LDPC codes", in Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings, pages 3–24, 2018.

- M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, "LEDAcrypt: QC-LDPC Code-Based Cryptosystems with Bounded Decryption Failure Rate", in Baldi M., Persichetti E., Santini P. (eds) Code-Based Cryptography. CBC 2019, Lecture Notes in Computer Science, vol11666.

- P. Santini, M. Battaglioni, F. Chiaraluce, M. Baldi, "Analysis of reaction and timing attacks against cryptosystems based on sparse parity-check codes", in Baldi M., Persichetti E., Santini P. (eds) Code-Based Cryptography. CBC 2019, Lecture Notes in Computer Science, vol11666.

- P. Santini, M. Battaglioni, M. Baldi and F. Chiaraluce, "Hard-Decision Iterative Decoding of LDPC Codes with Bounded Error Rate," ICC 2019 - 2019 IEEE

International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-6.

- P. Santini, M. Baldi, and F. Chiaraluce: “Assessing and countering reaction attacks against post-quantum public-key cryptosystems based on QC-LDPC codes”. In: Cryptology and Network Security - 17th International Conference, CANS 2018, Naples, Italy, September 30 - October 3, 2018, Proceedings. pp. 323–343 (2018).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The first use of term *cryptograph* dates back to the 19-th century, due to Edgar Allan Poe, in the novel The Gold Bug. The term refers to a message whose real meaning is hidden, and cannot be recovered without the knowledge of some secret, which is necessary to disclose the real cryptograph information. Mankind has, since ancient times, resorted to this kind of techniques: for instance, one of the oldest techniques to obtain a cryptograph is the "Caesar cypher", named after its inventor Julius Caesar. This method, obtained by replacing the letters of the alphabet in a fixed (and secret) order, was used by the Roman emperor to safely transmit orders to the army officers. Another famous historical example is that of the so called "Babington Plot": the plan orchestrated to kill the Queen Elizabeth I was fully discovered with the aim of Thomas Phelippes, a cryptanalyst that decrypted the letters exchanged between the conspirators (one of them being Mary Stuart, Queen of Scots).

Generally, the study of techniques that are used to create a cryptograph is known as *cryptography*, while *cryptanalysis* refers to the methods which can disclose the hidden message without knowing the secret. Nowadays, we use cryptographic techniques everyday: every time we send a message, we connect to a site, we make an online payment, our communications are protected by a cryptographic protocol. Undoubtedly, our lives have significantly improved because of the opportunities provided by means such as Internet, smartphones, IOT technologies, etc: without cryptography, the whole digital world would collapse.

Cryptographic protocols divide into *symmetric* and *asymmetric* schemes. The first ones allow two parties to establish secure communications, provided that they have already shared a common secret, which is used to perform both encryption and decryption. On the contrary, asymmetric schemes, sometimes called *public key cryptosystems*, allow two (or more) parties to safely exchange messages even if no secret has already been shared. These techniques are, for instance, at the base of communications over the Internet (via HTPPS), in which they are employed to provide authentication and to safely instantiate trusted channels. The main difference between these two paradigms is in the fact that asymmetric cryptosystems require a pair of keys: the *public key* can only be used to perform encryption, while the *secret key* is used to

decrypt. The keys are generated as a pair, in the sense that ciphertexts produced with a specific public key can be decrypted only via the paired secret key. The rationale of this mechanism is based on the concept of *trapdoor*: in a proper cryptosystem, recovering a plaintext from a ciphertext requires on average a huge effort, but becomes easy with the knowledge of the secret key.

Formally, each cryptosystem is based on some mathematical problem, which must be generally hard to solve, apart from some particular instances. First, the secret key is chosen to provide an easy instance of the problem; then, its structure is hidden into that of the public key, which must not reveal any useful information and, thus, must be indistinguishable from a random sequence. An adversary trying to attack the scheme can be modeled as an algorithm trying to solve an arbitrary instance of the underlying mathematical problem: the system is considered secure if the expected number of operations performed by such an algorithm is below some sufficiently large security threshold. With the use of the secret key, the hard instance represented by the ciphertext can be turned into an easy instance, and can thus be efficiently solved: finding such a solution corresponds to perform decryption.

The majority of the public key cryptosystems that we use nowadays is based on classical hard problems such as that of finding the factorization of large integers (RSA), or on that of solving the discrete logarithm (Diffe-Hellman key exchange). However, recent results show that these problems can be efficiently solved by a *quantum computer*, that is, a machine that exploits the quantum mechanic to increase its computational capacity. Roughly speaking, a quantum computer uses quantum bits (called qubits) to perform parallel computations, by taking simultaneously into account many configurations of the input variables: for certain mathematical problem, this property can be exploited to devise algorithms which run significantly faster than their classical counterparts. This is the case of Shor's algorithm, proposed in 1994 [1], which can be used to efficiently (i.e., in polynomial time) factor integers and to compute discrete logarithms. As a consequence, systems such as RSA and Diffie-Hellman cannot be considered adequate anymore because, to guarantee acceptable security levels, the required public keys would become massive (in the order of tens of gigabytes). For this reason, the National Institute for Standardization and Technology (NIST) has initiated the process to standardize a new class of cryptographic primitives [2,3], provided with post-quantum security, i.e., able to withstand cryptanalysis performed through quantum algorithms. The NIST initiative, which has started in 2016, consists in a "competition", in which teams of researchers from all over the world have submitted proposals to undergo evaluation. NIST criteria for deciding the "winners" are based on both security and efficiency analysis; the competition is divided into three rounds and, at the end of the third round, the "survived" algorithms will be standardized. Post-quantum cryptosystems are based on a whole new and wide class of mathematical problems, which includes, for instance, multivariate, lattices, coding theory and

isogenies problems: for all of these problems, an efficient quantum solver is unlikely to exist.

Code-based cryptosystems, initiated by McEliece in 1978 [4], are among the oldest public key schemes. These schemes are mostly based on the problem of decoding a random linear code, proven NP-hard in 1978 [5], for which no efficient quantum algorithm is currently known [6]. The McEliece cryptosystem, named after its inventor, is based on error correcting codes, i.e., codes equipped with an efficient decoding algorithm that can correct a non trivial amount of errors. The secret key is obtained by randomly picking a code with such properties, while the public key is derived by obfuscating its structure into that of a code which is made indistinguishable from a random one. Encryption is performed by corrupting a codeword with a fixed amount of intentional errors: an adversary trying to decipher is faced with an instance of the general decoding problem, while the legitimate user can rely on the hidden structure of the secret code to efficiently decode (and, thus, recover the original codeword).

The original McEliece proposal was based on Goppa codes, a class of binary error correcting errors: after more than 40 years of cryptanalysis, this scheme is still essentially unbroken. Despite its largely recognized security, however, the McEliece cryptosystems has encountered scarce applications, because of its public key size which is significantly larger than that of some competing solutions. To address this issue, researchers have tried to design cryptosystems based on codes with a compact representation, with the goal of obtaining reductions in the key size. For instance, we can mention quasi-dyadic codes [7] and Quasi-Cyclic (QC) algebraic codes [8]. This kind of solutions, however, is normally deemed as not promising, since the addition of a geometrical structure to an algebraic code seriously threatens the scheme security [9]. Roughly speaking, satisfying both algebraic and geometrical constraints results in a secret code that, in many cases, can efficiently be deducted from the public one.

This idea of quasi-clicity then been extended to codes of a different nature, namely the Low-Density Parity-Check (LDPC) codes [10–13] and their recent variant known as Moderate-Density Parity-Check (MDPC) codes [14]. The family of LDPC codes, introduced by Gallager in 1963 [15], is particularly interesting because of the possibility of having a random-like design which, at the same time, guarantees the existence of efficient decoding techniques. In such a case, the geometrical structure does not arise security issues: the resulting schemes are thus provably secure and, benefiting from the compact representation, have small public key sizes. However, the price to pay is represented by the fact that LDPC codes, differently from algebraic codes, do not admit efficient bounded distance decoders. Thus, decoding of LDPC is normally characterized by some failure probability. When employed in cryptosystems, the resulting decryption procedure is then characterized by some failure probability as well, which may be exploited by an opponent to mount so-called *reaction attacks* [16–19]. To avoid these attacks, the failure probability needs to be provably low; however, the

required values (which, for practical applications, are smaller than 2^{-128}) are far away from those that we can reach with numerical simulations, such that only theoretical arguments can be provided to guarantee this feature. For all these reasons, the use of LDPC in cryptography represents an interesting, and surely challenging, matter of study.

In this manuscript we investigate the use of structured LDPC codes in code-based cryptosystems. We first analyze LDPC codes from the decoding perspective, and introduce methodologies and tools which guarantee the design of efficient decoding techniques with provably low failure probabilities. We then introduce the general framework of *reproducible codes*, that is, codes admitting a compact representation. We then embed LDPC into this framework to describe LEDAcrypt [20], one of the algorithms that is currently under evaluation in the NIST competition. We provide insights on the main cryptanalysis techniques that can be used against this family of codes and describe the design criteria of LEDA instances.

1.1 Main contributions of the thesis

In the following list the main contributions of this thesis are reported.

Chapter 3

- Proposal of a McEliece variant based on Generalized Reed Solomon (GRS) codes.

Chapter 4

- Study of the total error correction capability of LDPC codes, under parallel Bit Flipping (BF) decoding.
- Definition of an analytical upper bound of the DFR of LDPC codes, for one iteration of BF decoding.

Chapter 5

- Introduction of the concept of reproducible codes as a generalization of codes with compact representation.
- Definition of general properties of reproducible codes.

Chapter 6

- Introduction of new design criteria for QC-LDPC codes based cryptosystems.

- Introduction of a new efficient decoding algorithm for QC-LDPC codes.
- Generalization of known statistical attacks against QC-LDPC codes based cryptosystems.

Chapter 2

Preliminaries

In this section we recall some basic notions of coding theory and cryptography, which will be useful to provide a general background of the topics we treat in this manuscript.

2.1 Notation

For q being a prime power, we will use $\mathbb{F}_q$ to denote the finite field with q elements. The sum in the binary field (i.e, $q = 2$) will somehow denoted with $\oplus$, to avoid confusion between integers and elements over a finite field.
Vector and matrices will be denoted with bold lower and upper case letters, respectively. Given a vector $\mathbf{a}$, we use a_i to denote its entry in the i-th position. For a matrix $\mathbf{A}$, we will use $a_{i,j}$ to denote its entry in the i-th row and j-th column. The length-n vector formed by all zeros will be denoted as $\mathbf{0}_n$; in analogous way, $\mathbf{0}_{k\times n}$ denotes the null $k \times n$ matrix. The identity matrix of size k is denoted as $\mathbf{I}_k$.
For a set A, we denote its cardinality, i.e., the number of its elements, as $|A|$; $a \xleftarrow{\$}$ will denote the fact that a is randomly picked among the elements of A. Given a set J and a vector $\mathbf{a}$, we denote with $\mathbf{a}_J$ the vector formed by the entries of $\mathbf{a}$ which are indexed by J. For a matrix $\mathbf{A}$, we use $\mathbf{A}_J$ to denote the matrix formed by the columns of $\mathbf{A}$ that are indexed by J. The cartesian product of two sets A and B, i.e., the set of all couples (a, b), with $a \in A$ and $b \in B$, is denoted as $A \times B$.
For a function $f(x)$, we will write $g(x) = O\big(f(x)\big)$ if asymptotically $f(x)$ tends to $g(x)$; the definition can be straightforwardly be adapted to the case in which $f(x)$ is a multivariate function.

2.2 Public key cryptography

A public key cryptographic scheme is a protocol in which two parties can establish a secure communication, even if they do not posses any shared secret. Public key cryptosystems make use of a pair of keys, which we call *public key* and *private key* and denote respectively as pk and sk; the first one is used to perform encryption, while the

latter one is used to perform decryption. Cryptanalysis of the scheme may be devoted to recover the secret key and/or to decrypt an intercepted ciphertext. The security of the scheme is commonly associated to the concept of *computational security*; this property is captured by the so called *Security Level* or *security parameter* λ, with the following meaning: a scheme reaches λ-bits security if any attack requires more than 2^λ operations. The number of operations required to attack a cryptosystem is also sometimes referred to as *work factor*.

Formally, a public-key encryption scheme is described as a triple of algorithms:

i) a Key Generation algorithm which, on input the security parameter λ, returns the key pair $\{sk, pk\}$;

ii) an Encryption algorithm which takes as input a message m and the public key pk and returns the ciphertext $\mathbf{x}$; some randomness may additionally be provided as input;

iii) a Decryption algorithm that, on input a ciphertext $\mathbf{x}$ and the secret key sk, returns either a message m or a failure $\perp$.

Many cryptographic primitives rely on *hash functions*, which can mathematically be defined as follows.

Definition 1 *An* hash function $\mathcal{H}$ *is a function that takes as input a sequence (with no constraints on its length) and returns a sequence of fixed length. We consider an hash function secure, up to the security parameter* λ*, if finding a collision, i.e., finding two inputs* m *and* m' *such that* $\mathcal{H}(m) = \mathcal{H}(m')$*, requires a work factor larger than* 2^λ*. Normally, the hash image of a message is called* digest.

2.3 Coding theory

In this section we provide a brief introduction to some basic concepts of coding theory.

Definition 2 *A linear code* $\mathcal{C}$ *of length* n *and dimension* k *over* $\mathbb{F}_q$ *is a linear subspace of* $\mathbb{F}_q^n$ *of dimension* k*. The parameters* n*,* k *and* $r = n - k$ *are called, respectively, length, dimension and redundancy of the code.*

In particular, the description of a linear code can be provided in terms of two matrices, which are defined in the following.

Definition 3 *Given a linear code* $\mathcal{C}$ *with length* n *and dimension* k*, we say that* $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ *is a* generator matrix *for* $\mathcal{C}$ *if and only if*

$$\mathcal{C} = \left\{ \mathbf{uG} |\ \mathbf{u} \in \mathbb{F}_q^k \right\}.$$

If $\mathbf{G} = [\mathbf{I}_k | \mathbf{V}]$*, with* $\mathbf{V} \in \mathbb{F}_q^{k \times n-k}$*, we say that* $\mathbf{G}$ *is the* systematic generator matrix *for* $\mathcal{C}$*.*

Definition 4 *Given a linear code* $\mathcal{C}$ *with length* n *and dimension* k*, we say that* $\mathbf{H}$ *is a* parity-check matrix *for* $\mathcal{C}$ *if and only if*

$$\mathcal{C} = \left\{ \mathbf{c} \in \mathbb{F}_q^n \ \ s.t. \ \ \mathbf{H}\mathbf{c}^\top = \mathbf{0}_{n-k \times 1} \right\}.$$

If $\mathbf{H} = [\mathbf{V} | \mathbf{I}_r]$*, with* $\mathbf{V} \in \mathbb{F}_q^{r \times n-r}$*, we say that* $\mathbf{H}$ *is the* systematic parity-check matrix *for* $\mathcal{C}$*.*

Remark 1 *If* $\mathbf{G}$ *and* $\mathbf{H}$ *are, respectively, generator and parity-check matrices of a code* $\mathcal{C}$*, then* $\mathbf{H}\mathbf{G}^\top = \mathbf{0}_{n-k \times k}$*.*

Remark 2 *If* $\mathbf{G}$ *(resp.* $\mathbf{H}$*) is a generator (resp. parity-check) matrix for* $\mathcal{C}$*, then each matrix in the form* $\mathbf{G}' = \mathbf{S}\mathbf{G}$*, with* $\mathbf{S} \in \mathbb{F}_q^{k \times k}$ *of full rank (resp.,* $\mathbf{H}' = \mathbf{S}\mathbf{H}$*, with* $\mathbf{S} \in \mathbb{F}_q^{r \times r}$ *of full rank) is a valid generator (resp. parity-check) matrix for* $\mathcal{C}$*.*

Codes are normally with respect to their error correction capability; in particular, this concept is strongly related to that of the minimum distance of the code, which is defined as follows.

Definition 5 *Given a vector* $\mathbf{c} \in \mathbb{F}_q^n$*, we define its Hamming weight, and denote it with* $\mathrm{wt}(\mathbf{c})$*, as the number of its non null entries.*

Definition 6 *Given a vector* $\mathbf{c} \in \mathbb{F}_q^n$*, we define its support, and denote it as* $\mathrm{S}(\mathbf{c})$*, as the integers set defined as*

$$\mathrm{S}(\mathbf{c}) = \{ i \ \ s.t. \ \ c_i \neq 0 \}.$$

Clearly, $|\mathrm{S}(\mathbf{c})| = \mathrm{wt}(\mathbf{c})$*.*

Definition 7 *For two vectors* $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$*, we define the* Hamming distance *as*

$$\mathrm{d}(\mathbf{a}, \mathbf{b}) = \mathrm{wt}(\mathbf{a} - \mathbf{b}) = \mathrm{wt}(\mathbf{b} - \mathbf{a})$$

Definition 8 *For a code* $\mathcal{C}$*, we define its* minimum distance *as*

$$d = \min_{\substack{\mathbf{c}, \mathbf{c}' \in \mathcal{C} \\ \mathbf{c} \neq \mathbf{c}'}} \{ \mathrm{d}(\mathbf{c}, \mathbf{c}') \}.$$

If the code is linear, we additionally have

$$d = \min_{\substack{\mathbf{c}, \mathbf{c}' \in \mathcal{C} \\ \mathbf{c} \neq \mathbf{c}'}} \{ \mathrm{d}(\mathbf{c}, \mathbf{c}') \} = \min_{\mathbf{c} \in \mathcal{C}} \{ \mathrm{wt}(\mathbf{c}) \}.$$

Remark 3 *In the above definitions, we have only considered the Hamming distance, which gives rise to the Hamming weight. However, the presented concepts are general and can be adapted to any distance function (such as the Rank distance, the Lee distance, etc.).*

The concept of minimum distance is strongly connected to that of error correction. First of all, we remember the Gilbert-Varshamov bound, which is frequently used as a tight estimate of the minimum distance of a random code.

Definition 9 *Given a linear code $\mathcal{C}$ over $\mathbb{F}_q$, with length n and dimension k, the* Gilbert-Varshamov distance *is defined as*

$$d^{(\mathtt{GV})} = \max\left\{ d \in \mathbb{N} \;\; s.t. \;\; \sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i \leq q^{n-k} \right\}.$$

For random linear codes, the GV-bound is normally used as an approximation of the minimum distance; the bound becomes tighter as the code length grows.

Codes are commonly used to correct errors that corrupt a transmitted codeword; a typical study case is that of additive errors, i.e., such that the received word is $\mathbf{x} = \mathbf{c} + \mathbf{e}$, with $\mathbf{c}$ being a codeword and $\mathbf{e}$ being the error vector. In this situation, the parity-check matrix can be used to detect error-affected codewords.

Definition 10 *Given $\mathbf{x} \in \mathbb{F}_q^n$ and a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{r \times n}$, we call* syndrome *the length-r vector $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$.*

By definition, a codeword has null syndrome, while an error corrupted word may have non null syndrome. We indeed have

$$\mathbf{s} = \mathbf{H}\mathbf{x}^\top = \mathbf{H}\mathbf{c}^\top + \mathbf{H}\mathbf{e}^\top.$$

Then, $\mathbf{s}$ corresponds to the syndrome of the error vector; if $\mathbf{e} \notin \mathcal{C}$, then $\mathbf{s} \neq \mathbf{0}_r$. In such a case, we say that the error has been detected. In particular, because of the minimum distance property, it is clear that all error vectors with weight $t \leq d-1$ can be detected.

However, error detection is commonly not employed, since codes are rather used to correct errors. In such a case, decoding is performed through a *decoding algorithm* $\mathcal{D}$ that, on input $\mathbf{x}$, returns either a codeword or a decoding failure. From the properties of the minimum distance, it can be seen that, given $\mathbf{x} = \mathbf{c} + \mathbf{e}$, $\mathbf{e}$ can be unequivocally determined only if its Hamming weight is lower than $d/2$. In all the other cases, the decoding problem, i.e., finding $\mathbf{e}$ and $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x} = \mathbf{c} + \mathbf{e}$, may have more than one solution.

The problem of decoding is central in coding theory, and becomes crucial in cryptography. Indeed, almost all code-based cryptosystems are built on the decoding trap-

door, that is, on the difficulty of decoding a code with no apparent structure. In principles, the optimal decoder is the one that, on input an n-uple $\mathbf{x} \in \mathbb{F}_q^n$, returns a codeword $\mathbf{c} \in \mathcal{C}$ such that

$$\mathrm{d}(\mathbf{c}', \mathbf{x}) \geq \mathrm{d}(\mathbf{c}, \mathbf{x}), \quad \forall \mathbf{c}' \in \mathcal{C}.$$

The operating principle of this decoder, which we define *Maximum Likelihood* decoder, can be easily described as follows:

1. the decoder computes the distance between the received $\mathbf{x}$ and all codewords of $\mathcal{C}$;
2. if there is only one codeword minimizing the distance from $\mathbf{x}$, then the decoder output corresponds to such word;
3. if there is more than one codeword at the same (minimum) distance from $\mathbf{x}$, then the decoder randomly selects one of such words.

The complexity of this decoding technique is easy to estimate: for a code with length n and dimension k, we have

$$C_{\mathtt{ML}} = O(nq^k).$$

It is clear that this decoding technique is not efficient, since its complexity grows exponentially with the code dimension. Indeed, normally error correction is performed only when specific families of error correcting codes are used, equipped with an efficient decoding technique.

2.4 Hard problems from the coding theory

A central problem in coding theory is the *Maximum Likelihood Decoding Problem*, which is defined as follows.

Problem 1 ***Maximum Likelihood Decoding Problem (MLDP)*** *Given $\mathcal{C} \subseteq \mathbb{F}_q^n$ and $\mathbf{x} \in \mathbb{F}_q^n$, find*

$$\mathbf{c} \in \mathcal{C} \;\; \textit{s.t.} \;\; \mathrm{d}(\mathbf{c}, \mathbf{x}) = \min_{\mathbf{c}' \in \mathcal{C},\ \mathbf{c}' \neq \mathbf{c}} \{\mathrm{d}(\mathbf{c}', \mathbf{x})\}$$

Related to MLDP, we can define many other problems in coding theory; in this manuscript we focus on the *Syndrome Decoding Problem (SDP)* and the *Minimum Distance Problem (MDP)*, which we formalize in the following.

Problem 2 ***Syndrome Decoding Problem (SDP)*** *Given $\mathbf{H} \in \mathbb{F}_q^{r \times n}$, $\mathbf{s} \in \mathbb{F}_q^r$ and $t \in \mathbb{N}$, find $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathrm{wt}(\mathbf{e}) = t$ and $\mathbf{H}\mathbf{e}^\top = \mathbf{s}$.*

Problem 3 ***Minimum Distance Problem (MDP)*** *Given $\mathbf{H} \in \mathbb{F}_q^{r \times n}$ and $w \in \mathbb{N}$, find $\mathbf{c} \in \mathbb{F}_q^n$ such that $\mathrm{wt}(\mathbf{c}) \leq w$ and $\mathbf{H}\mathbf{c}^\top = \mathbf{0}$.*

It can be shown that the three above problems are somehow all related; in particular, MDP is sometimes referred to Homogeneous SDP, since it can be seen as a particular instance of SDP with $\mathbf{s} = \mathbf{0}_r$. In particular, MLDP, SDP and MDP problems are NP-hard [5, 21] for general codes over $\mathbb{F}_q$. The implications of this fact are really important, since it means that we cannot dispose of an efficient general decoding algorithm, able to decode an arbitrary code; then, in order to dispose of codes equipped with an efficient decoding algorithm, we need to rely on specific code constructions.

Chapter 3

McEliece and Niederreiter cryptosystems based on algebraic codes

In this chapter we introduce the first historically proposed code-based cryptosystems, the McEliece [4] and Niederreiter [22] schemes, named after their inventors and which date back, respectively, to 1978 and 1986. These two schemes can be interpreted as two faces of a coin since they are different formulations of the same trapdoor, and are essentially equivalent from a security point of view. In this chapter we describe a general framework upon which the two schemes can be constructed, briefly resume the main attack avenues and describe why they are equivalent, when the same family of codes is used. Then, we briefly describe how algebraic codes can fit into such schemes, and which issues arise from the use of such codes. Finally, we present and analyze variants of the McEliece and Niederreiter cryptosystems based on Generalized Reed Solomon codes.
Throughout the chapter, we will use $\mathcal{L}(n, k, t, \mathcal{D})$ to denote a family of codes with length n and dimension k, equipped with an efficient decoding algorithm $\mathcal{D}$ that can decode up to t errors. The set of $k \times k$ non-singular matrices over $\mathbb{F}_q$ will be denoted as $\mathrm{GL}_k\big(\mathbb{F}_q\big)$, while $\mathcal{P}_n$ will denote the ensemble of permutation matrices of size n.

The McEliece cryptosystem

In the original McEliece cryptosystem, the secret key sk is

$$\{\mathcal{C}, \mathbf{S}, \mathbf{P}\} \xleftarrow{\$} \mathcal{L}(n, k, t, \mathcal{D}) \times \mathrm{GL}_k\big(\mathbb{F}_q\big) \times \mathcal{P}_n.$$

Let $\mathbf{G}$ be a generator matrix for $\mathcal{C}$; then, the public key pk is computed as

$$\mathbf{G}' = \mathbf{S}^{-1}\mathbf{G}\mathbf{P}^{-1}.$$

The ciphertext is in the form

$$\mathbf{c} = \mathbf{m}\mathbf{G}' + \mathbf{e},$$

where $\mathbf{m} \in \mathbb{F}_q^k$, $\mathbf{e} \in \mathbb{F}_q^n$ and $\mathrm{wt}\,(\mathbf{e}) = t$.
To decrypt, one first computes

$$\begin{aligned}\mathbf{c}' &= \mathbf{c}\mathbf{P} \\ &= \mathbf{m}\mathbf{S}^{-1}\mathbf{G} + \mathbf{e}\mathbf{P} \\ &= \mathbf{m}'\mathbf{G} + \mathbf{e}',\end{aligned}$$

where $\mathrm{wt}\,(\mathbf{e}') = t$ because $\mathbf{P}$ is a permutation matrix. Then, since $\mathbf{m}'\mathbf{G} \in \mathcal{C}$, the decoding algorithm $\mathcal{D}$ is used to recover the pair $\{\mathbf{m}', \mathbf{e}'\}$ from which, with simple linear algebra, the pair $\{\mathbf{m}, \mathbf{e}\}$ can be obtained.

The Niederreiter cryptosystem

In the original Niederreiter cryptosystem, the secret key sk is

$$\{\mathcal{C}, \mathbf{S}, \mathbf{P}\} \xleftarrow{\$} \mathcal{L}(n, k, t, \mathcal{D}) \times \mathrm{GL}_r\big(\mathbb{F}_q\big) \times \mathcal{P}_n.$$

Let $\mathbf{H}$ be a parity-check matrix for $\mathcal{C}$; then, the public key pk is computed as

$$\mathbf{H}' = \mathbf{S}^{-1}\mathbf{H}\mathbf{P}^{-1}.$$

The ciphertext is in the form

$$\mathbf{s} = \mathbf{H}'\mathbf{e}^\top,$$

where $\mathbf{e} \in \mathbb{F}_q^n$ and $\mathrm{wt}\,(\mathbf{e}) = t$.
To decrypt, one first computes

$$\begin{aligned}\mathbf{s}' &= \mathbf{S}\mathbf{s} \\ &= \mathbf{H}(\mathbf{e}\mathbf{P})^\top \\ &= \mathbf{H}\mathbf{e}'^\top,\end{aligned}$$

where $\mathrm{wt}\,(\mathbf{e}') = t$ since $\mathbf{P}$ is a permutation matrix. Then, the algorithm $\mathcal{D}$ is run to recover $\mathbf{e}'$; the initial error vectors is obtained as $\mathbf{e} = \mathbf{e}'\mathbf{P}^{-1}$.

Equivalence of the McEliece and Niederreiter cryptosystems

In the McEliece cryptosystem, the public key is obtained by hiding the structure of the secret code $\mathcal{C}$, through a scrambling matrix $\mathbf{S}$ and a permutation matrix $\mathbf{P}$. In other words, the public key $\mathbf{G}'$ is made indistinguishable from a random matrix. In such a case, no efficient decoding algorithm can be applied to recover $\mathbf{e}$ from $\mathbf{c}$ and, thus, the security level of the scheme is based on the hardness of solving an arbitrary MLDP instance.
In the Niederreiter scheme, in analogous way, the public key is obtained by obfuscating the structure of the secret code. Then, security of the scheme is based on the hardness of solving an SDP instance.

When the McEliece and the Niederreiter cryptosystems are instantiated with the same code, they are equivalent from the security point of view. In other words, it can be easily shown that the underlying problems are equivalent, in the sense that one can be transformed into the other in polynomial time. For the sake of simplicity, we show this equivalence just in one case, by showing how an instance of MLDP can be turned into an instance of SDP.

Let $\mathbf{G}$ be a generator matrix of a code $\mathcal{C}$ and suppose that, given $\mathbf{c} = \mathbf{mG} + \mathbf{e}$, we want to recover $\mathbf{m}$ and $\mathbf{e}$ such that $\mathrm{wt}(\mathbf{e}) = t$ and $\mathbf{mG} + \mathbf{e} = \mathbf{c}$. With linear algebra, a parity-check matrix $\mathbf{H}$ for $\mathcal{C}$ can always be efficiently (i.e., in polynomial time) computed from $\mathbf{G}$. We can then compute $\mathbf{s} = \mathbf{Hc}^\top = \mathbf{He}^\top$. Then, the triple $\{\mathbf{H}, \mathbf{s}, t\}$ corresponds to an SDP instance, whose solution is $\mathbf{e}$. If we can find $\mathbf{e}$, then with linear algebra we can easily obtain $\mathbf{m}$ from $\mathbf{c} - \mathbf{e}$.

3.1 Modern solutions based on Goppa codes

In the original proposal, the McEliece cryptosystem was instantiated with binary Goppa codes, which are subfield subcodes of Generalized Reed-Solomon (GRS) codes. To obtain a binary Goppa code with error correction capability equal to t, we first choose a polynomial $g(x) \in \mathbb{F}_{2^m}[x]$ of degree t. Then, we choose a set $L = \{l_0, \cdots, l_{n-1}\}$ of n distinct elements from $\mathbb{F}_{2^m}$ which are non zeroes of $g(x)$; the set L is called *support* of the code. The corresponding Goppa code is defined as the set of vectors $\mathbf{c} \in \mathbb{F}_2^n$ such that

$$\sum_{i=0}^{n-1} \frac{c_i}{x - l_i} \equiv 0 \mod g(x). \tag{3.1}$$

If the polynomial $g(x)$ is irreducible, then L can contain the whole field $\mathbb{F}_{2^m}$ and the code length can be maximum (i.e., $n = 2^m$).

To express Eq. (3.1) in a more convenient way, we can describe the code in terms

of its parity-check matrix, which has the following structure

$$\mathbf{H} = \begin{bmatrix} \frac{1}{g(l_0)} & \frac{1}{g(l_1)} & \cdots & \frac{1}{g(l_{n-1})} \\ \frac{l_0}{g(l_0)} & \frac{l_1}{g(l_1)} & \cdots & \frac{l_{n-1}}{g(l_{n-1})} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{l_0^{t-1}}{g(l_0)} & \frac{l_1^{t-1}}{g(l_1)} & \cdots & \frac{l_{n-1}^{t-1}}{g(l_{n-1})} \end{bmatrix}.$$

By projecting $\mathbf{H}$ to the base field $\mathbb{F}_2$, we obtain the parity-check matrix of the corresponding binary Goppa code with size $n = 2^m$, dimension $k = n - mt$ and minimum distance $\geq 2t$. Decoding of such codes can be performed through Patterson's algorithm, which takes as input $g(x)$ and the support L and runs in time $O(nt)$.

Despite more than 40 years of cryptanalysis, the original McEliece proposal is still essentially unbroken, in the sense that a simple parameters update is enough to achieve modern security levels. Indeed, the choice of Goppa codes seems to be very appropriate, since there is no known way to recover the secret key from the public one (apart from a distinguisher which, however, works only in the case of very high rate codes [23]). Then, the only known attacks against this family of codes consists in general decoding algorithms, which we describe in the following section.

To improve efficiency and to reduce the public-key size, modern solutions based on Goppa codes, such as the Classic McEliece submission to the NIST process [24], do not use a permutation anymore and, instead of using a generic $\mathbf{S}$ to scramble the public code, rely on Gaussian elimination to obtain the public key. In other words, the matrices $\mathbf{G}'$ and $\mathbf{H}'$, which are used for the public keys in McEliece and Niederreiter, respectively, are nowadays commonly obtained as the systematic forms of the secret $\mathbf{G}$ and $\mathbf{H}$, respectively. The first obvious advantage of this choice comes from the reduction of the public key size, since the identity matrix does not need to be published; furthermore, this choice reduces the number of operations which are required to perform a complete run of the algorithm (so, it leads to a lower algorithmic complexity). For instance, in the McEliece case, the public key is in the form $\mathbf{G} = [\mathbf{I}_k, \mathbf{V}]$, such that

$$\mathbf{c} = [\mathbf{m}, \mathbf{mV}] + \mathbf{e}.$$

Thus, once $\mathbf{e}$ has been recovered through Patterson's algorithm, to recover $\mathbf{m}$ it is enough to consider the first k entries of $\mathbf{c} - \mathbf{e}$.

3.2 Information Set Decoding

In this section we describe the main cryptanalysis technique which can be used against the McEliece and Niederreiter cryptosystems. Without loss of generality, we focus on the binary McEliece case: given the public key $\mathbf{G}' \in \mathbb{F}_2^{k \times n}$ and a vector

$\mathbf{c} \in \mathbb{F}_2^n$, the adversary tries to determine a vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight t such that $\mathbf{c} + \mathbf{e}$ is a codeword of the code generated by $\mathbf{G}'$. Since the public key does not reveal any information about the secret key, $\mathbf{G}'$ can be considered as the generator matrix of a random code and no efficient decoding technique can be applied. In this context, the best known algorithmic solutions to the problem are known as Information Set Decoding (ISD).

ISD algorithms were initiated by Prange in 1962 [25], and witnessed many improvements along the years; some of the most well known variants are those due to Lee and Brickell [26], Leon [27], Stern [28], Finiasz and Sendrier [29], May, Meurer and Thomae [30], and Becker, Joux, May and Meurer [31]. We refer the interested reader to [32] for a complete review of all the aforementioned algorithms. ISD algorithms can be described either as i) solvers of the general decoding problem, or ii) finders of low-weight codeword in a given code. Actually, these two problems are strongly related, and each ISD algorithm can indeed be used (with some little tweaks) in both ways. For the sake of simplicity, we will only describe ISD as algorithms to solve the general decoding algorithms.

All ISD algorithms have a common structure in which (as hinted by the name itself) *information sets* are used to solve the decoding problem; to this end, we first need to introduce the concept of information set.

Definition 11 *Given a code $\mathcal{C}$ of length n and dimension k, an information set is a set $J \subseteq [0; n-1]$ of cardinality k such that*

$$\mathrm{d}\left(\mathbf{c}_J, \mathbf{c}'_J\right) > 0, \quad \forall \mathbf{c}, \mathbf{c}' \in \mathcal{C}, \quad \mathbf{c} \neq \mathbf{c}'$$

Following from the definition, it can be easily proven that, if J is an information set for a code $\mathcal{C}$, then any generator matrix $\mathbf{G}$ is such that the columns indexed by J are linearly independent; in other words, given $\mathbf{G}$, then $\mathbf{G}_J$ is a $k \times k$ non-singular matrix. Information sets define a unique mapping between a sequence and the corresponding codeword, and can clearly be used to invert the encoding mapping: given $\mathbf{c} = \mathbf{mG}$, for any information set J, we have

$$\mathbf{m} = \mathbf{c}_J \mathbf{G}_J^{-1}.$$

Now, let us consider a McEliece scheme in which the secret code has minimum distance $d > 2t$ and can, thus, correct up to t errors; let $\mathbf{c} = \mathbf{mG} + \mathbf{e}$ be an intercepted ciphertext such that $\mathrm{wt}\left(\mathbf{e}\right) = t$. Clearly, decoding $\mathbf{c}$ by testing all n-uples with weight t is unfeasible, unless t is extremely low.
Let J be an information set, and let us write

$$\mathbf{c}_J = \mathbf{mG}_J + \mathbf{e}_J,$$

from which

$$\mathbf{m} = (\mathbf{c}_J + \mathbf{e}_J)\,\mathbf{G}_J^{-1}.$$

We note that, if $\mathbf{e}_J = \mathbf{0}_k$, then $\mathbf{m} = \mathbf{c}\mathbf{G}_J^{-1}$.

ISD algorithms are based on the following main principle: guess an information set, establish a set of candidates $\hat{\mathbf{e}}_J$ for $\mathbf{e}_J$ and, for each one of them, compute $(\mathbf{c}+\hat{\mathbf{e}}_J)\mathbf{G}_J^{-1}$ and check the correctness of the derived solution. Then, the average complexity of such an algorithm gets estimated as

$$C_{\mathtt{isd}} = \frac{C_{\mathtt{iter}}}{\mathrm{Pr}_{\mathtt{guess}}},$$

where $C_{\mathtt{iter}}$ is the number of operations that are performed to test each guess and $\mathrm{Pr}_{\mathtt{guess}}$ corresponds to the probability that a guessed set J satisfies some conditions (which vary according to the considered ISD variant).

Let $\mathbf{m}'$ be a candidate solution; to verify its correcteness, it is enough to compute

$$\begin{aligned}\mathbf{e}' &= \mathbf{c} + \mathbf{m}'\mathbf{G} \\ &= (\mathbf{m} + \mathbf{m}')\mathbf{G} + \mathbf{e}. \end{aligned} \tag{3.2}$$

If $\mathbf{m}' = \mathbf{m}$, then $\mathbf{e}' = \mathbf{e}$ and has weight t. In the other cases, we have $(\mathbf{m}' + \mathbf{m})\mathbf{G} \in \mathcal{C} \setminus \{0\}^n$ and, thus, has weight $\geq d > 2t$; thus, $\mathrm{wt}\,(\mathbf{e}') > t$. Then, this simple check allows to verify the correctness of a candidate solution.
In the following, we describe the first two ever proposed ISD algorithm. Modern variants are essentially based on the same approach and furthermore, as we describe next, they are likely to perform bad when considering quantum implementations. For there reasons, the choice of describing only the basic ISD algorithms is actually meaningful.

Prange ISD

Prange's ISD is successful anytime the guesses information set J is such that $\mathrm{wt}\,(\mathbf{e}_J) = 0$. This algorithm simply consists in selecting an information set, computing $\mathbf{m}' = \mathbf{c}_J\mathbf{G}_J^{-1}$ and verifying the weight of $\mathbf{e}'$ (obtained through Eq. (3.2)). Thus, the success probability of an iteration corresponds to the probability that a randomly chosen J i) is actually an information set, and ii) does not overlap with the support of $\mathbf{e}$. Condition i) corresponds to having a non singular $\mathbf{G}_J$; to derive the probability that such an event occurs, we consider $\mathbf{G}_J$ as a random $k \times k$ matrix over $\mathbb{F}_2$, such that the probability that it is invertible can be estimated as

$$\mathrm{Pr}_{\mathtt{inv}} = \frac{1}{\prod_{i=1}^{r} 1 - 2^{-i}},$$

which, for parameters of practical interest, can be approximated as 0.2788. Then, we need to consider the probability that the chosen information set does not overlap with the support of the error vector; this probability can be easily obtained as

$$\Pr\left[\mathrm{wt}\left(\mathbf{e}_J\right)=0\right]=\frac{\binom{n-t}{k}}{\binom{n}{k}}.$$

We then have

$$\Pr_{\texttt{guess}}=\Pr_{\texttt{inv}}\cdot\Pr\left[\mathrm{wt}\left(\mathbf{e}_J\right)=0\right]=\frac{\binom{n-t}{k}}{\binom{n}{k}\prod_{i=1}^{r}1-2^{-i}}.$$

When $t \ll k$, we have

$$pr_{\texttt{guess}}\approx\frac{1}{0.2788}2^{-t\log_2\left(1-\frac{k}{n}\right)}.$$

The complexity of an iteration can be estimated with the cost of inverting $\mathbf{G}_J$: we thus have

$$C_{\texttt{iter}}=O(k^3).$$

Lee & Brickell ISD

This algorithm improves upon Prange's ISD by allowing the chosen information set to have a small intersection with the support of $\mathbf{e}$. This comes with an increase in the operations that are performed in a single iteration but, when such intersection is sufficiently small, then the overall algorithm complexity gets reduced. In particular, the size of such an intersection is measured by a parameter p, such that the success probability becomes

$$\Pr_{\texttt{guess}}=\Pr_{\texttt{inv}}\cdot\Pr\left[\mathrm{wt}\left(\mathbf{e}_J\right)\leq p\right],$$

where, again, $\Pr_{\texttt{inv}}=0.2788$ and

$$\Pr\left[\mathrm{wt}\left(\mathbf{e}_j\right)\leq p\right]=\sum_{l=0}^{p}\frac{\binom{t}{l}\binom{n-t}{k-l}}{\binom{n}{k}}.$$

Then, for each information set J, in each iteration the algorithm tests all solutions obtained as

$$\mathbf{m}'=(\mathbf{c}_J+\mathbf{y})\mathbf{G}_J^{-1},\ \ \mathbf{y}\in\mathbb{F}_2^k,\ \ \mathrm{wt}\left(\mathbf{y}\right)\leq p.$$

Normally, the algorithm is optimized with $p=2$, such that the complexity of each iteration can be still measured as $C_{\texttt{iter}}=O(k^3)$.

Modern variants

Modern variants of ISD all follow the Lee & Brickell principle of relaxing the conditions on $\mathbf{e}_J$, to facilitate the information set guessing, at the cost of a limited increase in the iteration complexity. However, we remark the fact that, for all cited variants, the overall complexity remains an exponential function of the number of errors. In particular, the complexity of modern variants is well approximated by the following simple expression [33]

$$C_{\mathtt{isd}} \approx 2^{ct}, \quad c = \log_2\left(\frac{1}{1-\frac{k}{n}}\right). \tag{3.3}$$

Non binary ISD

When codes over non binary field are considered, the ISD approach may be extended [34]. Essentially, ISD algorithms operate in the same way as that of the binary ones. We point out that, as the finite field size grows while all the other parameters do not change, a light increase in the ISD's complexity appears.

3.2.1 Quantum ISD

To understand how quantum algorithms (not) affect code-based cryptography, we consider the results discussed by Bernstein in [6], in which Grover's quantum algorithm [35] is exploited to speed-up the information set guessing phase. Roughly speaking, Grover's algorithm can be used to efficiently find a root of a function: given $f : X \to \mathbb{F}_2$, with u roots, Grover's algorithm can, with high probability, find one of such roots after $\sqrt{|X|/u}$ function evaluations in random points from X. We recall that, by root, we mean a value $x \in X$ such that $f(x) = 0$. The main result of [6] consists in the fact that, given an instance of the decoding problem represented by $\mathbf{G} \in \mathbb{F}_2^{k\times n}$, $\mathbf{c} \in \mathbb{F}_2^n$ and $t \in \mathbb{N}$, one iteration of Prange's ISD can be described as a function f which, on input a set J, acts as follows

1. gives up (and returns 1) if $\mathbf{G}_J$ is singular, otherwise computes $\mathbf{G}_J^{-1}$;
2. computes $\mathbf{m}' = \mathbf{c}\mathbf{G}_J^{-1}$;
3. gives up (and returns 1) if $\mathrm{wt}\,(\mathbf{m}'\mathbf{G} + \mathbf{c}) > t$;
4. returns 0.

It is then clear that finding a root of f corresponds to decode $\mathbf{c}$, i.e., corresponds to find a set J such that $\mathbf{e}_J = \mathbf{0}_k$. Prange's classical ISD evaluates f on randomly sampled sets J, and stops as soon as a root is found; by considering Grover's algorithm, such a search can be sped up. Indeed, the domain of f corresponds to the number of size-k subsets of $[0; n-1]$, while the number of roots of f corresponds to the number of information sets that do not overlap with the support of $\mathbf{e}$. Considering that there are approximately $0.2788\binom{n-t}{k}$ such sets, and that the domain has size $\binom{n}{k}$, the number of required function evaluations (i.e., the average number of ISD iterations) is

$$\sqrt{\frac{0.2788\binom{n}{k}}{\binom{n-t}{k}}}.$$

It is easily seen that this number corresponds to the square root of $\mathrm{Pr}_{\mathtt{guess}}^{-1}$ which we have defined in the previous section. We point out that the complexity of each iteration is not affected by the application of Grover (thus, still requires $O(k^3)$ qubit elementary operations).

When considering more advanced ISD algorithms, a precise evaluation of the actual complexity is strongly needed. However, as stated in [6], quantum versions of these algorithms are likely to gain, at maximum, a very reduced factor with respect to quantum Prange's ISD. Indeed, according to the analysis in [36], Lee & Brickell and all subsequent variants, with respect to Prange, tend to increase the operations performed in each iteration by a factor that is very close to the square root of the increase in the iteration success probability. Then, in the end, their complexity should be very close to the one of quantum Prange. A common and conservative solution to take into account Grover's speed-up is that of using the square root of the classical approximation provided in Eq. (3.3) as a lower bound to the complexity of a quantum ISD algorithm.

3.3 Cryptosystems based on Generalized Reed-Solomon Codes

In this section we consider cryptosystems based on Generalized Reed-Solomon (GRS) codes [37]. On the one hand, GRS codes seem natural candidates for code-based cryptosystems, since they are maximum distance separable codes: their error correction capability is optimum and, for this reason, they should lead to reductions in the key size, with respect to Goppa codes. On the other hand, GRS codes have more structure than Goppa codes and, therefore, may be less secure. The use of GRS codes in cryptosystems has been firstly proposed by Niederreiter in 1986 in its seminal paper [22]; the scheme has been successfully attacked in 1992 by Sidelnikov and Shestakov [38], exploiting the fact that for a GRS code, differently from Goppa codes, permutating the secret code is not enough to hide its structure. The BBCRS

scheme [39] tries to address this issue, by means of a stronger masking of the secret GRS code; however, the proposed instances have been shown to be vulnerable to a polynomial-time attack [40].

In this section we describe a modification of the BBCRS scheme, proposed in [37]; this solution allows avoiding attacks such as those in [40,41] and, furthermore, reduces the decryption complexity of the original BBCRS scheme. This, however, comes with a small increase in the ciphertext length and public key size.

3.3.1 The BBCRS scheme

In this section we describe the main features of the BBCRS scheme; for the sake of simplicity, we focus on the Niederreiter version.

Key generation

The secret key is constituted by the triple, $\{\mathbf{H}, \mathbf{S}, \mathbf{Q}\}$, where

- $\mathbf{H} \in \mathbb{F}_q^{r \times n}$ is the parity-check matrix of a GRS code $\mathcal{C}$, with length n and dimension $k = n - r$, able to correct $t = \frac{r}{2}$ errors;
- $\mathbf{S} \in \mathbb{F}_q^{r \times r}$ is a non-singular matrix;
- $\mathbf{Q} \in \mathbb{F}_q^{n \times n}$ is a non-singular transformation matrix obtained as $\mathbf{Q} = \mathbf{R} + \mathbf{T}$, where $\mathbf{R} \in \mathbb{F}_q^{n \times n}$ and has rank $z \ll n$, while $\mathbf{T} \in \mathbb{F}_q^{n \times n}$ has average row and column weight $m \ll n$. The low rank matrix $\mathbf{R}$ is obtained as $\mathbf{R} = \mathbf{a}^\top \cdot \mathbf{b}$, where $\mathbf{a} \in \mathbb{F}_q^{z \times n}$ and $\mathbf{b} \in \mathbb{F}_q^{z \times n}$.

The public key $\mathbf{H}'$ is computed as

$$\mathbf{H}' = \mathbf{S}^{-1}\mathbf{H}\mathbf{Q}^\top. \tag{3.4}$$

Encryption

The ciphertext is in the form

$$\mathbf{x} = \mathbf{H}'\mathbf{e}^\top,$$

where $\mathrm{wt}(\mathbf{e}) = t_p = \lfloor \frac{t}{m} \rfloor$.

Decryption

To decrypt $\mathbf{x}$, one first computes

$$\begin{aligned}\mathbf{x}' &= \mathbf{S}\mathbf{x} \\ &= \mathbf{H}\mathbf{Q}^\top\mathbf{e}^\top \\ &= \mathbf{H}\left(\mathbf{e}\mathbf{Q}\right)^\top \\ &= \mathbf{H}\left[\mathbf{e}\left(\mathbf{R}+\mathbf{T}\right)\right]^\top \\ &= \mathbf{H}\mathbf{b}^\top\gamma + \mathbf{H}\mathbf{T}^\top\mathbf{e}^\top,\end{aligned}$$

where $\gamma = \mathbf{a}\mathbf{e}^\top$. It is clear that $\mathbf{x}'$ is, in general, the syndrome of an error vector whose weight is far below the error correction of the GRS code. Indeed, the matrix $\mathbf{Q}$ can be considered as a random matrix over $\mathbb{F}_q$, thus, the average weight of its rows can be estimated as $n\frac{q-1}{q}$. The vector $\mathbf{x}'$ corresponds to the syndrome of the vector $\mathbf{e}\mathbf{Q}$, computed through the parity-check matrix $\mathbf{H}$; in particular, $\mathbf{e}\mathbf{Q}$ corresponds to the linear combination of $t_p \ll n$ rows of $\mathbf{Q}$. Since each one of these rows can be considered as a random vector over $\mathbb{F}_q$, then the average weight of $\mathbf{e}\mathbf{Q}$ is, again, $n\frac{q-1}{q}$: clearly, such an amount of errors cannot be efficiently corrected. To decode, the entries of γ must be guessed: to do this, all z-uples over $\mathbb{F}_q$ must be considered. Each candidate z-uple can then be tested, in order to verify its correctness (see [39] for more details); on average, this requires testing $q^z/2$ guesses.
Once γ has been guessed, decryption proceeds by computing

$$\begin{aligned}\mathbf{x}'' &= \mathbf{x}' - \mathbf{H}\mathbf{b}^\top\gamma \\ &= \mathbf{H}\mathbf{T}^\top\mathbf{e}^\top \\ &= \mathbf{H}\mathbf{e}_\mathbf{T}^\top.\end{aligned}$$

Since $\mathbf{e}_\mathbf{T} = \mathbf{e}\mathbf{T}$ has weight $\leq m \cdot t_p \leq t$, $\mathbf{x}''$ is a correctable syndrome through the secret GRS code. Thus, decoding of the secret code returns $\mathbf{e}_\mathbf{T}$, from which the original $\mathbf{e}$ can easily be recovered as $\mathbf{e} = \mathbf{e}_\mathbf{T}\mathbf{T}^{-1}$.

Parameters choice

To have practical public key sizes, the underlying secret GRS code must have moderate length; at the same time, its length must be large enough to guarantee correction of mt_p errors. To obtain such features, the desired value of m should be small. In the same way, z should be small as well, to avoid a too complex decryption phase (indeed, guessing of γ requires to test, on average, $q^z/2$ candidates). However, keeping both

z and m too small exposes the system to polynomial-time attacks [40], therefore a security / performance trade-off arises. In fact, the attack in [40] can be applied only if m and z are such that the following two conditions are simultaneously verified

$$\begin{cases} 1 \leq m \leq 1 + R - \frac{1}{n} - \sqrt{\frac{8}{n}R + \frac{1}{n^2}} < 2, \\ z = 1, \end{cases} \tag{3.5}$$

where $R = k/n$ denotes, as usual, the code rate. In particular, the attack in [40] is built upon a distinguisher of GRS codes based on computing the dimension of the square of shortenings of the public code. The squares of some of these shortenings have a smaller dimension than that of shortened random codes of the same size, due to the structure of the hidden private code. The core of the attack in [40] is an algorithm to distinguish between rows of $\mathbf{T}$ with Hamming weight 1 and rows of $\mathbf{T}$ with Hamming weight 2. In fact, for $1 < m < 2$ the rows of $\mathbf{T}$ have Hamming weight 1 or 2. Then, the effect of weight-2 columns of $\mathbf{T}$ is reverted to that of weight-1 columns through linear combinations of columns of the public parity-check matrix. Through these steps, the public key of an alternative system with the same private code but with $m = 1$ is recovered by the opponent, who can then mount the attack in [41] against such an alternative system to recover the private key.

To counter these attacks, the system parameters must be chosen such that conditions (3.5) on m and z are not verified. However, this has a detrimental effect on the public key size and complexity. In the next section we describe a variant of the BBCRS scheme [37], which overcomes these issues with some little tweaks to the original BBCSR proposal.

3.4 A variant of the BBCRS scheme

This scheme comes from the necessity of countering attacks against the BBCRS scheme, without having significant increases in both m and z. The main idea behind this variant, which we call BCRSS scheme (the acronym is formed by the initial of the authors of [37]), consists in increasing z and, at the same time, in avoiding increases in the decryption complexity by publishing $\mathbf{a}$. The main differences with the original BBCRS scheme are emphasized in the following.

i) The public key is

$$pk = \{\mathbf{M}, \mathbf{a}\},$$

where $\mathbf{M} \in \mathbb{F}_q^{r \times k}$ is such that $\mathbf{H}' = [\mathbf{M}, \mathbf{I}_r]$ is the systematic parity-check matrix of the secret GRS code.

ii) The ciphertext is constituted by the pair

$$\{\mathbf{x} = \mathbf{H}\mathbf{e}^\top \ , \ \gamma = \mathbf{a}\mathbf{e}^\top\}.$$

iii) During decryption, guessing of γ is clearly no longer needed; this saves a factor of $q^z/2$ in the average decryption time.

3.4.1 Security analysis

Since $\mathbf{a}$ is public, the system may be exposed to the subcode vulnerability described in [39, Sect. 3.1]. In fact, from (3.4) it follows that

$$\begin{aligned}\mathbf{H}' &= \mathbf{S}^{-1}\mathbf{H}\mathbf{R}^\top + \mathbf{S}^{-1}\mathbf{H}\mathbf{T}^\top \\ &= \mathbf{S}^{-1}\mathbf{H}\mathbf{b}^\top\mathbf{a} + \mathbf{S}^{-1}\mathbf{H}\mathbf{T}^\top.\end{aligned}$$

An attacker could consider the following alternative parity-check matrix

$$\mathbf{H}_S = \begin{bmatrix} \mathbf{H}' \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{S}^{-1}\mathbf{H}\mathbf{b}^\top\mathbf{a} + \mathbf{S}^{-1}\mathbf{H}\mathbf{T}^\top \\ \mathbf{a} \end{bmatrix}. \tag{3.6}$$

Compared to $\mathbf{H}'$, $\mathbf{H}_S$ includes an additional set of parity-check equations, defined by $\mathbf{a}$, which imply that any codeword $\mathbf{c}$ belonging to the code defined by $\mathbf{H}_S$ satisfies $\mathbf{a}\mathbf{c}^\top = \mathbf{0}$. This in turn implies that $\mathbf{S}^{-1}\mathbf{H}\mathbf{b}^\top\mathbf{a}\mathbf{c}^\top = \mathbf{0}$. Therefore, the constraint imposed by the set of parity-check equations appearing in (3.6) due to $\mathbf{H}'$ becomes $\mathbf{S}^{-1}\mathbf{H}\mathbf{T}^\top\mathbf{c}^\top = \mathbf{0}$, for any codeword $\mathbf{c}$ belonging to the code defined by $\mathbf{H}_S$. In summary, $\mathbf{H}_S$ as in (3.6) defines a subcode of the public code where any codeword $\mathbf{c}$ satisfies $\mathbf{S}^{-1}\mathbf{H}\mathbf{T}^\top\mathbf{c}^\top = \mathbf{0}$. If $\mathbf{T}$ is a permutation matrix, then the subcode defined by $\mathbf{H}_S$ is permutation-equivalent to a subcode of the secret code. A cryptosystem exposing a subcode of a private GRS code is the Berger-Loidreau (BL) scheme [42]. Based on the above considerations, for the case $m = 1$ (i.e., for $\mathbf{T}$ being a permutation), the security of the new system is equivalent to that of a BL scheme with the same subcode parameters. This also makes the attack presented in [43] applicable, and designing parameters for the BL scheme that are secure against known attacks is related to designing secure parameters for the BCRSS scheme, in the case $m = 1$. However, the dimension of such a subcode is equal to $n - \operatorname{rank}(\mathbf{H}_S)$ and we can choose parameters which avoid known attacks on the subcode (which apply in case $m = 1$): if we look at [43, Eq. 8], for dimension $k \geq n/2$ we can avoid the attack by requiring $k - z - 1 < 2k - n + 1$.
In the end, when $m = 1$, a high rate R and $z \geq n - k$ are required, while for $m > 1$ the value of z can be lowered.

Another point to take into account is that, although this type of subcode attack may be avoided, knowing $\mathbf{a}$ and γ facilitates decoding attacks. In fact, $\begin{bmatrix} \mathbf{x} \\ \gamma \end{bmatrix} = \mathbf{H}_S \cdot \mathbf{e}^\top$ and an attacker could perform syndrome decoding on the code defined by $\mathbf{H}_S$, rather than the public code. Such a code has rate $\frac{k-z}{n} < \frac{k}{n}$, and this facilitates ISD decoding attacks. The attack complexity decreases as long as z increases (when $z \geq k$ the

attack becomes very simple, since an adversary can find a full rank $\mathbf{H}_S$ with size $n \times n$ and just invert it). To avoid these issues, optimal parameters correspond to a large dimension k and a relatively small z.

Finally, structural attacks must be kept into account. Starting with the work of Sidelnikov and Shestakov [38], it has been recognized that McEliece type systems having as an underlying (disguised) structure a GRS code are insecure. The main reason for this is that the Schur square code of a GRS code has in general a very small dimension compared to a random code of the same dimension; furthermore, the dimension of the Schur square is an invariant under monomial transformations (such as permutations). Then, the Schur square can be used to distinguish a disguised GRS codes from random codes. Indeed, the attack in [40] exploits the Schur square to attack the BBCRS scheme, while an attack in [44] explains how to attack the BL system [42] through the Schur square. Furthermore, recently Couvreur et al. [45] came up with general polynomial time attacks against a large class of algebraic geometric codes and their subcodes (and GRS codes are algebraic geometric).

The question therefore is if the Schur square of the proposed code could be dimension deficient. For this, let us consider once more the relevant equations

$$\mathbf{x} = \mathbf{H}'\mathbf{e}^\top \text{ and } \gamma = \mathbf{a} \cdot \mathbf{e}^\top.$$

Combining these equations one can study a related parity-check matrix, that is

$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{0}_{r \times z} & \mathbf{H}' \\ -\gamma & \mathbf{a} \end{bmatrix}.$$

Clearly if one has efficient decoding for this parity-check matrix the system becomes insecure. For this, we wish to comment on two extreme cases.

1. Assume that $m = 1$, i.e., $\mathbf{H}$ simply represents a disguised GRS code. Then, it is fairly clear that the Schur square of the defined code is dimension deficient unless the size of z is chosen sufficiently large. So, it is important that $m > 1$.

2. Consider the other extreme case where $z = 0$. Here Couvreur et al. [40] derived a polynomial time attack in case that $1 < m < 2$. Bolkema et al. [46] studied the case of $m = 2$ and they called this situation "the weight two masking of the GRS system". Using extensive simulations, they conjectured that over large fields the Schur square of the public code has the expected dimension of a random code. More recently, Weger [47] has shown that the probability that the Schur square has maximal dimension approaches 1 as the field size q goes to infinity.

These remarks should make it clear that a distinguisher attack using the Schur square computation becomes out of reach when m and z are chosen sufficiently large.

When $z = 0$ it seems that m should be chosen at least two in order to avoid distinguisher attacks. When $z > 0$, at the best of our knowledge, there is no result establishing the minimal value for m which allows avoiding the Schur square distinguisher attack

3.4.2 Concrete instances and comparison with other schemes

In this section we compare performances of the BCRSS scheme with those of competing schemes, such as the McEliece with Goppa codes and the BL scheme. First of all, in the BCRSS scheme the public key corresponds to $kr + z(n - z)$ values over $\mathbb{F}_q$, i.e is made of $(kr + zn - z^2) \log_2 q$ bits. In the BL scheme using a subcode of dimension $k - z$, the public key size is $(k - z)(r + z) \log_2 q$ bits. It follows that BL always has a smaller key size, although for fixed z and increasing k, the difference in key size between the BCRSS and the BL schemes decreases. When $m = 1$, the BCRSS and the BL schemes require the same vale of z to achieve the same security level. Therefore, in such a case the BL system exhibits some advantage over the BCRSS scheme in terms of public key size. However, when $m > 1$, the BCRSS scheme has smaller values of z than those used in the BL scheme, and this yields significant reductions in the public key size.

We now focus on the encryption rate, i.e., on the maximum quantity of information that can be sent with a single ciphertext. For the Niederreiter versions, these values correspond to

$$R_e = \frac{\log_2 \binom{n}{t}}{(n - k)}$$

for the Goppa code-based system, to

$$R_e = \frac{\log_2 \binom{n}{t_p} + t_p \log_2(q - 1)}{(n - k + z) \log_2 q} \tag{3.7}$$

for the BCRSS system (considering γ of length z as part of the ciphertext), and to

$$R_e = \frac{\log_2 \binom{n}{t} + t \log_2(q - 1)}{(n - k + z) \log_2 q} \tag{3.8}$$

for the BL system.

In the McEliece versions, the encryption rates are $\frac{k}{n}$ for the Goppa code-based system, $\frac{k-z}{n}$ for BL and $\frac{k}{n+z}$ for the new system. From (3.7) and (3.8) we observe that, with the Niederreiter version, when the same set of parameters is chosen and $m = 1$ (hence, $t_p = t$), the encryption rate of the BCRSS scheme equals that of BL with the same security level, while with the McEliece version it is higher.

On the basis of the above performance and security metrics, we can compare the

Table 3.1: System performance comparison for $SL = 2^{180}$: (a) Goppa code-based system, (b) BCRSS with $m = 1$ and $z = n - k$, (c) Berger-Loidreau.

Variant	n	k	$t = t_p$	KS (KiB)	R_e (Niederreiter)	R_e (McEliece)
(a)	4096	3004	91	400.44	0.5724	0.7334
(b)	1282	1146	68	392.89	0.3850	0.8082
(c)	1212	1062	75	342.15	0.3806	0.7525

Table 3.2: System performance comparison for $SL = 2^{260}$: (a) Goppa code-based system, (b) BCRSS with $m = 1$ and $z = n - k$, (c) Berger-Loidreau.

Variant	n	k	$t = t_p$	KS (KiB)	R_e (Niederreiter)	R_e (McEliece)
(a)	8192	6957	95	1048.82	0.6012	0.8492
(b)	1950	1754	98	917.37	0.3798	0.8173
(c)	1788	1560	114	801.13	0.3732	0.7450

BCRSS scheme with the classical binary Goppa code-based system and the BL system based on GRS subcodes. For such purpose, we consider some instances of these systems approximately achieving the same security level and compare their features. For GRS code-based systems, we consider full length GRS codes defined over $\mathbb{F}_q$, with $q = n + 1$ being a prime. Goppa code-based systems instead exploit irreducible binary Goppa codes with length equal to a power of two.

In Tables 3.1 and 3.2, we respectively consider codes with a security level (SL) of at least 2^{180} and 2^{260}, estimated as the work factor (WF) of attacks based on ISD, computed according to [34]. On the basis of an exhaustive search performed over the range of parameters of interest, we report the solutions achieving the smallest public key size (KS) expressed in kibibytes (KiB) for the classical binary Goppa code-based cryptosystem, for the new variant of GRS code-based cryptosystem with $m = 1$, and for the BL cryptosystem, using $z = n-k$. We can notice that the BCRSS scheme with $m = 1$ is able to achieve smaller key sizes than the Goppa-based solution, and it may have higher encryption rate in the McEliece version. The BCRSS scheme has always higher encryption rate than BL, particularly for the McEliece version, but larger key size.

As mentioned, $m = 1$ was considered as in this case the new system is comparable to BL, but an m slightly larger than 1 is certainly preferable, to better protect the secret code. In Tables 3.3, 3.4 and 3.5 a few values of m and z are tested and the instances with smallest key sizes are presented. It is evident how it is possible to achieve very interesting parameters, namely high encryption rates and compact public keys. Considering the instances in the tables, the reduction in public key size with respect to the Goppa code-based solution with the same security level can reach 73%.

Table 3.3: BCRSS performances for $m = 1.2$, $SL = 2^{180}$ and $SL = 2^{260}$.

SL	z	n	k	t	t_p	KS (KiB)	R_e (Niederreiter)	R_e (McEliece)
180	10	796	634	81	67	130.09	0.5870	0.7866
180	30	886	722	82	68	172.24	0.5304	0.7882
180	50	918	740	89	74	210.43	0.4879	0.7644
260	10	1162	928	117	97	284.27	0.5894	0.7918
260	30	1222	976	123	102	435.37	0.5467	0.7796
260	50	1276	1018	129	107	408.04	0.5128	0.7677

Table 3.4: BCRSS performances for $m = 1.3$, $SL = 2^{180}$ and $SL = 2^{260}$.

SL	z	n	k	t	t_p	KS (KiB)	R_e (Niederreiter)	R_e (McEliece)
180	10	760	544	108	83	146.06	0.5399	0.7065
180	30	810	576	117	90	186.60	0.4989	0.6857
180	50	852	594	129	99	229.80	0.4671	0.6585
260	10	1122	810	156	120	326.36	0.5399	0.7155
260	30	1200	888	156	120	389.81	0.5104	0.7220
260	50	1236	898	169	130	454.98	0.4843	0.6983

To conclude the comparison, in Table 3.6 we report two instances of the original GRS code-based cryptosystem in [39], achieving smallest key sizes for a security level of 2^{180} and 2^{260}, respectively, and satisfying $m = 1.1 \cdot (1 + R) > 1 + R$ to avoid the attack in [40]. Through the comparison with the previous tables, we observe that, for the same security levels, the BCRSS scheme is able to achieve a reduction in public key size by more than 50%.

Table 3.5: BCRSS performance for $m = 1.8$, $SL = 2^{180}$ and $SL = 2^{260}$.

SL	z	n	k	t	t_p	KS (KiB)	R_e (Niederreiter)	R_e (McEliece)
180	10	1092	804	144	80	298.65	0.4042	0.7296
180	30	1116	792	162	90	357.44	0.3789	0.6911
180	50	1180	852	164	91	418.54	0.3594	0.6927
260	10	1600	1168	216	120	676.31	0.4012	0.7255
260	30	1626	1154	236	131	771.67	0.3828	0.7054
260	50	1722	1268	227	126	865.19	0.3691	0.7156

Table 3.6: Original GRS code-based system performance for $SL = 2^{180}$ and $SL = 2^{260}$.

SL	n	k	m	t	t_p	KS (KiB)	R_e (Niederreiter)	R_e (McEliece)
180	946	504	1.686	221	131	268.87	0.4209	0.5328
260	1422	786	1.708	318	186	639.19	0.4111	0.5527

Chapter 4

Bounds on the error correction of LDPC codes

In this chapter we focus on the family of Low-Density Parit-Check (LDPC) codes, which have been introduced by Gallager in 1963 [15]. We will only consider LDPC codes from the reliability point of view, while their cryptographic applications will be discussed in Chapter 6.

Roughly speaking, we say that a code $\mathcal{C}$ is an LDPC if its parity-check matrix $\mathbf{H}$ contains a low number of set entries; in other words, a code is an LDPC when its parity-check matrix can defined sparse. A formal definition of this class of codes is provided in the following definition.

Definition 12 *We say that a code $\mathcal{C}$ is an LDPC code if it can be described by a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{r \times n}$ such that*

i) the maximum row Hamming weight is $\ll n$;

ii) the maximum column Hamming weight is $\ll r$.

A common choice is that of distributing the weights of rows and columns of $\mathbf{H}$ in a "regular" way; in such a case, we say that the corresponding codes are *regular*.

Definition 13 *We say that a code $\mathcal{C}$ described by a parity-check matrix $\mathbf{H}$ is a (w, v)-regular LDPC code if all the rows and columns of $\mathbf{H}$ have weights respectively equal to $w \ll n$ and $v \ll r$.*

LDPC codes that are commonly used in crypto are indeed regular, and in thesis we will mainly focus on them.

A representation of an LDPC code can be provided in terms of its Tanner graph, that is, an undirected simple bipartite graph $\mathcal{G}$ formed by the set of variable nodes $V = \{v_0, \cdots, v_{n-1}\}$ and that of check-nodes $C = \{c_0, \cdots, c_{r-1}\}$. The set of edges E is defined as follows

$$E = \{(v_i, c_j) \in V \times C \text{ s.t. } h_{j,i} = 1\}.$$

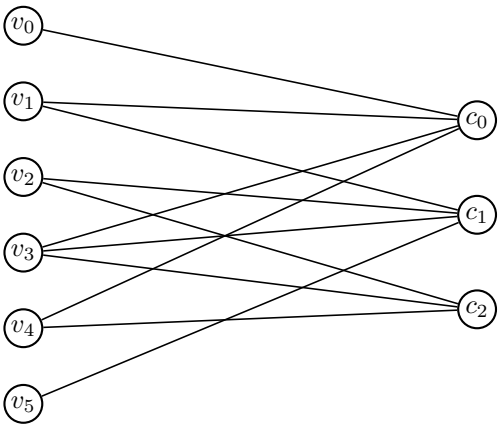


Figure 4.1: Tanner graph example

To provide an example, the graph in Figure 4.1 is the Tanner graph of the following parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

If $\mathbf{H}$ is randomly picked among all $r \times n$ matrices over $\mathbb{F}_2$, then its Tanner graph will contain on average $\frac{rn}{2}$ edges. For the parity-check matrix of an LDPC code, this value is significantly smaller, i.e., $|E| \ll \frac{rn}{2}$. Let $N(v_i)$ denote the *neighbourhood* of v_i, that is, all nodes in $\mathcal{G}$ that are connected to v_i; with analogous meaning, we define $N(c_j)$. Due to the definition of LDPC, we have

$$\begin{cases} |N(v_i)| \ll r & \forall v_i \in V, \\ |N(c_j)| \ll n & \forall c_j \in C. \end{cases}$$

If the code is (w, v)-regular, we furthermore have

$$\begin{cases} |N(v_i)| = v \ll r & \forall v_i \in V, \\ |N(c_j)| = w \ll n & \forall c_j \in C. \end{cases}$$

4.1 Decoding LDPC codes

The sparse nature of parity-check matrices of LDPC codes allows for efficient decoding techniques. In this manuscript we focus on the Bit-Flipping (BF) decoder, originally proposed by Gallager [15]; this simple decoding technique, because of its very low complexity (which grows linearly with the code length) has a crucial importance in code-based cryptography, as we explain in Section 6.

A common BF decoding procedure is depicted in Algorithm 1.

Algorithm 1 BFdecoder

Input: parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, syndrome $\mathbf{s} \in \mathbb{F}_2^r$,
maximum number of iterations $\mathtt{i}_{\mathtt{max}} \in \mathbb{N}$, threshold $b \in \mathbb{N}$
Output: error vector estimate $\tilde{\mathbf{e}} \in \mathbb{F}_2^n$ or failure $\perp$

$\tilde{\mathbf{e}} \leftarrow \mathbf{0}_n$ ▷ Error vector estimate
$i \leftarrow 0$ ▷ Number of performed iterations
while $\mathrm{wt}(\mathbf{s}) > 0 \wedge i < \mathtt{i}_{\mathtt{max}}$ **do**
  $F \leftarrow \varnothing$ ▷ Set of error-affected estimated bits
  for $j \leftarrow 0$ **to** $n-1$ **do**
    $\sigma_j \leftarrow 0$ ▷ Counter initialization
    for $l \in \mathrm{S}(\mathbf{h}_j)$ **do**
      $\sigma_j \leftarrow \sigma_j + s_l$
    end for
    if $\sigma_j \geq b$ **then**
      $F \leftarrow F \cup j$ ▷ Position j is estimated as error affected
    end if
  end for
  for $j \in F$ **do**
    $\tilde{e}_j \leftarrow \neg \tilde{e}_j$ ▷ Error vector estimate update
    $\mathbf{s} \leftarrow \mathbf{s} \oplus \mathbf{h}_j$ ▷ Syndrome update
  end for
  $i \leftarrow i + 1$
end while
if $\mathrm{wt}(\mathbf{s}) = 0$ **then**
  return $\tilde{\mathbf{e}}$
else
  return $\perp$
end if

The BF decoding procedure is probabilistic and aims at estimating the error vector associated to a syndrome $\mathbf{s}$ through a partial guessing of its set positions. Each time a bit is estimated as error affected, the syndrome is updated by summing the corresponding column of the parity-check matrix. The procedure runs through a maximum number of iterations, and stops as soon as a null syndrome is obtained or the maximum number of iterations is reached. If, at the end of the decoding procedure, the syndrome is not null, then we have encountered a decoding failure. Indeed, let $\mathbf{e}$ be the error vector corresponding to the input syndrome; let $\tilde{\mathbf{e}}$ denote the error vector estimate as some point in the decoding process, and denote with $\tilde{\mathbf{s}}$ the corresponding syndrome. It can be easily seen that

$$\tilde{\mathbf{s}} = \mathbf{H}(\mathbf{e} \oplus \tilde{\mathbf{e}})^\top.$$

If, at some point, we have $\mathbf{s} = \mathbf{0}_r$, this means that $\tilde{\mathbf{e}} \oplus \mathbf{e} \in \mathcal{C}$: with overwhelming

probability we actually have $\tilde{\mathbf{e}} = \mathbf{e}$, which means that the decoder has found the error vector (and then, decoding can stop).

To understand the BF principle, we provide a simple reasoning; for the sake of simplicity, we focus on the case of a (w, v)-regular code. Let $\mathbf{e}$ be an error vector of weight $t \ll n$, with syndrome

$$\mathbf{s} = \bigoplus_{i \in \mathrm{S}(\mathbf{e})} \mathbf{h}_i,$$

where $\mathbf{h}_i$ is the i-th column of $\mathbf{H}$; clearly, $\mathrm{wt}(\mathbf{s}) \leq \min\{r, vt\}$. When $v \ll r$ and $t \ll n$, we expect the weight of $\mathbf{s}$ to be slightly smaller than vt. In other words, this means that few cancellations happen in the syndrome computation: indeed, the syndrome $\mathbf{s}$ is obtained as the sum of a small number of columns of $\mathbf{H}$ which, due to sparsity of $\mathbf{H}$, overlap in a very limited number of positions. Then, let us consider a generic j-th column and define $\mathbf{s}' = \mathbf{s} \oplus \mathbf{h}_j$; because of the previous reasoning, the following properties hold:

i) if $j \notin \mathrm{S}(\mathbf{e})$ then, with high probability, we have $\mathrm{wt}(\mathbf{s}') > \mathrm{wt}(\mathbf{s})$;

ii) if $j \in \mathrm{S}(\mathbf{e})$ then, with high probability, we have $\mathrm{wt}(\mathbf{s}') < \mathrm{wt}(\mathbf{s})$.

In particular, the difference between the weights of $\mathbf{s}'$ and $\mathbf{s}$ is exactly what the BF decoder uses as the criterion to estimate error affected positions. Indeed, let σ_j be computed as in Algorithm 1; we then have

$$\mathrm{wt}(\mathbf{s}) - \mathrm{wt}(\mathbf{s}') = 2v - \sigma_j.$$

The more σ_j is close to v, the larger the reduction in the weight is.

We point out that σ_j, which is normally called *counter*, can also be seen as the number of unsatisfied parity-check equations in which the j-th bit participates: with this description, the BF principle can be resumed in a few words. Indeed, if a parity-check equation is unsatisfied, this means that its support overlaps with that of the error vector in an odd number of positions. Since both the rows of $\mathbf{H}$ and $\mathbf{e}$ are sparse, the majority of parity-check equations will overlap with the error in no more than one position. Then, if a bit participates in a big number of unsatisfied parity-check equations, then it is probably the only "responsible" of their parity-check values: thus, this bit can be safely considered as error affected.

The above reasoning should make explicit (in a qualitative way) that BF-decoding is intrinsically probabilistic; as we have already anticipated, this feature has a crucial (negative) influence on code-based crytosystems, as we explain in Chapter 6. In particular, all currently known efficient decoders for LDPC codes fails with some non trivial probability, which is normally assessed through numerical simulations. Indeed, the iterative nature of the decoder makes devising theoretical models for the failure

probability a really challenging (and quite involved) task. The existence of this probability represents a serious problem in cryptography since, as we explain in Section 6, it opens up for the possibility of mounting key recovery attacks. To avoid this issue, the failure probability must be kept below some value which, typically, is far beyond the ones that can be reached through mere numerical simulations. For instance, nowadays applications require failure probabilities to be, at least, lower than 2^{-128}. Thus, providing strong and reliable theoretical tools to predict the failure probability of a BF decoder has now become a major need in cryptography. In the following sections we deal with this problem, and describe some ways to estimate such probability.

One final remark is about the flipping strategy employed in Algorithm 1. This algorithm is normally referred to as *out-of-place* BF-decoder, since the error vector and syndrome updates are performed after all the bits have been evaluated. A different approach is that of *in-place* decoding, in which each bit is eventually flipped after its evaluation (i.e., the counter computation). The two approaches, despite being based on the same principles, may have significantly different performances, concerning both error correction capacities and algorithmic complexity. In this manuscript we have focused on the out-of-place strategy, which is normally the one actually implemented in code-based cryptosystems.

4.1.1 The error correction capability of a BF-decoder

Historically, the first bound on the error correction capability of the out-of-place BF decoder dates back to 2008, and is due to the work of Chilappagari et al. [48]. It is based on the concept of *girth* which, for a parity-check matrix $\mathbf{H}$, is defined as the shortest cycle in the Tanner graph.

Theorem 1 *For a code defined by a parity-check matrix $\mathbf{H}$ with girth g in which every column has weight v, BF decoding with decoding threshold $b = \lceil \frac{v}{2} \rceil$ allows correction of all error patterns of weight less than*

$$\begin{cases} \frac{1}{2} + \frac{v}{4} \sum_{i=0}^{j-1} \left(\frac{v-2}{2}\right)^i & \text{if } g = 4j+2, \\ \sum_{i=0}^{j-1} \left(\frac{v-2}{2}\right)^i & \text{if } g = 4j. \end{cases} \tag{4.1}$$

Then, in 2018 Tillich [49] established a connection between the *maximum column intersection* and the error correction capability, as stated by the following theorem.

Theorem 2 *Consider a code defined by a parity-check matrix for which every column has weight at least v and with maximum column intersection is δ, which is defined as*

$$\delta = \max_{\substack{i,j \in [0;n-1] \\ i \neq j}} \{|\mathrm{S}(\mathbf{h}_i) \cap \mathrm{S}(\mathbf{h}_j)|\}.$$

Then, one iteration of BF-decoder with threshold $b = \lceil \frac{v}{2} \rceil$ *allows the correction of all error vectors with weight* $t \leq t_M$*, where* $t_M = \lfloor \frac{v}{2\delta} \rfloor$.

For $g = 4$, $g = 6$ and $g = 8$, the bounds on the error correction capability computed according to Theorem 1 are 0, $\lfloor \frac{v+2}{4} \rfloor$ and $\lfloor \frac{v}{2} \rfloor$, respectively. So, for $g = 4$ (4.1) is useless. Theorem 2 can instead take into account the case of $g = 4$ and, if $\delta \leq \frac{v}{2}$, the obtained error correction is always ≥ 1. However, Theorem 2 does not take into account the girth of the code. Indeed, it can be easily shown that, when $g \geq 6$, we have $\delta = 1$: in all such cases, the error correction capability is always the same, and corresponds to $\lfloor \frac{v}{2} \rfloor$.

The bound expressed by Theorem 2 has then been improved by Santini et al. in [50], by considering the whole parity-check matrix structure and the possibility of employing different threshold values. We will describe these results in the following section.

4.1.2 An improved bound on the error correction capability

The results provided in [50] are based on the concept of *partial parity-check matrices*, which we briefly recall in the following.

Definition 14 *Given* $\mathbf{H}$*, let us consider the rows of* $\mathbf{H}$ *indexed by* $\mathrm{S}(\mathbf{h}_i)$ *and put them into a matrix* $\mathbf{H}^{(i)}$*. We define* $\mathbf{H}^{(i)}$ *as the* i*-th* partial parity-check matrix.

If the i-th column of $\mathbf{H}$ has weight v_i, then $\mathbf{H}^{(i)}$ is a $v_i \times n$ matrix, and its i-th column contains only ones. Furthermore, when decoding a syndrome $\mathbf{s} = \mathbf{H}\mathbf{e}^\top$, the number of unsatisfied parity-check equations in which the i-th bit participates, which we call counter and denote as σ_i, is

$$\sigma_i = \mathbf{H}^{(i)} * \mathbf{e}^\top,$$

where $*$ denotes the integer product. The partial parity-check matrices can be used to define the error correction capability of a given matrix $\mathbf{H}$; to this end, we first introduce another quantity, which we will then use in the main result of this section.

Definition 15 *Let* $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ *and* $\mathbf{H}^{(i)}$ *be the corresponding* i*-th partial parity-check matrix, and denote its* j*-th column as* $\mathbf{h}_j^{(i)}$*. For* $z \in [1; n-1]$*, let*

$$\delta^{(i)}(\mathbf{H}^{(i)}, z) = \max_{\substack{M \subseteq [0; n-1] \\ |M| = z,\ i \notin M}} \left\{ \mathrm{wt}\left(\bigoplus_{j \in M} \mathbf{h}_j^{(i)} \right) \right\},$$

where M *is a set containing the indexes of* z *columns of* $\mathbf{H}^{(i)}$*, except for the* i*-th. We call the* maximum column intersection of order z*, and denote as* $\delta(\mathbf{H}, z)$*, the quantity*

defined as

$$\delta(\mathbf{H}, z) = \max_{0 \leq i \leq n-1} \left\{ \delta^{(i)}(\mathbf{H}^{(i)}, z) \right\}.$$

Basically, $\delta^{(i)}(\mathbf{H}^{(i)}, z)$ corresponds to the maximum weight of a vector that can be obtained as the binary sum of z columns of $\mathbf{H}^{(i)}$, except for the i-th one. Then, $\delta(\mathbf{H}, z)$ is obtained as the maximum weight of all such vectors, computed considering all indexes $i \in [0; n-1]$. It is easy to see that, in the case of $z = 1$, $\delta(\mathbf{H}, z)$ corresponds to the maximum column intersection defined in Theorem 2.

We now describe how the columns of the partial parity-check matrices can be related to the counters that are used in the BF decoder. To this end, we consider a codeword affected by an error vector $\mathbf{e} \in \mathbb{F}_2^n$. We focus on a generic i-th bit, and denote with v_i the weight of the i-th column in $\mathbf{H}$. We remember that σ_i corresponds to the number of unsatisfied parity-check equations in which the i-th bit participates. If that bit is affected by an error, that is, $e_i = 1$, then we denote σ_i as $\sigma_i^{(1)}$, and we have

$$\begin{aligned} \sigma_i^{(1)} &= \mathbf{h}_i^{(i)} - \mathrm{wt}\left(\bigoplus_{j \in \mathrm{S}(\mathbf{e}) \setminus i} \mathbf{h}_j^{(i)} \right) \\ &= v_i - \mathrm{wt}\left(\bigoplus_{j \in \mathrm{S}(\mathbf{e}) \setminus i} \mathbf{h}_j^{(i)} \right) \end{aligned} \tag{4.2}$$

Indeed, if an entry in $\bigoplus_{j \in \mathrm{S}(\mathbf{e}) \setminus i} \mathbf{h}_j^{(i)}$ is a 1, this means that an odd number of errors at positions other than i participate in the corresponding parity-check equation, thus making the corresponding parity check equal to zero. In the same way, when the i-th bit is error free, that is, $e_i = 0$, we denote σ_i as $\sigma_i^{(0)}$, and we have

$$\sigma_i^{(0)} = \mathrm{wt}\left(\bigoplus_{j \in \mathrm{S}(\mathbf{e})} \mathbf{h}_j^{(i)} \right). \tag{4.3}$$

We are now ready to describe how the maximum column intersection of order larger than 1 can be exploited to derive a bound on the number of errors that can be corrected by one iteration of BF decoding; this result in stated in the following theorem.

Theorem 3 *Let us consider a code defined by a parity-check matrix* $\mathbf{H}$ *for which every column has weight at least* v*. Let* t *be an integer such that*

$$v > \delta(\mathbf{H}, t) + \delta(\mathbf{H}, t-1).$$

Then, one iteration of a BF decoder, with decoding threshold set as

$$b \in [\delta(\mathbf{H}, t) + 1, v - \delta(\mathbf{H}, t-1)]\,,$$

corrects all the error vectors of weight t.

Proof. One iteration of BF decoding can correct any error vector $\mathbf{e}$ of weight t if there exists a value of b such that

$$\min_{\substack{\mathbf{e}\in\mathbb{F}_2^n \\ \mathrm{wt}(\mathbf{e})=t}} \{\sigma_i^{(1)}\} \geq b > \max_{\substack{\mathbf{e}\in\mathbb{F}_2^n \\ \mathrm{wt}(\mathbf{e})=t}} \{\sigma_j^{(0)}\},\ \ \forall i \in \mathrm{S}\,(\mathbf{e})\,,\ \ \forall j \notin \mathrm{S}\,(\mathbf{e})\,; \tag{4.4}$$

indeed, the above condition guarantees that erred bits participate in a number of unsatisfied parity-check equations that is always larger than that of correct bits. Thus, erred bits can always unambiguously be distinguished from correct ones.

Let us first consider a bit at position $i \in [0, n-1]$, such that $e_i = 1$; based on (4.2), we have

$$\begin{aligned}
\sigma_i^{(1)} &= v_i - \mathrm{wt}\left(\bigoplus_{j\in\mathrm{S}(\mathbf{e})\setminus i} \mathbf{h}_j^{(i)}\right) \\
&\geq v - \mathrm{wt}\left(\bigoplus_{j\in\mathrm{S}(\mathbf{e})\setminus i} \mathbf{h}_j^{(i)}\right) \\
&\geq v - \delta(\mathbf{H}, t-1),
\end{aligned} \tag{4.5}$$

where $\mathrm{wt}\left(\sum_{j\in\mathrm{S}(\mathbf{e})\setminus i} \mathbf{h}_j^{(i)}\right) \leq \delta(\mathbf{H}, t-1)$ by definition.

Let us also consider the j-th bit, $j \neq i$, such that $e_j = 0$; based on (4.3), we have

$$\sigma_j^{(0)} = \mathrm{wt}\left(\bigoplus_{k\in\mathrm{S}(\mathbf{e})} \mathbf{h}_k^{(j)}\right) \leq \delta(\mathbf{H}, t). \tag{4.6}$$

By including (4.5) and (4.6) into (4.4), we obtain

$$v - \delta(\mathbf{H}, t-1) \geq b > \delta(\mathbf{H}, t), \tag{4.7}$$

from which the following condition is derived

$$v - \delta(\mathbf{H}, t-1) > \delta(\mathbf{H}, t). \tag{4.8}$$

According to (4.7), any $b \in [\delta(\mathbf{H}, t) + 1, v - \delta(\mathbf{H}, t-1)]$ guarantees that, on the one hand, all bits such that $e_j = 0$ are characterized by values of $\sigma_j^{(0)}$ that never exceed b and, thus, are not flipped; on the other hand, all bits such that $e_i = 1$ are characterized by values of $\sigma_i^{(1)}$ larger than or equal to b, and thus are flipped. ■

When $\delta(\mathbf{H}, t)$ is a non decreasing function of t (as it commonly happens), then the largest value of t such that Theorem 3 is satisfied provides the maximum error correction capability of the code. We prove in Corollary 1 that this bound on the error correction capability improves upon the one provided by Theorem 2.

Corollary 1 *For a code defined by a parity-check matrix* $\mathbf{H}$ *for which every column has weight not smaller than* v*, let* $t_M^{(2)}$ *be the largest integer such that Theorem 3 is satisfied, and* $\delta(\mathbf{H}, i) \leq \delta(\mathbf{H}, j),\ \forall i < j \leq t_M^{(2)}$*. Let* $t_M^{(1)}$ *denote the correction capability of BF decoding given in Theorem 2; then,* $t_M^{(1)} \leq t_M^{(2)}$*.*

Proof. Let us remind that the maximum weight of the j-th column of $\mathbf{H}^{(i)}$, where $i \in [0, n-1]$ and $j \neq i$, corresponds to $\delta^{(i)}(\mathbf{H}^{(i)}, 1)$. The following expression holds

$$\delta(\mathbf{H}, z) \leq z\delta(\mathbf{H}, 1). \tag{4.9}$$

Indeed, $\delta(\mathbf{H}, 1)$ corresponds to the maximum weight of a column in any partial parity-check matrix $\mathbf{H}^{(i)}$, excluding the i-th one; when we pick and sum z columns together, the weight of the resulting vector cannot be larger than the sum of the weights of the selected columns, which have weight larger than or equal to $\delta(\mathbf{H}, 1)$. From Theorem 2, we have

$$t_M^{(1)} \leq \left\lfloor \frac{v}{2\delta(\mathbf{H}, 1)} \right\rfloor \leq \frac{v}{2\delta(\mathbf{H}, 1)},$$

from which

$$v \geq t_M^{(1)}\delta(\mathbf{H}, 1). \tag{4.10}$$

By taking into account (4.9), inequality (4.10) can be developed as

$$\begin{aligned} v &\geq 2t_M^{(1)}\delta(\mathbf{H}, 1) \\ &> t_M^{(1)}\delta(\mathbf{H}, 1) + (t_M^{(1)} - 1)\delta(\mathbf{H}, 1) \\ &\geq \delta(\mathbf{H}, t_M^{(1)}) + \delta(\mathbf{H}, t_M^{(1)} - 1). \end{aligned}$$

This proves that any value of $t_M^{(1)}$ satisfies Theorem 3. Since $t_M^{(2)}$ is, by definition, the maximum value of t which satisfies Theorem 3, it must be $t_M^{(2)} \geq t_M^{(1)}$. ■

Computation of the gap between $t_M^{(2)}$ and $t_M^{(1)}$ implies to solve (4.8). However, computing the exact value of $\delta(\mathbf{H}, z)$ may be prohibitively complex when n and z are high. Indeed, in principles, all sums of z columns, for all partial parity-check matrices $\mathbf{H}^{(i)}$, need to be considered. Thus, a naive computation would require to test $n\binom{n-1}{z}$ sums which, clearly, becomes unfeasible unless n or z are trivial. For this reason, we consider an upper bound on $\delta(\mathbf{H}, z)$ which, on the one hand, is expected to be tight for sparse parity-check matrices and, on the other hand, can be easily computed.

Definition 16 *Given* $\mathbf{H}$ *and the* i*-th partial parity-check matrix* $\mathbf{H}^{(i)}$*,* $\forall i$*, we define* $U^{(i)}(\mathbf{H}^{(i)}, z) = \{u_0^{(i)}, \ldots, u_{z-1}^{(i)}\}$ *as any of the sets containing the weights of the* z

columns of $\mathbf{H}^{(i)}$*, except for the* i*-th, with the largest weights, and* $\mu^{(i)}(\mathbf{H}^{(i)}, z) = \sum_{j=0}^{z-1} u_j^{(i)}$*, where the sum is performed over the set of integers* $\mathbb{Z}$*. We then define the* maximum column union of order z*, denoted as* $\mu(\mathbf{H}, z)$*, the quantity*

$$\mu(\mathbf{H}, z) = \max_{0 \leq i \leq n-1} \left\{ \mu^{(i)}(\mathbf{H}^{(i)}, z) \right\}.$$

Lemma 1 *Given a code with blocklength* n *defined by the parity-check matrix* $\mathbf{H}$*, the following inequalities hold*

$$\delta(\mathbf{H}, z) \leq \mu(\mathbf{H}, z) \leq z\delta(\mathbf{H}, 1), \quad \forall z \leq n-1. \tag{4.11}$$

Proof. Given $\mathbf{H}^{(i)}$, $\forall i$, the sum of the weights of z columns with the largest weights, except for the ith, upper bounds the value of $\delta^{(i)}(\mathbf{H}^{(i)}, z)$. Then, the first inequality in (4.11) easily derives from Definition 15. Furthermore, z times the weight of the column of $\mathbf{H}^{(i)}$ with the largest weight, except for the ith, cannot be smaller than the sum of the weights of its z columns with the largest weights, excluding the ith. This proves the second inequality. ■

The value of $\mu(\mathbf{H}, z)$ can clearly be computed more easily than $\delta(\mathbf{H}, z)$, as it only depends on the computation of the weights of the columns of the partial parity-check matrices. Additionally, we also have $\mu(\mathbf{H}, i) \leq \mu(\mathbf{H}, j), \ \ \forall i < j$. Based on these premises, we can use the maximum column union values to determine a lower bound on the error correction capability of the decoder. In particular, the error correction capability of the decoder can be underestimated as

$$t_M \geq \max_t \{t \ \text{ s.t. } \ v > \mu(\mathbf{H}, t) + \mu(\mathbf{H}, t-1)\}. \tag{4.12}$$

Indeed, because of Lemma 1, condition (4.12) is always more restrictive than that of Theorem 3; then, setting the threshold of the decoder as $\mu(\mathbf{H}, t_M) + 1$ guarantees the correction of all error patterns with weight up to t_M. It can be shown that, with a similar proof of that of Corollary 1 and also considering Lemma 1, the bound on the error correction capability of the decoder defined by (4.12) is never worse than that of Theorem 2. Notice that, for sparse matrices, we can expect $\delta(\mathbf{H}, z)$ and $\mu(\mathbf{H}, z)$ to be very close, unless z assumes very large values. Indeed, the columns of the partial matrices have weights that are much smaller than v with high probability. Then, when summing z such columns, the expected number of cancellations is low, which means that $\mu(\mathbf{H}, z)$ exceeds $\delta(\mathbf{H}, z)$ by a small quantity with high probability.

To validate our approach, we have performed numerical simulations considering randomly generated parity-check matrices. To speed up the simulations, we have considered codes described by parity-check matrices in the form

$$\mathbf{H} = [\mathbf{H}_0, \mathbf{H}_1],$$

where each matrix $\mathbf{H}_i$ is a circulant of weight v. In particular, a circulant matrix is such that each row corresponds to the cyclic shift by one position of the previous row; a formal definition of circulant matrices is provided in Chapter 5. The corresponding code, which is called quasi-cyclic (QC) because of the particular structure of the parity-check matrix, has length $n = 2p$ and rate $\frac{1}{2}$. This choice is motivated by the fact that, because of the QC structure, we just need to analyze the partial parity-check matrices $\mathbf{H}^{(0)}$ and $\mathbf{H}^{(p)}$. In fact, it is straightforward to verify that, as a consequence of the QC structure, partial matrices associated to columns belonging to the same circulant block are formed by the same columns, but in a different order.

The results of our simulations are shown in Table 4.1; the error correction capability has been underestimated through the maximum column union. The number of tested matrices and the number of occurrences of each value of t_M are reported in the last column of the table. As we can see from the tables, the analysis based on the maximum column union provides results that considerably improve upon the ones obtained through Theorem 2.

Table 4.1: Estimated t_M for randomly generated $(v, 2v)$-regular QC codes

v	p	t_M Th. 2 [49]	t_M Th. 3	No. occurrences
450	48000	13	≥ 16	31 out of 123
			≥ 17	91 out of 123
			≥ 18	1 out of 123
600	100000	18	≥ 22	2 out of 162
			≥ 23	82 out of 162
			≥ 24	75 out of 162
			≥ 25	3 out of 162
600	200000	25	≥ 31	1 out of 144
			≥ 32	26 out of 144
			≥ 33	111 out of 144
			≥ 34	6 out of 144
900	400000	34	≥ 43	2 out of 127
			≥ 44	21 out of 127
			≥ 45	73 out of 127
			≥ 46	26 out of 127
			≥ 47	5 out of 127

4.1.3 Error sets and failure probability for one BF iteration

In this section we briefly resume the results of [51], and describe an analytical tool to assess the decoding failure probability of one BF iteration. The analysis we describe does not require any assumption, and takes into account the parity-check matrix structure to overestimate the number of error vectors of given weight that cannot

be corrected by the decoder. Before proceeding with our analysis, we introduce the concept of *adjacency matrix*, which is at the basis of the results we discuss.

Definition 17 *Given a matrix* $\mathbf{H} \in \mathbb{F}_2^{r\times n}$*, the* adjacency matrix *of* $\mathbf{H}$*, denoted as* $\mathbf{\Gamma}$*, is the* $n \times n$ *matrix whose element in position* (i, j) *is such that*

$$\gamma_{i,j} = \begin{cases} |\mathrm{S}(\mathbf{h}_i) \cap \mathrm{S}(\mathbf{h}_j)| & \text{if} \quad i \neq j \\ 0 & \text{if} \quad i = j \end{cases}.$$

We consider the first and only iteration of a BF decoder, and assume that the employed decoding thresholds may vary with the position. Thus, given $[b_0, b_1, \cdots, b_{n-1}]$, the i-th bit is flipped if and only if $\sigma_i \geq b_i$. We denote with $\mathcal{U}(B_{n,t})$ the uniform distribution over the binary n-uples of weight t, and analyze the decoder behaviour when processing a syndrome $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$, with $\mathbf{e} \sim \mathcal{U}(B_{n,t})$. As we have described in Chapter 3, having a fixed number of errors (t) is clearly a scenario of interest in code-based cryptography. Nevertheless, once having characterized the decoder performance for a given number of errors, it is easy to extend such a characterization to channel models (like the Binary Symmetric Channel (BSC)) in which the statistic of the number of errors is known. In fact, a BSC with crossover probability ρ can be straightforwardly studied by considering that the probability that the channel introduces exactly t errors is equal to $\Pr[\mathrm{wt}(\mathbf{e}) = t] = \binom{n}{t}\rho^t(1-\rho)^{n-t}$. So, denoting the error vector after the first iteration as $\mathbf{e}'$, and the decoding failure probability as P_f, for the BSC we have

$$P_f = \sum_{t=0}^{n} \Pr[\mathbf{e}' \neq \mathbf{e} \mid \mathrm{wt}(\mathbf{e}) = t] \Pr[\mathrm{wt}(\mathbf{e}) = t],$$

where $\Pr[\mathbf{e}' \neq \mathbf{e} \mid \mathrm{wt}(\mathbf{e}) = t]$ can be upper bounded through the method we describe in this section. For the sake of brevity, from now on we only focus on the case in which t is constant and fixed.

For $i \in [0, n-1]$, we define f_i as the binary variable obtained through the following rule

$$f_i = \begin{cases} 0 & \text{if } [(\sigma_i < b_i) \wedge (e_i = 0)] \vee [(\sigma_i \geq b_i) \wedge (e_i = 1)], \\ 1 & \text{if } [(\sigma_i \geq b_i) \wedge (e_i = 0)] \vee [(\sigma_i < b_i) \wedge (e_i = 1)]. \end{cases} \tag{4.13}$$

In other words, when $f_i = 0$, the decoder takes a right decision on the i-th bit. Conversely, when $f_i = 1$, the decoder takes a wrong decision on the i-th bit; a wrong decision can either be the flip of an error-free bit or the missing flip of a bit affected by an error. The error patterns that cause a decoding error in the i-th position are defined by the so-called *error sets*, which we introduce next.

Definition 18 *Let* $\mathbf{H} \in \mathbb{F}_2^{r\times n}$ *be the parity-check matrix of a code with blocklength* n*. We consider the first and only iteration of a BF decoder, with decoding thresholds* $[b_0, \cdots, b_{n-1}]$*. Let* f_i *be the binary variable defined as in* (4.13)*, for* $i \in [0; n-1]$*.*

Then, for $z \in \{0,1\}$, we define the error set *for the i-th bit as follows*

$$\mathcal{E}_{i,t,b_i}^{z} = \{\, \mathbf{e} \in \mathcal{B}_t \;\; s.t. \;\; f_i = 1 |\, e_i = z \}\,.$$

As we show in the following, a fundamental quantity in establishing the error correction capability of the first iteration of a BF decoder is represented by the cardinality of the error sets; to this end, in order to ease the notation, we define the following quantity.

Definition 19 *Let $P_{l,m} = \{p_0, \cdots, p_{m-1}\}$ be a set of distinct integers $0 \leq p_i < l$, with $l \geq m$ and $p_i \neq p_j$, $\forall i, j$. We define $\mathcal{P}_{l,m}$ as the ensemble containing all such distinct non-ordered sets; clearly, $|\mathcal{P}_{l,m}| = \binom{l}{m}$. Let $\alpha \in \mathbb{N}$ and let $\mathbf{a} \in \mathbb{N}^l$ be a length-l vector of non-negative integers; then, we define*

$$\mathcal{N}_{m,\alpha}^{\mathbf{a}} = \left\{ P_{l,m} \in \mathcal{P}_{l,m} \;\; s.t. \;\; \sum_{i=0}^{m-1} a_{p_i} > \alpha \right\}.$$

We now introduce a property of the error sets that will be crucial in the proof of Theorem 4 that we state next.

Lemma 2 *Let $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ be a parity-check matrix, with adjacency matrix $\mathbf{\Gamma}$, and let $\mathcal{E}_{i,t,b_i}^{z}$, for $z \in \{0,1\}$, be the error set for the i-th bit. We denote with $\tilde{\boldsymbol{\gamma}}^{(i)}$ the vector formed by the entries of the i-th row of the adjacency matrix $\mathbf{\Gamma}$, except for the i-th one. Then, we have*

$$\left|\mathcal{E}_{i,t,b_i}^{1}\right| \leq \left|\mathcal{N}_{t-1,v_i-b_i}^{\tilde{\boldsymbol{\gamma}}^{(i)}}\right|, \tag{4.14}$$

$$\left|\mathcal{E}_{i,t,b_i}^{0}\right| \leq \left|\mathcal{N}_{t,b_i-1}^{\tilde{\boldsymbol{\gamma}}^{(i)}}\right|. \tag{4.15}$$

Proof. We focus on the i-th bit, characterized by a certain value of σ_i and flipping threshold b_i, and derive the conditions upon which the decoder takes a wrong decision (i.e., $f_i = 1$). We first consider the case of $e_i = 1$: a wrong decision is taken if the decoder does not flip the bit, i.e., if $\sigma_i < b_i$. From (4.2), we know that the value of σ_i is not lower than the difference between the weight of the i-th column (that is, v_i) and the sum of the values $\gamma_{i,j}$ indexed by $S(\mathbf{e})$, except the i-th index (that is, $\sum_{j \in S(\mathbf{e}) \setminus i} \gamma_{i,j}$). If such difference is not lower than b_i, then the condition $\sigma_i \geq b_i$ is guaranteed and the decoder flips the i-th bit. On the other hand, if $v_i - \sum_{j \in S(\mathbf{e}) \setminus i} \gamma_{i,j} < b_i$, σ_i might be lower than b_i and the decoder does not flip the i-th bit. Then, a necessary (but not sufficient condition) to have a wrong decision on the i-th bit is

$$\sum_{j \in S(\mathbf{e}) \setminus i} \gamma_{i,j} > v_i - b_i. \tag{4.16}$$

Because of the above reasoning, $\mathcal{E}_{i,t,b_i}^{1}$ is a subset of the error vectors satisfying (4.16). The set $S(\mathbf{e}) \setminus i$ in (4.16) corresponds to a subset of $[0, i-1] \cup [i+1, n-1]$, of size

$t-1$; furthermore, the values $\gamma_{i,j}$ which are possibly selected by $S(\mathbf{e}) \setminus i$ are entries of $\tilde{\boldsymbol{\gamma}}^{(i)} = [\gamma_{i,0}, \cdots, \gamma_{i,i-1}, \gamma_{i,i+1}, \cdots, \gamma_{i,n-1}]$, which has length $n-1$. Let $P_{n-1,t-1}$ be a subset of $[0, n-1]$, such that the sum of the entries in $\tilde{\boldsymbol{\gamma}}^{(i)}$ indexed by $P_{n-1,t-1}$ is larger than $v_i - b_i$. According to Definition 18, the number of such sets corresponds to the cardinality of $\mathcal{N}^{\tilde{\boldsymbol{\gamma}}^{(i)}}_{t-1,v_i-b_1}$. Furthermore, to each one of these subsets, we can associate an error vector satisfying (4.16), with support

$$\{j \in P_{n-1,t-1} \,|j < i\} \cup i \cup \{j+1 \in P_{n-1,t-1} \,|j > i\}.$$

Thus, we obtain

$$\begin{aligned}\left|\mathcal{E}^1_{i,t,b_i}\right| &\leq \left|\left\{e \in \mathcal{B}_t \text{ s.t. } (e_i = 1) \wedge \left(\sum_{j \in S(\mathbf{e}) \setminus i} \gamma_{i,j} > v_i - b_i\right)\right\}\right| \\ &= \left|\mathcal{N}^{\tilde{\boldsymbol{\gamma}}^{(i)}}_{t-1,v_i-b_i}\right|.\end{aligned}$$

Similarly, for the case of $e_i = 0$, we can derive from (4.3) that a necessary but not sufficient condition for $f_i = 1$ is $b_i \leq \sigma_i \leq \sum_{j \in S(e)} \gamma_{i,j}$. Similarly to the case of $e_1 = 1$, we have

$$\begin{aligned}\left|\mathcal{E}^0_{i,t,b_i}\right| &\leq \left|\left\{e \in \mathcal{B}_t \text{ s.t. } (e_i = 0) \wedge \left(\sum_{j \in S(e)} \gamma_{i,j} > b_i - 1\right)\right\}\right| \\ &= \left|\mathcal{N}^{\tilde{\boldsymbol{\gamma}}^{(i)}}_{t,b_i-1}\right|.\end{aligned}$$

■

Based on these relationships, we can now prove the following main theorem, that allows defining an upper bound on the decoding failure probability of one iteration of BF decoding, when the error vector corrupting a codeword is uniformly picked over $\mathcal{B}_t$.

Theorem 4 *Let $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ be a parity-check matrix. Let $\mathbf{e} \in \mathcal{B}_t$, and $\mathbf{s} = \mathbf{e}\mathbf{H}^T$ be the corresponding syndrome. We consider a single BF iteration applied on $\mathbf{s}$, with decoding threshold for the i-th bit denoted as b_i. Let $\tilde{\boldsymbol{\gamma}}^{(i)}$ denote the vector formed by the elements in the i-th row of $\boldsymbol{\Gamma}$, except for the i-th one. The probability that the decoder fails to decode $\mathbf{s}$ is upper bounded as follows*

$$P_f \leq \min\left\{1; \frac{\sum_{i=0}^{n-1}\left(|\mathcal{N}^{\tilde{\boldsymbol{\gamma}}^{(i)}}_{t-1,v_i-b_i}| + |\mathcal{N}^{\tilde{\boldsymbol{\gamma}}^{(i)}}_{t,b_i-1}|\right)}{\binom{n}{t}}\right\}.$$

Proof. Let $\mathcal{E}_{i,t,b_i}$ be the set of all error vectors such that $f_i = 1$; clearly

$$\mathcal{E}_{i,t,b_i} = \mathcal{E}^0_{i,t,b_i} \cup \mathcal{E}^1_{i,t,b_i}.$$

By considering that $\mathcal{E}^1_{i,t,b_i}$ and $\mathcal{E}^0_{i,t,b_i}$ are disjoint, as a bit may be either correct or incorrect, and by taking into account (4.14) and (4.15), we obtain

$$\begin{aligned}|\mathcal{E}_{i,t,b_i}| &= \left|\mathcal{E}^0_{i,t,b_i}\right| + \left|\mathcal{E}^1_{i,t,b_i}\right| \\ &\leq \left|\mathcal{N}^{\tilde{\gamma}^{(i)}}_{t-1,v_i-b_i}\right| + \left|\mathcal{N}^{\tilde{\gamma}^{(i)}}_{t,b_i-1}\right|.\end{aligned}$$

Then, the probability that decoding of $\mathbf{s} = \mathbf{e}\mathbf{H}^T$ fails can be upper bounded by means of the following chain of inequalities

$$\begin{aligned}P_f &= \frac{\left|\bigcup_{i=0}^{n-1} \mathcal{E}_{i,t,b_i}\right|}{|\mathcal{B}_t|} \\ &\leq \frac{\sum_{i=0}^{n-1} |\mathcal{E}_{i,t,b_i}|}{|\mathcal{B}_t|} \\ &\leq \frac{\sum_{i=0}^{n-1} \left(\left|\mathcal{N}^{\tilde{\gamma}^{(i)}}_{t-1,v_i-b_i}\right| + \left|\mathcal{N}^{\tilde{\gamma}^{(i)}}_{t,b_i}\right|\right)}{|\mathcal{B}_t|}.\end{aligned} \tag{4.17}$$

The thesis of the theorem is finally proved by considering that $|\mathcal{B}_t| = \binom{n}{t}$ and that trivially $P_f \leq 1$ (whereas the bound in (4.17) is not guaranteed to be smaller than or equal to 1). ■

An efficient way to compute the cardinalities of such sets is described in Appendix A, and applies only because, for LDPC codes, the entries of the adjacency matrix are extremely low. The expression of P_f derived above is coherent with Theorem 3. Indeed, let T_M denote the bound on the error correction capability: then, for all $t \leq T_M$, we have $\left|\mathcal{N}^{\tilde{\gamma}^{(i)}}_{t-1,v_i-b_i}\right| = 0$ and $\left|\mathcal{N}^{\tilde{\gamma}^{(i)}}_{t,b_i}\right| = 0$, from which $P_f = 0$.

The previous bound can be specialized, depending on the threshold values and on the parity-check matrix characteristics. From now on, we consider only regular codes, with column weight v; furthermore, we assume that the decoding threshold values are equal for all bits, and denote this unique value as b.

Then, when v is odd and $b = \lceil \frac{v}{2} \rceil$, the bound on P_f provided by Theorem 4 can be rewritten as

$$P_f \leq \min\left\{1; \frac{\sum_{i=0}^{n-1} \left|\mathcal{N}^{\gamma^{(i)}}_{t,\frac{v-1}{2}}\right|}{\binom{n}{t}}\right\}. \tag{4.18}$$

The proof is omitted for the sake of brevity.

When $g \geq 6$, we have

$$\gamma_{i,j} \in \{0, 1\}, \quad \forall i, j.$$

In particular, for (v, w)-regular codes, each row and each column of $\mathbf{\Gamma}$ contain exactly $v(w-1)$ non-zero entries. Then, the bound on P_f can further specialized; to this end, we first consider the following lemma.

Lemma 3 *Let $\mathbf{a} \in \mathbb{F}_2^l$ be a vector of weight m; then, we have $\left|\mathcal{N}_{x,\alpha}^{\mathbf{a}}\right| = \theta(l,x,m,\alpha)$, with*

$$\theta(l,x,m,\alpha) = \begin{cases} 0 & \text{if } \alpha > m \quad \text{or} \quad x \leq \alpha \\ \sum_{j=\alpha+1}^{\min\{m,x\}} \binom{m}{j}\binom{l-m}{x-j} & \text{otherwise} \end{cases}.$$

The following Theorem specializes Theorem 4 to the case of a regular code with girth larger than 4, and reformulates (4.18) for such a case.

Theorem 5 *Let $\mathbf{H} \in \mathbb{F}_2^{r\times n}$ be the parity-check matrix of a (v,w)-regular code with girth $g \geq 6$. Let $\mathbf{e} \in \mathcal{B}_t$, and $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$. We consider a single iteration of BF decoding applied to $\mathbf{s}$, with a unique decoding threshold b. If v is odd and $b = \lceil \frac{v}{2} \rceil$, we have*

$$\begin{cases} P_f = 0 & \text{if } t \leq \frac{v-1}{2} \\ P_f \leq \min\left\{1; \frac{n\theta(n,t,v(w-1),\frac{v-1}{2})}{\binom{n}{t}}\right\} & \text{otherwise} \end{cases},$$

where

$$\theta(n,t,v(w-1),\frac{v-1}{2}) = \sum_{j=\frac{v+1}{2}}^{\min\{v(w-1),t\}} \binom{v(w-1)}{j}\binom{n-v(w-1)}{t-j}.$$

Proof. The proof can be straightforwardly derived by taking into account Lemma 3. ■

We now compare the bounds we have derived with the results of numerical simulations. As in the previous section, we focus on the case of QC-LDPC codes with parity-check matrix in the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{H}_1 \end{bmatrix}, \tag{4.19}$$

where each $\mathbf{H}_i$, $i \in \{0,1\}$, is a circulant matrix of size p and row/column weight v. In this case, the matrix $\mathbf{\Gamma}$ can be written as

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{\Gamma}_{0,0} & \mathbf{\Gamma}_{0,1} \\ \mathbf{\Gamma}_{1,0} & \mathbf{\Gamma}_{1,1} \end{bmatrix},$$

where each $\mathbf{\Gamma}_{i,j}$ is a $p \times p$ matrix; in particular, $\mathbf{\Gamma}$ is symmetric, and this means that $\mathbf{\Gamma}_{0,0}$ and $\mathbf{\Gamma}_{1,1}$ are symmetric as well, while $\mathbf{\Gamma}_{0,1}^\top = \mathbf{\Gamma}_{1,0}$. Moreover, each block $\mathbf{\Gamma}_{i,j}$ is circulant. In particular, let $\boldsymbol{\gamma}^{(i)}$ be the i-th row of $\mathbf{\Gamma}$; then, all rows $\boldsymbol{\gamma}^{(j)}$ such that $\lfloor i/p \rfloor = \lfloor j/p \rfloor$ are identical up to a quasi-cyclic shift; this means that

$$\left|\mathcal{E}_{i,t,b}^z\right| = \left|\mathcal{E}_{j,t,b}^z\right|, \quad \forall b,t, \quad \forall i,j \text{ s.t. } \lfloor i/p \rfloor = \lfloor j/p \rfloor,$$

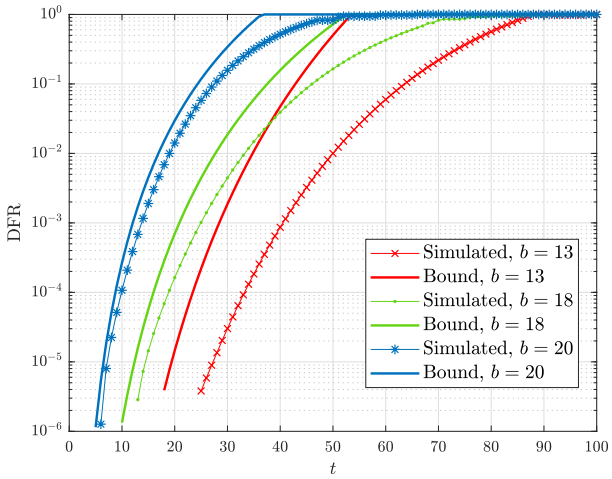


Figure 4.2: Comparison of the DFR resulting from Monte Carlo simulations with our bound for a code with $p = 9851$, $v = 25$, $g = 4$, and different threshold values.

with $z \in \{0, 1\}$. Then, from Theorem 4 we obtain

$$P_f \leq \min\left\{1; p\,\frac{\mathcal{N}_{\text{tot}}}{\binom{n}{t}}\right\},$$

with

$$\mathcal{N}_{\text{tot}} = \left|\mathcal{N}^{\tilde{\gamma}^{(0)}}_{t-1,v-b}\right| + \left|\mathcal{N}^{\tilde{\gamma}^{(0)}}_{t,b-1}\right| + \left|\mathcal{N}^{\tilde{\gamma}^{(p)}}_{t-1,v-b}\right| + \left|\mathcal{N}^{\tilde{\gamma}^{(p)}}_{t,b-1}\right|.$$

To assess the accuracy of the provided bounds, let us consider some codes defined by parity-check matrices as in (4.19). The first code we consider has $p = 9851$, $v = 25$ and $g = 4$; the second code has $p = 8779$, $v = 13$ and girth $g = 6$. We assess the DFR achieved by a single-iteration BF decoder with different threshold values through Monte Carlo simulations; for each value of t, the DFR has been estimated through the observation of 100 wrong decoding instances. The comparison of the simulation results with our bounds is shown in Figs. 4.2 and 4.3, respectively. From the figures it results that the bound becomes tighter and tighter for decreasing values of t.

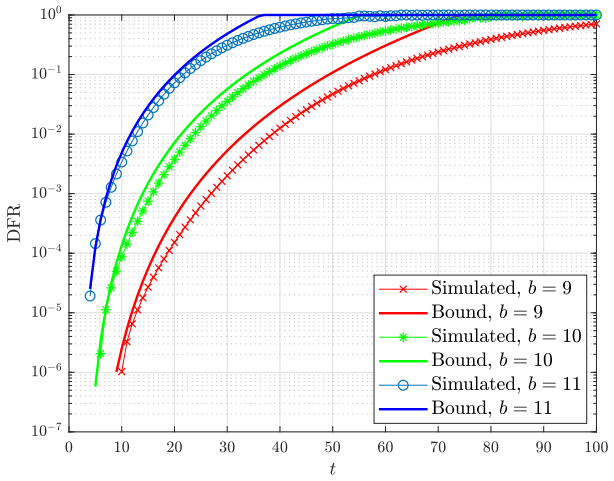


Figure 4.3: Comparison of the DFR resulting from Monte Carlo simulations with our bound, for a code with $p = 8779$, $v = 13$, $g = 6$, and different threshold values.

Chapter 5

A general framework for codes with regular geometric structure

In Chapter 3 we have introduced the McEliece and Niederreiter cryptosystems. We have considered some general attacks (such as ISD algorithms), and have focused on schemes based on binary Goppa codes and GRS codes, such as the BL [42] and the BCRSS [37]. Despite the recognized security of these proposal, they all suffer from large public keys which, as we have already said, is the Achille's heel of code-based cryptosystems.

To address this issue, a major research avenue through the years has been that of adding some geometrical structure to the secret code and mirroring it into the public key. In such cases, the additional public key structure can be exploited to achieve a compact representation. When the public key has such a structure, then transmitting only a bunch of its rows (or columns) is enough to provide enough information to reconstruct the whole matrix. This way, the public key size gets significantly reduced.

However, when algebraic codes are used, this choice may lead to some serious security flaws. Indeed, an algebraic code already has a very strong structure, i.e., has some binding relations constraining its representations. When some geometrical structure is added, these relations become more binding, and this may compromise the security of the private key. The most notable example of attacks against these solutions is that in [9]. Basically, in this cryptanalysis technique, the secret key is determined through a system of equations, which is obtained from the knowledge of the public key; the unknowns correspond to the elements in the secret key. Some equations are already provided by the algebraic structure of the underlying code; if we furthermore take into account the geometrical structure, then more equations can be added to the system. The number of solutions to the system becomes lower, such that an adversary may be able to test all of them and recover the secret key.

This attack strategy has seriously limited trust in cryptosystems based on algebraic structured codes; as an evidence of this fact, there are no NIST second round candidates based on structured algebraic codes. However, the situation becomes completely

different when the adopted family of codes is not algebraic and admits a random or pseudo-random design. In these cases, the secret key has a very weak algebraic structure, and the geometrical structure can be safely added. Some well established solutions use those based on random codes and LDPC codes.

In this section we generalize the idea of codes with a compact structure, by introducing the concept of *reproducibility*. We prove relations which guarantee the existence of matrices with a compact representation, and provide conditions that guarantee that such structure is invariant under matrix operations. We show how this framework encompasses existing families, such as those of cyclic, quasi-cyclic and quasi-dyadic codes, and provide examples of new concrete constructions. Finally, we describe how to instantiate cryptosystems based on such codes, with some possible benefits in defeating some attacks that speed-up solvers for the decoding problem (such as ISD). The results we present in this section are partly contained in [52].

5.1 Reproducible and quasi-reproducible codes

The idea of "compactness" that we use, is captured in the following definitions.

Definition 20 *Consider a matrix $\mathbf{A} \in \mathbb{F}_q^{k\times n}$. Let $\mathcal{R}$ be the set of the rows of $\mathbf{A}$ and let $2^{\mathcal{R}}$ be its power set. Let $\mathcal{F} = \{\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1, \cdots, \boldsymbol{\sigma}_\ell\}$ be a family of linear transformations on elements of $\mathbb{F}_q^{m\times n}$. We say that $\mathbf{A}$ is* $\mathcal{F}$-reproducible *if $\mathbf{A}$ can be entirely described as $\mathcal{F}(\mathbf{a})$, where $\mathbf{a}$ is an element of $2^{\mathcal{R}}$, of cardinality $m < k$, called the* signature set*, that is*

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}\sigma_0 \\ \mathbf{a}\boldsymbol{\sigma}_1 \\ \vdots \\ \mathbf{a}\boldsymbol{\sigma}_\ell \end{bmatrix}$$

Definition 21 *Let $\mathcal{F}$ be a family of linear transformations and $\mathcal{C}$ be a linear code over $\mathbb{F}_q$. If $\mathcal{C}$ can be described by an $\mathcal{F}$-reproducible generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ and/or an $\mathcal{F}$-reproducible parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{r\times n}$, then we say that $\mathcal{C}$ is in* $\mathcal{F}$-reproducible form*.*

Thus, a reproducible matrix is compactly described just by its signature set and by its family of linear functions. Consequently, having the generator matrix (and/or the parity-check matrix) in reproducible form leads to a compact representation of the code. The condition on the reproducibility of a matrix can be relaxed, in order to take into account other structures that allow a compact representation.

Definition 22 *Let $\mathbf{A}_{i,j} \in \mathbb{F}_q^{k_{i,j}\times n_{i,j}}$ be reproducible matrices, each with its own dimensions, signature set $\mathbf{a}_{i,j} \in \mathbb{F}_q^{m_{i,j}\times n_{i,j}}$ and family of linear functions $\mathcal{F}_{i,j}$. Let $\mathbf{A}$ be a matrix obtained using as building blocks the matrices $\mathbf{A}_{i,j}$; then, we say that $\mathbf{A}$ is* quasi-reproducible*.*

Definition 23 *Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$. If $\mathcal{C}$ can be described by a quasi-reproducible generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ and/or a quasi-reproducible parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{r\times n}$, then we say that $\mathcal{C}$ is in* quasi-reproducible form.

It is clear that, in order to describe a quasi-reproducible matrix, we just need the set of the signature sets of its building blocks, together with the corresponding families of linear functions. Quasi-reproducibility generalizes the concept of reproducibility, since each reproducible code can be seen as a particular quasi-reproducible code, with a generator matrix described just by one signature. A particular type of quasi-reproducible codes is the one in which the blocks $\mathbf{A}_{i,j}$ are square matrices, defined by the same family $\mathcal{F}$.

We are now ready to introduce a very important notion regarding the set of reproducible matrices obtained via a given family of transformations. Specifically, consider a family of linear functions $\mathcal{F} = \left\{\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1, \cdots, \boldsymbol{\sigma}_{\frac{p}{m}-1}\right\}$, where each $\boldsymbol{\sigma}_i$ is a $p \times p$ matrix over $\mathbb{F}_q$. We denote by $\mathcal{M}_q^{\mathcal{F},m}$ the set of all reproducible matrices over $\mathbb{F}_q$ obtained via signatures of size $m \times p$ and $\mathcal{F}$, equipped with the usual operations of matrix sum and multiplication. Then the following results[1] hold.

Theorem 6 *The set $\mathcal{M}_q^{\mathcal{F},m}$ is an abelian group with respect to the sum.*

Proof. Showing that $\mathcal{M}_q^{\mathcal{F},m}$ is an additive abelian group is quite straightforward. In fact, the signature of the sum of two matrices corresponds to the sum of the original signatures. Commutativity and associativity follow from the element-wise sum between two matrices. The identity is given by the null signature (i.e., the signature made of all zeros), while the inverse of a matrix with signature $\mathbf{a}$ is the matrix with signature $-\mathbf{a}$. ■

On the other hand, it is possible to show that the set, with respect to the multiplication, is a semigroup; in this case, the only requirements are closure and associativity. While associativity easily follows from the properties of the multiplication between two matrices, in order to guarantee closure, we must make an additional assumption.

Theorem 7 *$\mathcal{M}_q^{\mathcal{F},m}$ is a semigroup with respect to the multiplication if and only if for every matrix $\mathbf{M} \in \mathcal{M}_q^{\mathcal{F},m}$, we have*

$$\boldsymbol{\sigma}_i\mathbf{M} = \mathbf{M}\boldsymbol{\sigma}_i, \quad \forall i \in \mathbb{N}, 0 \leq i \leq \frac{p}{m} - 1.$$

Proof. We show that commutativity is necessary first. For what we discussed above, we only need to prove closure. Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices of $\mathcal{M}_q^{\mathcal{F},m}$, with respec-

[1] For simplicity we assume $\boldsymbol{\sigma}_0 = \mathbf{I}_p$, but this is not necessary and the results holds even if $\mathcal{F}$ does not contain the identity function.

tive signatures $\mathbf{a}_0$, $\mathbf{b}_0$, that is

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_0\boldsymbol{\sigma}_1 \\ \vdots \\ \mathbf{a}_0\boldsymbol{\sigma}_{\frac{p}{m}-1} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_{\frac{p}{m}-1} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_0\boldsymbol{\sigma}_1 \\ \vdots \\ \mathbf{b}_0\boldsymbol{\sigma}_{\frac{p}{m}-1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{\frac{p}{m}-1} \end{bmatrix}.$$

Multiplying these two matrices we get

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} \mathbf{a}_0\mathbf{B} \\ \mathbf{a}_1\mathbf{B} \\ \vdots \\ \mathbf{a}_{\frac{p}{m}-1}\mathbf{B} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0\mathbf{B} \\ \mathbf{a}_0\boldsymbol{\sigma}_1\mathbf{B} \\ \vdots \\ \mathbf{a}_0\boldsymbol{\sigma}_{\frac{p}{m}-1}\mathbf{B} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_{\frac{p}{m}-1} \end{bmatrix}.$$

Now by hypothesis

$$\mathbf{c}_i = \mathbf{a}_0\boldsymbol{\sigma}_i\mathbf{B} = \mathbf{a}_0\mathbf{B}\boldsymbol{\sigma}_i = \mathbf{c}_0\boldsymbol{\sigma}_i,$$

for all $i \leq \frac{p}{m} - 1$. It follows that $\mathbf{C}$ is reproducible and defined by $\mathcal{F}$.

Conversely, suppose $\mathcal{M}_q^{\mathcal{F},m}$ is a semigroup, and in particular that it is closed with respect to multiplication. Consider again two matrices $\mathbf{A}$ and $\mathbf{B}$ and their product, defined as in Equation 5.1. Since by hypothesis $\mathbf{C} \in \mathcal{M}_q^{\mathcal{F},m}$, and therefore is reproducible, we have that $\mathbf{c}_i = \mathbf{c}_0\boldsymbol{\sigma}_i$ for all $i \leq \frac{p}{m} - 1$. It follows that

$$\mathbf{a}_0\boldsymbol{\sigma}_i\mathbf{B} = \mathbf{c}_i = \mathbf{c}_0\boldsymbol{\sigma}_i = \mathbf{a}_0\mathbf{B}\boldsymbol{\sigma}_i.$$

Now, since equation (5.1) holds in general for every signature $\mathbf{a}_0$, it must be that $\boldsymbol{\sigma}_i\mathbf{B} = \mathbf{B}\boldsymbol{\sigma}_i$, which concludes the proof. ■

Finally, note that multiplication distributes over addition, as usual. This means that, if the conditions of Theorem 7 are satisfied, $\mathcal{M}_q^{\mathcal{F},m}$ verifies all the requisites of a mathematical *pseudo-ring*, i.e. a ring without multiplicative identity (also known as *rng*). We call this the *reproducible pseudo-ring* induced by $\mathcal{F}$ over $\mathbb{F}_q$.

5.2 Pseudo-rings induced by families of permutations

In the particular case of signatures made of just one row (i.e., $m = 1$) and the functions $\boldsymbol{\sigma}_i$ being permutations, further properties can be derived, as stated in Theorem 8. We point out that all the results we present in this section can be generalized, in order to consider the case $m > 1$, but we will not go into further details here. We first introduce some additional notation, which will be useful in the remainder of this section.

Since a $p \times p$ permutation corresponds to a matrix in which every row and column has weight equal to 1, it can equivalently be described as a bijection over $[0, p-1] \subset \mathbb{N}$. Given a permutation matrix $\boldsymbol{\sigma}_i$, we denote the corresponding bijection as $f_{\boldsymbol{\sigma}_i}$. If the element of $\boldsymbol{\sigma}_i$ in position (v, z) is equal to 1, then $f_{\boldsymbol{\sigma}_i}(v) = z$. The inverse of $f_{\boldsymbol{\sigma}_i}$, which we denoted as $f^{-1}_{\boldsymbol{\sigma}_i}$, is the bijection associated to the permutation matrix $\boldsymbol{\sigma}_i^{-1} = \boldsymbol{\sigma}_i^{\top}$; if $f_{\boldsymbol{\sigma}_i}(v) = j$, then $f^{-1}_{\boldsymbol{\sigma}_i}(j) = v$. Let $\mathbf{a} = [a_0, a_1, \cdots]$ and $\mathbf{a}' = [a'_0, a'_1, \cdots]$ be two row vectors, such that $\mathbf{a}' = \mathbf{a}\boldsymbol{\sigma}_i$. Then, $a'_j = a_{f^{-1}_{\boldsymbol{\sigma}_i}(j)}$. If instead $\mathbf{a}'^{\top} = \boldsymbol{\sigma}_i \mathbf{a}^{\top}$, then $a'_j = a_{f_{\boldsymbol{\sigma}_i}(j)}$. We use $f_{\boldsymbol{\sigma}_i} \circ f_{\boldsymbol{\sigma}_j}$ to denote the bijection defined by the application of $f_{\boldsymbol{\sigma}_i}$ after $f_{\boldsymbol{\sigma}_j}$. In other words, $f_{\boldsymbol{\sigma}_i} \circ f_{\boldsymbol{\sigma}_j}$ corresponds to the permutation matrix $\boldsymbol{\sigma}_i\boldsymbol{\sigma}_j$, and $f_{\boldsymbol{\sigma}_i} \circ f_{\boldsymbol{\sigma}_j}(v) = f_{\boldsymbol{\sigma}_i}\big(f_{\boldsymbol{\sigma}_j}(v)\big)$. The identity $\mathbf{I}_p$ can be seen as the particular permutation that does not change the order of the elements, and the corresponding bijection, which will be denoted as $f_{\mathbf{I}_p}$, is such that each element is mapped into itself (in other words, $f_{\mathbf{I}_p}(v) = v$).

Theorem 8 *Let $\mathcal{F} = \{\boldsymbol{\sigma}_0 = \mathbf{I}_p, \boldsymbol{\sigma}_1, \cdots, \boldsymbol{\sigma}_{p-1}\}$ be a family of linear transformations, with each $\boldsymbol{\sigma}_i$ being a permutation, and suppose that $\mathcal{F}$ induces the reproducible pseudo-ring $\mathcal{M}_q^{\mathcal{F},1}$ over $\mathbb{F}_q$. Then, the following relation must be satisfied*

$$\boldsymbol{\sigma}_j\boldsymbol{\sigma}_i = \boldsymbol{\sigma}_{f_{\boldsymbol{\sigma}_i}(j)}, \quad \forall i, j \in \mathbb{N}, \quad 0 \leq i \leq p-1, \quad 0 \leq j \leq p-1.$$

Proof. Since $\mathcal{M}_q^{\mathcal{F},1}$ is a pseudo-ring, we know from Theorem 7 that, for every matrix $\mathbf{B} \in \mathcal{M}_q^{\mathcal{F},1}$ and every function $\boldsymbol{\sigma}_i \in \mathcal{F}$, it must be $\boldsymbol{\sigma}_i\mathbf{B} = \mathbf{B}\boldsymbol{\sigma}_i$. In particular, the left-hand term multiplication of $\boldsymbol{\sigma}_i$ by $\mathbf{B}$ corresponds to a row permutation, such that

$$\boldsymbol{\sigma}_i\mathbf{B} = \begin{bmatrix} \mathbf{b}_{f_{\boldsymbol{\sigma}_i}(0)} \\ \mathbf{b}_{f_{\boldsymbol{\sigma}_i}(1)} \\ \vdots \\ \mathbf{b}_{f_{\boldsymbol{\sigma}_i}(p-1)} = \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0\boldsymbol{\sigma}_{f_{\boldsymbol{\sigma}_i}(0)} \\ \mathbf{b}_0\boldsymbol{\sigma}_{f_{\boldsymbol{\sigma}_i}(1)} \\ \vdots \\ \mathbf{b}_0\boldsymbol{\sigma}_{f_{\boldsymbol{\sigma}_i}(p-1)} \end{bmatrix}, \tag{5.1}$$

where $\mathbf{b}_i$ denotes the i-th row of $\mathbf{B}$. The product $\mathbf{B}\boldsymbol{\sigma}_i$ instead defines a column permutation of $\mathbf{B}$, and can be expressed as

$$\mathbf{B}\sigma_i = \begin{bmatrix} \mathbf{b}_0\boldsymbol{\sigma}_0 \\ \mathbf{b}_0\boldsymbol{\sigma}_1 \\ \vdots \\ \mathbf{b}_0\boldsymbol{\sigma}_{p-1} \end{bmatrix} \boldsymbol{\sigma}_i = \begin{bmatrix} \mathbf{b}_0\boldsymbol{\sigma}_0\boldsymbol{\sigma}_i \\ \mathbf{b}_0\boldsymbol{\sigma}_1\boldsymbol{\sigma}_i \\ \vdots \\ \mathbf{b}_0\boldsymbol{\sigma}_{p-1}\boldsymbol{\sigma}_i \end{bmatrix}. \tag{5.2}$$

Putting together equations (5.1) and (5.2), we obtain

$$\boldsymbol{\sigma}_j\boldsymbol{\sigma}_i = \boldsymbol{\sigma}_{f_{\boldsymbol{\sigma}_i}(j)},$$

which must be satisfied for every pair of indexes (i, j). ■

Starting from the result of Theorem 8, we can easily derive some other properties that $\mathcal{F}$ must satisfy.

Corollary 2 *Let $\mathcal{F}$ be a family of permutations satisfying Theorem 8. Then, $\mathcal{F}$ has the following properties*

1. $f_{\boldsymbol{\sigma}_i}(0) = i, \ \ \forall i$;
2. $\forall i \ \ \exists j \ \ s.t. \ \ f_{\boldsymbol{\sigma}_i} \circ f_{\boldsymbol{\sigma}_j} = f_{\mathbf{I}_p}$.

Proof. According to Theorem 8, we have

$$\boldsymbol{\sigma}_{f_{\boldsymbol{\sigma}_i}(0)} = \boldsymbol{\sigma}_0 \boldsymbol{\sigma}_i = \mathbf{I}_p \boldsymbol{\sigma}_i = \boldsymbol{\sigma}_i,$$

which can be satisfied only if $f_{\boldsymbol{\sigma}_i}(0) = i$, and this proves property (a).
Since each $f_{\boldsymbol{\sigma}_i}$ is a bijection of the integers in $[0, p-1]$, we know that, for a fixed value of i, there is a value $j \in [0, p-1]$ such that $f_{\boldsymbol{\sigma}_i}(j) = 0$. Then, we have

$$\boldsymbol{\sigma}_j \boldsymbol{\sigma}_i = \boldsymbol{\sigma}_{f_{\sigma_i}(j)} = \boldsymbol{\sigma}_0 = \mathbf{I}_p.$$

In other words, the bijections corresponding to $f_{\boldsymbol{\sigma}_i}$ and $f_{\boldsymbol{\sigma}_j}$ are one the inverse of the other, and this proves property (b). ■

Corollary 3 *Let $\mathcal{F}$ be a family of permutations satisfying Theorem 8. Then, $\mathcal{M}_q^{\mathcal{F},1}$ is a ring, which we call, by analogy,* reproducible ring *induced by $\mathcal{F}$.*

Proof. Let us show that $\mathcal{M}_q^{\mathcal{F},1}$ contains the multiplicative identity, i.e., the $p \times p$ identity matrix. Because of Corollary 2, $\mathcal{F}$ is formed by $p \times p$ permutations such that $f_{\boldsymbol{\sigma}_i}(0) = i, \forall i$. If we generate the element of $\mathcal{M}_q^{\mathcal{F},1}$ corresponding to the signature $\mathbf{u} = [1, 0, \cdots, 0]$, we easily obtain the $p \times p$ identity matrix $\mathbf{I}_p$. ■

Theorem 9 *Let $\mathcal{F}$ be a family of permutations satisfying Theorem 8. Then, $\mathcal{M}_q^{\mathcal{F},1}$ is a reproducible ring and the invertible elements of $\mathcal{M}_q^{\mathcal{F},1}$ constitute a multiplicative group over $\mathcal{M}_q^{\mathcal{F},1}$.*

Proof. Based on Corollary 3, $\mathcal{M}_q^{\mathcal{F},1}$ is a reproducible ring provided with multiplicative identity. Now, we need to prove that any non-singular matrix in $\mathcal{M}_q^{\mathcal{F},1}$ admits inverse in $\mathcal{M}_q^{\mathcal{F},1}$. Let us consider a matrix $\mathbf{A} \in \mathcal{M}_q^{\mathcal{F},m}$, with signature $\mathbf{a}$, and let $\mathbf{B}$ be its inverse. Since $\mathbf{AB} = \mathbf{I}_p$, we have

$$\mathbf{AB} = \begin{bmatrix} \mathbf{a} \\ \mathbf{a}\boldsymbol{\sigma}_1 \\ \vdots \\ \mathbf{a}\boldsymbol{\sigma}_{p-1} \end{bmatrix} \mathbf{B} = \mathbf{I}_p = \begin{bmatrix} \mathbf{u} \\ \mathbf{u}\boldsymbol{\sigma}_1 \\ \vdots \\ \mathbf{u}\boldsymbol{\sigma}_{p-1} \end{bmatrix},$$

with $\mathbf{u} = [1, 0, \cdots, 0]$ as in Corollary 3. Then we have $\mathbf{a}\boldsymbol{\sigma}_i\mathbf{B} = \mathbf{u}\boldsymbol{\sigma}_i$. For $i = 0$, we have $\mathbf{u} = \mathbf{aB}$. Hence, for whichever value i, we get

$$\mathbf{a}\boldsymbol{\sigma}_i\mathbf{B} = \mathbf{u}\boldsymbol{\sigma}_i = \mathbf{aB}\boldsymbol{\sigma}_i,$$

which can be satisfied for whichever $\mathbf{a}$ only if $\boldsymbol{\sigma}_i$ and $\mathbf{B}$ commute. Because of Theorem 7, this means that $\mathbf{B} \in \mathcal{M}_q^{\mathcal{F},1}$. ■

Sum and multiplication are not the only matrix operations we consider. In Theorem 10 we analyze how transposition acts on the matrices belonging to a reproducible pseudo-ring $\mathcal{M}_q^{\mathcal{F},1}$.

Theorem 10 *Let $\mathcal{M}_q^{\mathcal{F},1}$ be a reproducible pseudo-ring; if*

$$f_{\boldsymbol{\sigma}_j}^{-1}(i) = f_{\boldsymbol{\sigma}_v}^{-1}(0), \;\; v = f_{\boldsymbol{\sigma}_i}^{-1}(j), \;\; \forall i, j \;\; \textit{s.t.} \;\; 0 \leq i \leq p-1, 0 \leq j \leq p-1$$

then $\mathcal{M}_q^{\mathcal{F},1}$ is closed under the transposition operation.

Proof. Let $\mathbf{A} \in \mathcal{M}_q^{\mathcal{F},1}$, with signature $\mathbf{a}$, and denote as $\mathbf{B} = \mathbf{A}^\top$ its transpose. The i-th row of $\mathbf{B}$ corresponds to the i-th column of $\mathbf{A}$. In particular, the i-th column of $\mathbf{A}$ is defined as

$$\begin{bmatrix} a_i \\ a_{f_{\boldsymbol{\sigma}_1}^{-1}(i)} \\ a_{f_{\boldsymbol{\sigma}_2}^{-1}(i)} \\ \vdots \\ a_{f_{\boldsymbol{\sigma}_{p-1}}^{-1}(i)} \end{bmatrix}.$$

Because $\mathbf{B}$ is the transpose of $\mathbf{A}$, the i-th row of $\mathbf{B}$ corresponds to the i-th column of $\mathbf{A}$. Let us denote as $\mathbf{b}_0$ the first row of $\mathbf{B}$, that is

$$\mathbf{b}_0 = [a_0, a_{f_{\boldsymbol{\sigma}_1}^{-1}(0)}, \cdots, a_{f_{\boldsymbol{\sigma}_{p-1}}^{-1}(0)}] = [a_{f_{\boldsymbol{\sigma}_0}^{-1}(0)}, a_{f_{\boldsymbol{\sigma}_1}^{-1}(0)}, \cdots, a_{f_{\boldsymbol{\sigma}_{p-1}}^{-1}(0)}].$$

Let us consider the i-th row of $\mathbf{B}$, and denote it as $\mathbf{b}_i$; if transposition has closure in $\mathcal{M}_q^{\mathcal{F},1}$, then it must be

$$\mathbf{b}_i = [a_i, a_{f_{\boldsymbol{\sigma}_1}^{-1}(i)}, \cdots, a_{f_{\boldsymbol{\sigma}_{p-1}}^{-1}(i)}] = [a_{f_{\boldsymbol{\sigma}_0}^{-1}(i)}, a_{f_{\boldsymbol{\sigma}_1}^{-1}(i)}, \cdots, a_{f_{\boldsymbol{\sigma}_{p-1}}^{-1}(i)}] = \mathbf{b}_0\boldsymbol{\sigma}_i. \quad (5.3)$$

Now suppose that $f_{\boldsymbol{\sigma}_i}(v) = j$; then, the j-th entry of $\mathbf{b}_i$ corresponds to the v-th entry of $\mathbf{b}_0$, that is $a_{f_{\boldsymbol{\sigma}_v}^{-1}(0)}$. In other words, we have $b_{i,j} = a_z$, with

$$z = f_{\boldsymbol{\sigma}_v}^{-1}(0), \;\; v = f_{\boldsymbol{\sigma}_i}^{-1}(j).$$

In order to satisfy eq. (5.3), a_z must be equal to the j-th entry of the i-th column of

$\mathbf{A}$, that is $a_{f_{\boldsymbol{\sigma}_j}^{-1}(i)}$. Then, it must be $f_{\boldsymbol{\sigma}_j}^{-1}(i) = z$, that is

$$f_{\boldsymbol{\sigma}_j}^{-1}(i) = f_{\boldsymbol{\sigma}_v}^{-1}(0), \quad v = f_{\boldsymbol{\sigma}_i}^{-1}(j),$$

which concludes the proof. ■

Depending on the properties stated in the previous theorems, the family $\mathcal{F}$ might induce different algebraic structures over $\mathbb{F}_q^{p\times p}$. In particular, let us consider the case of $\mathcal{F}$ corresponding to $\mathcal{M}_q^{\mathcal{F},1}$ satisfying both Theorems 9 and 10. Let $\mathbf{A}$ be a square matrix whose elements are picked from $\mathcal{M}_q^{\mathcal{F},1}$. By definition, we have $\mathbf{A}^{-1} = \det(\mathbf{A})^{-1} \cdot \mathrm{adj}(\mathbf{A})$, where $\det(\mathbf{A})$ is the determinant of $\mathbf{A}$ and $\mathrm{adj}(\mathbf{A})$ is the adjugate of $\mathbf{A}$. Computing $\det(\mathbf{A})$ involves only sums and multiplications: this means that $\det(\mathbf{A}) \in \mathcal{M}_q^{\mathcal{F},1}$; because of Theorem 9, $\det(\mathbf{A})^{-1} \in \mathcal{M}_q^{\mathcal{F},1}$. Computing $\mathrm{adj}(\mathbf{A})$ involves sums, multiplications and transpositions: because of Theorem 10, we have that the entries of $\mathrm{adj}(\mathbf{A})$ are again elements of $\mathcal{M}_q^{\mathcal{F},1}$. This means that $\mathbf{A}^{-1}$ is a matrix whose elements belong to $\mathcal{M}_q^{\mathcal{F},1}$, and so has the same quasi-reproducible structure of $\mathbf{A}$.

5.2.1 Known examples of reproducible rings

In Section 5.2 we have described some properties that a family of permutations $\mathcal{F}$ must have to guarantee that it induces algebraic structures on $\mathbb{F}_q^{p\times p}$. Well-known cases of such objects, with common use in cryptography, are circulant and dyadic matrices.

Circulant Matrices

As we have seen before, a circulant matrix is a $p \times p$ matrix for which each row is obtained as the cyclic shift of the previous one. In particular, a circulant matrix can be seen as a square reproducible matrix, whose signature corresponds to the first row and the functions $\boldsymbol{\sigma}_i$ defining $\mathcal{F}$ correspond to $\boldsymbol{\pi}^i$, where $\boldsymbol{\pi}$ is the unitary circulant permutation matrix with entries

$$\pi_{l,j} = \begin{cases} 1 & \text{if } l+1 \equiv j \mod p \\ 0 & \text{otherwise} \end{cases}$$

Basically, the bijection representing $\boldsymbol{\pi}$ is defined as

$$f_{\boldsymbol{\pi}}(v) = v + 1 \mod p.$$

It can be easily shown that

$$f_{\boldsymbol{\sigma}_i}(v) = f_{\boldsymbol{\pi}^i}(v) = \underbrace{f_{\boldsymbol{\pi}} \circ f_{\boldsymbol{\pi}} \cdots \circ f_{\boldsymbol{\pi}}}_{i \text{ times}}(v) = v + i \mod p,$$

which leads to $\boldsymbol{\pi}^p = \mathbf{I}_p$ and $\boldsymbol{\pi}^i\boldsymbol{\pi}^j = \boldsymbol{\pi}^{i+j \mod p}$. Since permutation matrices are orthogonal, their inverses correspond to their transposes, and thus $(\boldsymbol{\pi}^i)^\top = \boldsymbol{\pi}^{p-i}$. With these properties, we have

$$\boldsymbol{\sigma}_i\boldsymbol{\sigma}_j = \boldsymbol{\pi}^{i+j \mod p} = \boldsymbol{\sigma}_{i+j \mod p},$$

which is compliant with Theorem 8, since $f_{\boldsymbol{\sigma}_i}(j) = i + j \mod p$. With some simple computations, it can be easily shown that circulant matrices satisfy Theorem 10 and that the multiplication between two circulant matrices is commutative.

Dyadic Matrices

A *dyadic* matrix is a $p \times p$ matrix, with p being a power of 2, whose signature is again its first row. The rows of a dyadic matrix are obtained by permuting the elements of the signature, such that the element at position (i, j) is the one in the signature at position $i \oplus j$, where $\oplus$ denotes the bitwise XOR between i and j. Then, a dyadic matrix can be described in terms of reproducible matrices, for which each function $\boldsymbol{\sigma}_i$ is the dyadic matrix whose signature has all-zero entries, except that at position i. This means that $\boldsymbol{\sigma}_i$ can be described by the following bijection

$$f_{\boldsymbol{\sigma}_i}(v) = v \oplus i \mod p.$$

If we combine two transformations, we obtain

$$f_{\boldsymbol{\sigma}_i} \circ f_{\boldsymbol{\sigma}_j}(v) = (v \oplus j) \oplus i = v \oplus (i \oplus j) = f_{\boldsymbol{\sigma}_{i\oplus j}}(v).$$

Since $f_{\boldsymbol{\sigma}_i}(j) = i \oplus j$, this proves that the family of dyadic matrices is compliant with Theorem 8. It can be straightforwardly proven that dyadic matrices are symmetric (and so, satisfy Theorem 10), and that the multiplication between two dyadic matrices is commutative.

Other reproducible pseudo-rings

Circulant and dyadic matrices are just two particular cases of reproducible pseudo-rings, and can obviously be further generalized by considering signatures that are composed by more than one row. In addition, several more constructions can be obtained. For instance, for every permutation matrix $\boldsymbol{\psi}$ and every reproducible pseudo-ring $\mathcal{M}_q^{\mathcal{F},m}$, induced by a family $\mathcal{F}$, we can obtain a new pseudo-ring as

$$\mathcal{M}_q^{\mathcal{F}',m} = \left\{ \mathbf{M}' \,|\, \mathbf{M}' = \boldsymbol{\psi}\mathbf{M}\boldsymbol{\psi}^\top,\ \ \mathbf{M} \in \mathcal{M}_q^{\mathcal{F},m} \right\}.$$

The corresponding family of transformations is $\mathcal{F}' = \left\{\sigma'_0, \sigma'_1, \cdots, \sigma'_{\frac{p}{m}-1}\right\}$, with $\boldsymbol{\sigma}'_i = \boldsymbol{\sigma}_{f_{\boldsymbol{\psi}}(i)}\boldsymbol{\psi}^\top$. Proving that $\mathcal{F}'$ actually induces a pseudo-ring is quite simple; indeed, for any two matrices $\mathbf{A} = \boldsymbol{\psi}\mathbf{M}_A\boldsymbol{\psi}^\top$ and $\mathbf{B} = \boldsymbol{\psi}\mathbf{M}_B\boldsymbol{\psi}^\top$, with $\mathbf{M}_A, \mathbf{M}_B \in \mathcal{M}_{\mathcal{F},m}$, we have

$$\mathbf{A} + \mathbf{B} = \boldsymbol{\psi}\mathbf{M}_A\boldsymbol{\psi}^\top + \boldsymbol{\psi}M_B\boldsymbol{\psi}^\top = \boldsymbol{\psi}(\mathbf{M}_A + \mathbf{M}_B)\boldsymbol{\psi}^\top,$$

$$\mathbf{AB} = \boldsymbol{\psi}\mathbf{M}_A\boldsymbol{\psi}^\top\boldsymbol{\psi}\mathbf{M}_B\boldsymbol{\psi}^\top = \boldsymbol{\psi}\mathbf{M}_A\mathbf{M}_B\boldsymbol{\psi}^\top,$$

which return matrices belonging to $\mathcal{M}_q^{\mathcal{F}',m}$, since $\mathbf{M}_A + \mathbf{M}_B \in \mathcal{M}_q^{\mathcal{F},m}$ and $\mathbf{M}_A\mathbf{M}_B \in \mathcal{M}_q^{\mathcal{F},m}$. In addition, if multiplication is commutative in $\mathcal{M}_q^{\mathcal{F},m}$, then it will be commutative in $\mathcal{M}_q^{\mathcal{F}',m}$ too. To prove this fact, let us consider two matrices $\mathbf{M}_A, \mathbf{M}_B \in \mathcal{M}_q^{\mathcal{F},m}$, such that $\mathbf{M}_A\mathbf{M}_B = \mathbf{M}_B\mathbf{M}_A$. Then, for $\mathbf{A} = \boldsymbol{\psi}\mathbf{M}_A\boldsymbol{\psi}^\top$ and $\mathbf{B} = \boldsymbol{\psi}\mathbf{M}_B\boldsymbol{\psi}^\top$, we have

$$\begin{aligned}\mathbf{AB} &= \boldsymbol{\psi}\mathbf{M}_A\boldsymbol{\psi}^\top\boldsymbol{\psi}\mathbf{M}_B\boldsymbol{\psi}^\top = \boldsymbol{\psi}\mathbf{M}_A\mathbf{M}_B\boldsymbol{\psi}^\top = \\ &= \boldsymbol{\psi}\mathbf{M}_B\mathbf{M}_A\boldsymbol{\psi}^\top = \boldsymbol{\psi}\mathbf{M}_B\boldsymbol{\psi}^\top\boldsymbol{\psi}\mathbf{M}_A\boldsymbol{\psi}^\top = \mathbf{BA}.\end{aligned}$$

It is easy to prove that, if $\mathcal{M}_q^{\mathcal{F},m}$ is closed under transposition, $\mathcal{M}_q^{\mathcal{F}',m}$ is too.

5.3 A general form for codes in reproducible form

In the previous section we have described the properties that a family of functions $\mathcal{F}$ must have in order to generate reproducible matrices. This opens a wide range of possibilities for obtaining codes with compact representations. In fact, reproducible pseudo-rings allow to design codes that can be described in a very compact manner. Codes of this type are of interest in code-based cryptography, where small public keys are important.

In this section we describe how to design codes with reproducible representations, and the properties that characterize them. In particular we study how to achieve a reproducible representation for the parity-check matrix $\mathbf{H}$ starting from a generator matrix $\mathbf{G}$ in reproducible form. In addition, we provide intuitive methods to obtain random-looking codes in reproducible form, starting from their parity-check matrix. The following theorem states some properties about the parity-check matrix that are sufficient (but not necessary) conditions for having a code with $\mathbf{G}$ and $\mathbf{H}$ in reproducible form.

Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ in reproducible form, with length n, dimension k and codimension $r = n - k$, with a reproducible generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ defined by signature $\mathbf{g}_0 \in F_q^{m\times n}$ and family of transformations $\mathcal{F}$. In particular, we write $\mathcal{F} = \left\{\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1, \cdots, \boldsymbol{\sigma}_{\frac{k}{m}-1}\right\}$, and suppose that $\boldsymbol{\sigma}_0 = \mathbf{I}_n$. The matrix $\mathbf{G}$ can thus be

expressed as

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{\frac{k}{m}-1} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_0\boldsymbol{\sigma}_1 \\ \vdots \\ \mathbf{g}_0\boldsymbol{\sigma}_{\frac{k}{m}-1} \end{bmatrix}.$$

Let $\mathbf{H} \in \mathbb{F}_q^{r\times n}$ be a parity check matrix for $\mathcal{C}$ and s be one of the factors of r; if r is a prime, necessarily $s = 1$. Then, $\mathbf{H}$ can be expressed as

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{\frac{r}{s}-1} \end{bmatrix},$$

where each $\mathbf{h}_i$ is a matrix with dimensions $s \times n$. Since by definition $\mathbf{G}\mathbf{H}^\top = 0_{k\times r}$, it must be

$$\mathbf{g}_i\mathbf{h}_j^\top = \mathbf{g}_0\boldsymbol{\sigma}_i\mathbf{h}_j^\top = 0_{m\times s}, \quad \forall i,j \in \mathbb{N} \text{ s.t. } 0 \le i \le \frac{k}{m}-1, \quad 0 \le j \le \frac{r}{s}-1. \tag{5.4}$$

Let us assume that $\mathbf{g}_0\mathbf{H}^\top = 0_{m\times n}$: as we explain later, in the practical case of a cryptographic scheme, this condition can be easily satisfied. The following theorem considers a particular construction for a reproducible code, and states some properties that its parity-check matrix must satisfy.

Theorem 11 *Let $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ be a reproducible matrix, with signature $\mathbf{g}_0 \in \mathbb{F}_q^{m\times n}$ (hence, m is among the factors of k) and family $\mathcal{F} = \left\{\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1, \cdots, \boldsymbol{\sigma}_{\frac{k}{m}-1}\right\}$. For simplicity, we suppose that $\boldsymbol{\sigma}_0 = \mathbf{I}_n$. Let $\mathbf{H} \in \mathbb{F}_q^{r\times n}$, such that $\mathbf{g}_0\mathbf{H}^\top = \mathbf{0}_{m\times r}$. We denote by $\mathbf{h}_j$ the subset of rows of $\mathbf{H}$ at positions $\{js, js+1, \cdots, (j+1)s-1\}$. If we can define a function $f(x_0, x_1) : [0, \frac{k}{m}-1] \times [0, \frac{r}{s}-1] \subset \mathbb{N}^2 \to [0, \frac{r}{s}-1] \subset \mathbb{N}$, such that*

$$\mathbf{h}_j\boldsymbol{\sigma}_i^\top = \mathbf{h}_{f(i,j)}, \quad \forall i,j \in \mathbb{N}, \quad 0 \le i \le \frac{k}{m}-1, \quad 0 \le j \le \frac{r}{s}-1, \tag{5.5}$$

then $\mathbf{G}$ and $\mathbf{H}^\top$ are orthogonal, i.e. $\mathbf{G}\mathbf{H}^\top = \mathbf{0}_{k\times r}$.

Proof. Since the generator matrix $\mathbf{G}$ is reproducible, with signature $\mathbf{g}_0$, we have

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{\frac{k}{m}-1} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_0\boldsymbol{\sigma}_1 \\ \vdots \\ \mathbf{g}_0\boldsymbol{\sigma}_{\frac{k}{m}-1} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{\frac{r}{s}-1} \end{bmatrix}.$$

In order for $\mathbf{G}$ to be a valid generator matrix, it must be $\mathbf{G}\mathbf{H}^\top = \mathbf{0}_{k\times r}$, that is

$$\mathbf{g}_i\mathbf{h}_j^\top = \mathbf{g}_0\boldsymbol{\sigma}_i\mathbf{h}_j^\top = \mathbf{0}_{m\times s},\ \ \forall i,j\in\mathbb{N}\ \text{ s.t. }\ 0\le i\le \frac{k}{m}-1,\ \ 0\le j\le \frac{r}{s}-1. \quad (5.6)$$

By hypothesis, $\mathbf{g}_0$ is an $m\times n$ matrix such that $\mathbf{g}_0\mathbf{H}^\top = \mathbf{0}_{m\times r}$, which means

$$\mathbf{g}_0\mathbf{h}_j^\top = \mathbf{0}_{m\times s},\ \ \forall j\in\mathbb{N}\ \text{ s.t. }\ 0\le j\le \frac{r}{s}-1. \quad (5.7)$$

Consider now the product $\mathbf{g}_i\mathbf{h}_j^\top = \mathbf{g}_0\boldsymbol{\sigma}_i\mathbf{h}_j^\top$, for $i\ge 1$. If we can define a function $f(x_0,x_1):[0,\frac{k}{m}-1]\times[0,\frac{r}{s}-1]\subset\mathbb{N}^2\to[0,\frac{r}{s}-1]\subset\mathbb{N}$ with the aforementioned property described by (5.5), then for all couples of indexes i,j we have

$$\boldsymbol{\sigma}_i\mathbf{h}_j^\top = \mathbf{h}_{f(i,j)}^\top,$$

and (5.6) is surely satisfied, since

$$\mathbf{g}_i\mathbf{h}_j^\top = \mathbf{g}_0\boldsymbol{\sigma}_i\mathbf{h}_j^\top = \mathbf{g}_0\mathbf{h}_{f(i,j)}^\top = \mathbf{0}_{m\times s},$$

where $\mathbf{g}_0\mathbf{h}_{f(i,j)}^\top = \mathbf{0}_{m\times s}$ because of (5.7). ■

For $\mathbf{G}$ and $\mathbf{H}$ to be, respectively, generator and parity-check matrix of a code $\mathcal{C}$, some conditions have to be verified, given in Corollary 4 below.

Corollary 4 *Let $\mathbf{G}\in\mathbb{F}_q^{k\times n}$ be a reproducible matrix, with signature $\mathbf{g}_0\in\mathbb{F}_q^{m\times n}$ (hence, m is among the factors of k) and family $\mathcal{F}=\left\{\boldsymbol{\sigma}_0,\boldsymbol{\sigma}_1,\cdots,\boldsymbol{\sigma}_{\frac{k}{m}-1}\right\}$. Let $\mathbf{H}\in\mathbb{F}_q^{r\times n}$ be a matrix such that $\mathbf{G}\mathbf{H}^\top=\mathbf{0}_{k\times n}$, and suppose that it satisfies the hypothesis of Theorem 11. For $\mathbf{H}$ and $\mathbf{G}$ to be, respectively, the parity-check and generator matrices of a code $\mathcal{C}$ with length n, dimension k and redundancy r, the following conditions are necessary*

(a) $\mathcal{F}$ contains $\frac{k}{m}$ distinct linear transformations;

(b) $\frac{k}{m}\le\frac{r}{s}$;

(c) for any three integers $i\in[0,\frac{k}{m}-1]$ and $j',j''\in[0,\frac{r}{s}-1]$, with $j'\neq j''$, it must be $f(i,j')\neq f(i,j'')$.

Proof. We want the reproducible $k\times n$ matrix $\mathbf{G}$ to be the generator matrix of a code with dimension k: then, $\mathbf{G}$ must have rank equal to k. If $\mathcal{F}$ contains two transformations $\boldsymbol{\sigma}_i=\boldsymbol{\sigma}_j$, with $i\neq j$, then the rows of $\mathbf{G}$ obtained as $\mathbf{g}_0\boldsymbol{\sigma}_i$ are identical to the ones obtained as $\mathbf{g}_0\boldsymbol{\sigma}_j$. If $\mathbf{G}$ has some identical rows, then its rank cannot be maximum, and this proves condition (a). It is straightforward to show that this condition can also be expressed as follows: there cannot exist three integers $i',i''\in[0,\frac{k}{m}-1]$, with $i'\neq i''$, and $j\in[0,\frac{r}{s}-1]$, such that $f(i',j)=f(i'',j)$. Indeed, if we can

determine such integers, then

$$\mathbf{h}_j \boldsymbol{\sigma}_{i'}^\top = \mathbf{h}_{f(i',j)} = \mathbf{h}_{f(i'',j)} = \mathbf{h}_j \boldsymbol{\sigma}_{i''}^\top,$$

which results in $\boldsymbol{\sigma}_{i'} = \boldsymbol{\sigma}_{i''}$.

We can then easily prove condition (b). Indeed, fix an integer $j \in [0, \frac{r}{s} - 1]$ and consider, for all $i \in [0, \frac{k}{m} - 1]$, all the images $f(i, j)$: because of condition (a), these images must be distinct. However, the dimension of the codomain of $f(i, j)$ is equal to $\frac{r}{s}$: if $\frac{k}{m} > \frac{r}{s}$, then (a) cannot be satisfied. This proves (b).
If $\mathbf{H}$ is the parity-check matrix of a code with redundancy r, then it must have rank equal to r. If we suppose that there exists three integers $i \in [0, \frac{k}{m} - 1]$, $j', j'' \in [0, \frac{r}{s} - 1]$, with $j' \neq j''$, such that $f(i, j') = f(i, j'')$ then, because of Theorem 11, we also have $\mathbf{h}_{j'} \boldsymbol{\sigma}_i^\top = \mathbf{h}_{j''} \boldsymbol{\sigma}_i^\top$, which implies $\mathbf{h}_{j'} = \mathbf{h}_{j''}$. If $\mathbf{H}$ has some identical rows, then its rank must be $< r$, and this proves condition (c). ■

Theorem 11 and Corollary 4 allow obtaining a code in reproducible form in a very simple way. Given a family of transformations $\mathcal{F}$, a matrix $\mathbf{H}$ with the characteristics required by the theorem can be found. Then, for the code $\mathcal{C}$ having $\mathbf{H}$ as parity-check matrix, a variety of reproducible generator matrices can be found. Indeed, let $\mathbf{G}$ be a generator matrix for $\mathcal{C}$: by definition, since $\mathbf{G}\mathbf{H}^\top = \mathbf{0}_{k \times r}$, we know that whichever subset $\mathbf{g}_0$ formed by m rows of $\mathbf{G}$ is such that $\mathbf{g}_0 \mathbf{H}^\top = \mathbf{0}_{m \times r}$. Then, $\mathbf{g}_0$ is a valid signature for a reproducible generator matrix, defined by the family $\mathcal{F}$. On condition that both $\mathbf{H}$ and $\mathbf{G}$ have full rank, then they can be used to represent the code $\mathcal{C}$, with length n, dimension k and redundancy r.

In some cases, a quasi-reproducible code can be seen as a particular case of a reproducible code (and viceversa). Let us consider a code $\mathcal{C}$ with length $n = n_0 p$, dimension $k = p$ and codimension $r = (n_0 - 1)p$, for some integer $n_0 \in \mathbb{N}$. Let us suppose that $\mathbf{G}$ is obtained as a row of n_0 blocks with dimensions $p \times p$, that is

$$\mathbf{G} = [\mathbf{G}_0 | \mathbf{G}_1 | \cdots | \mathbf{G}_{n_0 - 1}]. \tag{5.8}$$

This form of the generator matrix, for instance, is adopted in LEDA and BIKE [20,53]. Suppose that $\mathbf{G}$ in (5.8) is in quasi-reproducible form, i.e., each $\mathbf{G}_i$ is an element of the reproducible pseudo-ring $\mathcal{M}_q^{\mathcal{F}_i, m_i}$. If the signatures have all the same number of rows (that is, $m_i = m$), then such a $\mathbf{G}$ can be characterized as a particular reproducible matrix. Let us write the i-th family of transformations as $\mathcal{F}_i = \left\{ \boldsymbol{\sigma}_0^{(i)}, \boldsymbol{\sigma}_1^{(i)}, \cdots, \boldsymbol{\sigma}_{\frac{p}{m}-1}^{(i)} \right\}$

and define an overall family of transformations $\mathcal{F} = \left\{\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1, \cdots, \boldsymbol{\sigma}_{\frac{p}{m}-1}\right\}$, such that

$$\boldsymbol{\sigma}_i = \begin{bmatrix} \boldsymbol{\sigma}_i^{(0)} & \mathbf{0}_{p\times p} & \mathbf{0}_{p\times p} & \cdots & \mathbf{0}_{p\times p} \\ \mathbf{0}_{p\times p} & \boldsymbol{\sigma}_i^{(1)} & \mathbf{0}_{p\times p} & \cdots & \mathbf{0}_{p\times p} \\ \mathbf{0}_{p\times p} & \mathbf{0}_{p\times p} & \boldsymbol{\sigma}_i^{(2)} & \cdots & \mathbf{0}_{p\times p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{p\times p} & \mathbf{0}_{p\times p} & \mathbf{0}_{p\times p} & \cdots & \boldsymbol{\sigma}_i^{(n_0-1)} \end{bmatrix}. \tag{5.9}$$

Then, it is easy to see that a matrix in the form (5.8) can be described as a reproducible matrix obtained through $\mathcal{F}$ in (5.9), with signature

$$\mathbf{g}_0 = \left[\mathbf{g}_0^{(0)} \middle| \mathbf{g}_0^{(1)} \middle| \cdots \middle| \mathbf{g}_0^{(n_0-1)}\right],$$

where $\mathbf{g}_0^{(i)}$ is the signature of $\mathbf{G}_i$ in Eq, (5.8).

5.3.1 Reproducible codes from Householder matrices

A *Householder matrix* [54] is a matrix that is at the same time orthogonal and symmetric. Let us consider a set of distinct Householder matrices $\boldsymbol{\psi}_0, \cdots, \boldsymbol{\psi}_{v-1}$. We have that, for all $j = 0, \ldots, v-1$, it must be $\boldsymbol{\psi}_j^{-1} = \boldsymbol{\psi}_j^{\top} = \boldsymbol{\psi}_j$. In order to fulfill the conditions of Theorem 11, these matrices must form a commutative group, that is

$$\boldsymbol{\psi}_i \boldsymbol{\psi}_j = \boldsymbol{\psi}_j \boldsymbol{\psi}_i, \ 0 \leq i, j \leq v-1. \tag{5.10}$$

Let us consider two sets containing all the 2^v distinct binary v-tuples, i.e.

$$\left\{\mathbf{a}^{(i)} \middle| \ 0 \leq i \leq 2^v - 1, \ \mathbf{a}^{(i)} \in \mathbb{F}_2^v, \ \text{ s.t. } \ \mathbf{a}^{(i)} \neq \mathbf{a}^{(j)}, \ \ \forall i \neq j\right\},$$
$$\left\{\mathbf{b}^{(i)} \middle| \ 0 \leq i \leq 2^v - 1, \ \mathbf{b}^{(i)} \in \mathbb{F}_2^v, \ \text{ s.t. } \ \mathbf{b}^{(i)} \neq \mathbf{b}^{(j)}, \ \ \forall i \neq j\right\}.$$

For the sake of simplicity, let us fix $\mathbf{a}^{(0)} = \mathbf{0}_{1\times v}$. It is clear that these two sets are identical, except for the order of their elements. We can now define a family of transformations $\mathcal{F}$, containing 2^v linear functions $\boldsymbol{\sigma}_i$, defined as

$$\boldsymbol{\sigma}_i = \prod_{l=0}^{v-1} \boldsymbol{\psi}_l^{a_l^{(i)}},$$

where $a_l^{(i)}$ is the l-th entry of $\mathbf{a}^{(i)}$. Since we are considering Householder matrices with the property (5.10), it is easy to verify that $\boldsymbol{\sigma}_i^2 = \mathbf{I}_n$, and it follows that each function is an involution.

The family $\mathcal{F}$ can be used to define a reproducible generator matrix $\mathbf{G}$ of a code $\mathcal{C}$; a parity-check matrix for $\mathcal{C}$ can be the reproducible matrix $\mathbf{H}$, with signature $\mathbf{h}_0 \in$

$\mathbb{F}_q^{s\times n}$, whose rows are obtained as

$$\mathbf{h}_j = \mathbf{h}_0 \left(\prod_{l=0}^{v-1} \boldsymbol{\psi}_l^{b_l^{(j)}} \right)^\top .$$

If $\mathbf{H}$ has full rank, the corresponding code has redundancy $r = s2^v$, and

$$\mathbf{h}_j \boldsymbol{\sigma}_i^\top = \mathbf{h}_j \left(\prod_{l=0}^{v-1} \boldsymbol{\psi}_l^{a_l^{(i)}} \right)^\top = \mathbf{h}_0 \left(\prod_{l=0}^{v-1} \boldsymbol{\psi}_l^{b_l^{(j)}} \right)^\top \left(\prod_{l=0}^{v-1} \boldsymbol{\psi}_l^{a_l^{(i)}} \right)^\top =$$
$$= \mathbf{h}_0 \left(\prod_{l=0}^{v-1} \boldsymbol{\psi}_l^{a_l^{(j)} \oplus b_l^{(i)}} \right)^\top = \mathbf{h}_{f(i,j)},$$

where $\oplus$ denotes the modulo 2 sum and

$$f(i,j) = u, \;\; \text{s.t.} \;\; \mathbf{b}^{(u)} = \mathbf{a}^{(i)} \oplus \mathbf{b}^{(j)}.$$

It is straightforward to show that such a function satisfies the properties required by Theorem 11 and Corollary 4. The corresponding code has length n, dimension $k = m2^v$ and redundancy $r = s2^v$, thus the code rate corresponds to $\frac{m}{m+s}$. In addition, we point out that it might be possible to tune the code parameters, by selecting only proper subsets of all the binary v-tuples, in order to form the rows of both $\mathbf{G}$ and $\mathbf{H}$.

5.3.2 Reproducible codes from powers of a single function

In this section we present another construction of reproducible codes satisfying Theorem 11. Let us consider an $n \times n$ matrix $\boldsymbol{\pi}$ such that $\boldsymbol{\pi}^b = \mathbf{I}_n$, for some integer b. Let v be a divisor of b; obviously, if b is a prime, then $v = 1$. Then, we can use $\boldsymbol{\pi}$ to build a family $\mathcal{F}$ of $\frac{k}{m} \leq \frac{b}{v}$ linear transformations, where k is the desired code dimension and m is the number of rows in a signature. Indeed, the functions in $\mathcal{F}$ can be defined as $\boldsymbol{\sigma}_i = \boldsymbol{\pi}^{vz_i}$, where the values z_i are distinct integers $\leq \frac{b}{v}$. For simplicity, we assume $z_0 = 0$, i.e. $\boldsymbol{\sigma}_0 = \mathbf{I}_n$. Then, given a $m \times n$ signature $\mathbf{g}_0$, we can use the family $\mathcal{F}$ to obtain a generator matrix $\mathbf{G}$ for a code $\mathcal{C}$ as

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{\frac{k}{m}-1} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_0\boldsymbol{\pi}^{vz_1} \\ \mathbf{g}_0\boldsymbol{\pi}^{vz_2} \\ \vdots \\ \mathbf{g}_0\boldsymbol{\pi}^{vz_{\frac{k}{m}-1}} \end{bmatrix}.$$

A parity-check matrix for $\mathcal{C}$ can be chosen in reproducible form, by taking an $s \times n$ matrix $\mathbf{h}_0$, and use it to generate the parity-check matrix $\mathbf{H}$ as

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_{\frac{b}{v}-1} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_0(\boldsymbol{\pi}^{b-v})^\top \\ \mathbf{h}_0(\boldsymbol{\pi}^{b-2v})^\top \\ \vdots \\ \mathbf{h}_0(\boldsymbol{\pi}^{v})^\top \end{bmatrix}.$$

When $\mathbf{H}$ has full rank, then $\mathcal{C}$ has redundancy $r = s\frac{b}{v}$; the code dimension and redundancy must be linked to the code length according to $k + s\frac{b}{v} = n$.

It is quite easy to show that such a parity check matrix is compliant with Theorem 11. In fact, we have

$$\mathbf{h}_j \boldsymbol{\sigma}_i^\top = \mathbf{h}_0(\boldsymbol{\pi}^{b-jv})^\top(\boldsymbol{\pi}^{vz_i})^\top = \mathbf{h}_0 \left[\boldsymbol{\pi}^{b+(z_i-j)v}\right]^\top.$$

If $z_i \geq j$, we have

$$\begin{aligned}\left[\boldsymbol{\pi}^{b+(z_i-j)v}\right]^\top &= \left[\boldsymbol{\pi}^{2b-b+(z_i-j)v}\right]^\top = \left[\boldsymbol{\pi}^{b-(\frac{b}{v}+j-z_i)v}\right]^\top \left[\boldsymbol{\pi}^{b}\right]^\top = \\ &= \left[\boldsymbol{\pi}^{b-(\frac{b}{v}+j-z_i)v}\right]^\top = \left[\boldsymbol{\pi}^{b-(j-z_i \mod \frac{b}{v})v}\right]^\top.\end{aligned}$$

In the case of $z_i < j$, we can write

$$\left[\boldsymbol{\pi}^{b+(z_i-j)v}\right]^\top = \left[\boldsymbol{\pi}^{b-(j-z_i)v}\right]^\top \left[\boldsymbol{\pi}^{b-(j-z_i \mod \frac{b}{v})v}\right]^\top.$$

Thus, we have proven that

$$\mathbf{h}_j \boldsymbol{\sigma}_i^\top = \mathbf{h}_0 \left[\boldsymbol{\pi}^{b-(j-z_i \mod \frac{b}{v})v}\right]^\top = \mathbf{h}_{(j-z_i \mod \frac{b}{v})},$$

such that the function $f(x_0, x_1)$ required by Theorem 11 is defined as

$$f(x_0, x_1) = x_1 - z_{x_0} \mod \frac{b}{v}.$$

For instance, a simple construction can be obtained by choosing $m = s = 1$ and $k = r = n/2$: the matrices $\mathbf{G}$ and $\mathbf{H}$ are two reproducible matrices, with signatures that are row vectors of length n, and are characterized by the same number of rows (thus, $\mathcal{C}$ has rate $1/2$).

For what concerns property (b), we can consider the following equivalence

$$x_0 - x_1' \equiv x_0 - x_1'' \mod \frac{r}{s},$$

which turns into

$$x_1'' - x_1' \equiv 0 \mod \frac{r}{s}.$$

Then, it is clear that it must be $x', x'' < \frac{r}{s}$: however, this condition is quite straightforward, since j denotes the row index of the matrix blocks in $\mathbf{H}$. In the same way, when considering the index of the transformation $\boldsymbol{\sigma}_i$, we have

$$x_0' - x_1 \equiv x_0'' - x_1 \mod \frac{r}{s},$$

which turns into

$$x_0' - x_0'' \equiv 0 \mod \frac{r}{s}.$$

Again, in order to guarantee that the previous equivalence has no solution, it must be $x_0', x_0'' < \frac{r}{s}$. This basically means that we must have $k \leq m\frac{r}{s}$.

5.4 Code-based schemes from quasi-reproducible codes

The algebraic structures we have introduced in the previous sections can be used to generate key pairs in code-based cryptosystems. For instance, let us consider a parity-check matrix $\mathbf{H}$ made of $r_0 \times n_0$ matrices belonging to a pseudo-ring $\mathcal{M}_q^{\mathcal{F},m}$. In order to use $\mathbf{H}$ as the private key of a sparse-matrix code-based instance of the Niederreiter cryptosystem, we must guarantee that $\mathbf{H}$ is sufficiently sparse: this property can be easily achieved by choosing a family $\mathcal{F}$ of sparse matrices $\boldsymbol{\sigma}_i$, which guarantee that a matrix defined by a sparse signature will be sparse as well. In such a case, we can obtain the public key as $\mathbf{H}' = \mathbf{SH}$, where $\mathbf{S}$ is a random dense matrix, whose elements are picked over $\mathcal{M}_q^{\mathcal{F},m}$. Because of Theorem 7, the entries of $\mathbf{H}'$ belong to $\mathcal{M}_q^{\mathcal{F},m}$, thus they maintain the same reproducible structure defined by $\mathcal{F}$.

If $m = 1$ and $\mathcal{F}$ is a family of permutations satisfying Theorem 8, then $\mathcal{M}_q^{\mathcal{F},1}$ is actually a ring (see Corollary 3). Then, the secret key can be chosen as $\mathbf{H} = [\mathbf{H}_0, \mathbf{H}_1, \cdots, \mathbf{H}_{n_0-1}]$, with $\mathbf{H}_i \in \mathcal{M}_q^{\mathcal{F},1}$, while the public key can corresponds to the systematic form of $\mathbf{H}$, that is $\mathbf{H}' = \mathbf{H}_0^{-1}\mathbf{H}$. Indeed, because of Theorem 9, we have $\mathbf{H}_0^{-1} \in \mathcal{M}_q^{\mathcal{F},1}$, and so $\mathbf{H}'$ is a matrix constituted of blocks over $\mathcal{M}_q^{\mathcal{F},1}$.

Suppose we have a family $\mathcal{F}$ satisfying Theorem 10, for which multiplication in $\mathcal{M}_q^{\mathcal{F},1}$ is commutative (see Section 5.2.1 for some examples). Then, we can use the reproducible pseudo-ring induced by $\mathcal{F}$ to obtain key pairs for a McEliece cryptosystem. For instance, we can choose $\mathbf{H} = [\mathbf{H}_0, \mathbf{H}_1]$, with $\mathbf{H}_i \in \mathcal{M}_q^{\mathcal{F},1}$, as the secret parity-check matrix, and obtain a generator matrix as $\mathbf{G} = \mathbf{S}[\mathbf{H}_1^\top, -\mathbf{H}_0^\top]$, with $\mathbf{S} \in \mathcal{M}_q^{\mathcal{F},1}$. The matrices $\mathbf{H}$ and $\mathbf{G}$ can be used as the private and public key, respectively, for a McEliece cryptosystem.

When both Theorems 9 and 10 are satisfied, we can obtain a generator matrix in systematic form, which maintains a quasi-reproducible structure. In fact, starting from a $r \times n$ parity-check matrix $\mathbf{H}$, where the elements are picked randomly from $\mathcal{M}_q^{\mathcal{F},1}$, we can use the corresponding parity-check matrix in systematic form as the public key for a Niederreiter cryptosystem instance. In the same way, we can compute the systematic generator matrix, and use it as the public key in a McEliece cryptosystem instance.

The idea of using codes that are completely reproducible, and not formed by reproducible pseudo-rings, opens up for the possibility of a whole new way of generating key pairs in the McEliece cryptosystem. Indeed, once we have generated a sparse parity-check matrix $\mathbf{H}$, we can use it as the secret key. Then, a possible public key can be obtained by taking a bunch of linearly independent codewords, and using them as the signature of the public generator matrix. If such codewords correspond to rows of the generator matrix in systematic form, then we obviously obtain another significant reduction in the public key size, since there is no need for publishing the first k bits of each one of the selected codewords.

It is clear that having the public code described by a matrix with some reproducible structure leads to a very significant reduction in the public-key size. Indeed, once the structure of the matrix is fixed by the protocol (i.e. dimensions, family $\mathcal{F}$), the whole public-key can be efficiently represented using just the signatures of each building block.

5.5 Defeating DOOM

In [55], Sendrier introduced Decoding One Out of Many (DOOM), a variant of MLDP, in which multiple decoding instances are considered: the problem asks to find the solution of (at least) one instance among the given ones. With some abuse of notation, normally DOOM also corresponds to the name of the algorithm, introduced in the same work [55], which allows taking into account the presence of multiple instances to speed up the execution of ISD. In this section we discuss about the applicability of DOOM to our more general context of reproducible codes.

When ISD is used to perform a decoding attack, the gain obtained from DOOM can be explained as follows. Consider the public parity-check matrix $\mathbf{H}'$ and a set of N different syndromes $\mathcal{S} = \left\{\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \cdots, \mathbf{s}^{(N-1)}\right\}$ to be decoded. Suppose that, $\forall \mathbf{e}^{(i)}$ such that $\mathbf{H}'{\mathbf{e}^{(i)}}^\top = \mathbf{s}^{(i)}$, there exists a bijective function that allows to obtain $\mathbf{e}^{(i)}$ from $\mathbf{e}^{(0)}$ and vice-versa. We denote such a function by ψ, so that $\mathbf{e}^{(i)} = \psi(\mathbf{e}^{(0)})$ and $\mathbf{e}^{(0)} = \psi^{-1}(\mathbf{e}^{(i)})$. Then each pair $\left\{\mathbf{s}^{(i)}, \mathbf{H}'\right\}$ can be considered as the input of an ISD instance aimed at finding $\mathbf{e}^{(0)}$ with weight $\leq w$ such that $\mathbf{H}'\psi(\mathbf{e}^{(0)})^\top = \mathbf{H}'{\mathbf{e}^{(i)}}^\top = \mathbf{s}^{(i)}$. According to DOOM, we consider N_i independent calls to an ISD algorithm: as

soon as one of these runs successfully ends, the whole algorithm ends since $\mathbf{e}^{(0)}$ has been found. The corresponding gain is equal to $|\mathcal{S}| / \sqrt{N_i} = N/\sqrt{N_i}$, which becomes $\sqrt{N}$ when $N_i = N$. Obviously, exploiting DOOM is beneficial when the N_i decoding instances have comparable complexity. This occurs on condition that $\mathbf{e}^{(i)} = \psi(\mathbf{e}^{(0)})$ has the same Hamming weight as $\mathbf{e}^{(0)}$, or almost the same.

So, the rationale of exploiting DOOM for a decoding attack is to intercept one ciphertext and then try to obtain other valid ciphertexts from it, corresponding to transformed versions of the same error vector. Let us consider the case in which the opponent intercepts a ciphertext corresponding to an initial syndrome $\mathbf{s}^{(0)}$, and wants to recover the vector $\mathbf{e}^{(0)}$ used during encryption. Then, in order to apply DOOM, the opponent must produce other syndromes corresponding to as many error vectors being deterministic functions of $\mathbf{e}^{(0)}$. In other words, suppose that ISD returns the solution $\mathbf{e}^{(i)}$ for $\mathbf{s}^{(i)}$, then it must be $\mathbf{e}^{(i)} = \mathbf{A}\mathbf{e}^{(0)}$, with $\mathbf{A}$ being a full-rank matrix. For instance, in the QC case, the opponent can obtain a set of p syndromes $\mathcal{S}$ just by cyclically shifting the initial syndrome $\mathbf{s}^{(0)}$ and the corresponding error vector $\mathbf{e}^{(0)}$.

In general terms, the applicability of DOOM can be modeled as follows. Starting from a syndrome $\mathbf{s}^{(0)} = \mathbf{H}'\mathbf{e}^{(0)T}$, we want to determine a transformation $\mathbf{\Phi}$ of the syndrome that corresponds to a transformation $\mathbf{\Psi}$ of the error vector, that is

$$\mathbf{\Phi}\mathbf{s}^{(0)} = \mathbf{\Phi}\mathbf{H}'\mathbf{e}^{(0)T} = \mathbf{H}'\left(\mathbf{e}^{(0)}\mathbf{\Psi}\right)^{\top} = \mathbf{H}'\mathbf{\Psi}^{\top}\mathbf{e}^{(0)T}, \tag{5.11}$$

where $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are two matrices over $\mathbb{F}_q$, with size $r \times r$ and $n \times n$, respectively. The previous equation must be satisfied for every vector $\mathbf{e}^{(0)}$; this can happen only if

$$\exists\ \mathbf{\Phi} \in \mathbb{F}_q^{r\times r},\ \mathbf{\Psi}\mathbb{F}_q^{n\times n} \ \text{ s.t. } \ \mathbf{\Phi}\mathbf{H}' = \mathbf{H}'\mathbf{\Psi}^{\top}.$$

For the general class of reproducible codes, the applicability of DOOM must be carefully analyzed. For instance, consider a code obtained with the procedure described in Section 5.3.2, using a family of functions $\mathcal{F}$ consisting of powers of a single function. If this function is a permutation, due to Theorem 11, we have that $\mathbf{H}\boldsymbol{\sigma}_i$ with $\boldsymbol{\sigma}_i \in \mathcal{F}$ always results in a permutation of the rows of $\mathbf{H}$. So, the opponent can build the set $\mathcal{S}$, which is used as input for the DOOM algorithm, by multiplying the initial syndrome by the matrices $\boldsymbol{\sigma}_i$.

However, the case in which $\boldsymbol{\pi}$ is not a permutation matrix is of interest. We consider

a parity-check matrix satisfying as in Section 5.3.2, obtained as

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1\boldsymbol{\pi} \\ \mathbf{h}_2\boldsymbol{\pi}^2 \\ \vdots \\ \mathbf{h}_{r-1}\boldsymbol{\pi}^{r-1} \end{bmatrix},$$

where $\boldsymbol{\pi}^r = \mathbf{I}_n$. In this case, the opponent is still able to produce a set $\mathcal{S}$, since equation (5.11) can be satisfied by choosing $\boldsymbol{\Psi} = \boldsymbol{\pi}^i$; the corresponding reordering of the rows of $\mathbf{H}$ is a cyclic shift by i positions. However, it results that $\mathbf{e}^{(i)} = \mathbf{e}^{(0)}\boldsymbol{\pi}^i$. If $\boldsymbol{\pi}$ is not a permutation matrix, then generally $\mathbf{e}^{(i)}$ has a weight different from that if $\mathbf{e}^{(0)}$. If we consider $\boldsymbol{\pi}$, and its powers, as random matrices over $\mathbb{F}_q$, then the expected weight of $\mathbf{e}^{(i)}$, for all i, is $n\frac{q-1}{q}$. Furthermore, it can be shown that the probability that $\mathbf{e}^{(i)}$ has weight comparable with t is negligible.

According to [56], we can approximate the time complexity for solving ISD on $\mathbf{s}^{(0)}$ as 2^{ct}, where $c = -\log_2(1 - \frac{k}{n})$ and t is the Hamming weight of $\mathbf{s}^{(0)}$. Since the syndromes $\mathbf{s}^{(i)}$, $i \geq 1$, are associated to error vectors that have weight γt, applying ISD on them requires a time-complexity that is significantly larger than 2^{ct}. Then, there is no gain in considering this set of multiple instances, since the additional instances produced by the opponent are associated to an ISD complexity that is significantly larger than that of the original instance.

Chapter 6

LEDAcrypt

In this chapter we describe LEDA, which is a submission to the NIST competition for the standardization of post-quantum primitives. LEDA corresponds to the merge of LEDAkem and LEDApkc, two proposals to the first competition round; at the current state of the competition, LEDA is among the candidates that have been successfully admitted to the secoud round of the competition. The codes employed in LEDA are QC-LDPC codes, which combine the error correction capability of LDPC codes with the compactness and the algorithmic efficiency provided by the QC structure.

Historically, the idea of using LDPC codes in crypto dates back to 2000, and is due to Monico, Rosenthal and Shokrollahi [10], that considered adoption of a sparse parity-check matrix as the public key, in order to exploit sparsity to reach a compact representation (by publishing only the positions of set entries). However, as stated by the authors themselves, this solution is not secure, since the sparsity of the public key may be used to recover the secret key. Furthermore, it must be considered that the dual of an LDPC code has low-weight codewords that, with overwhelming probability, correspond to the rows of its sparse parity-check matrix. In particular, as we describe in this chapter, an ISD algorithm may be used to search for these codewords which, when found, reveal rows of the secret key (which corresponds to the sparse parity-check matrix representing the LDPC code). Remember that the complexity of ISD grows exponentially with the weight of the searched vector: thus, such attacks can easily be avoided by increasing the secret parity-check matrix density. However, to maintain the low density property of the secret key, the code length must be increased as well: if the code has no geometrical structure, this procedure will end up in having public keys of unpractical size.

To avoid this issue, Baldi and Chiaraluce [11–13] have proposed to i) consider QC codes, and ii) use a transformation matrix which hides the structure of the secret code into the public one by increasing the density of low weight codewords in its dual. The QC structure leads to very compact keys and additionally provides the possibility of reaching very low algorithmic complexities. Differently from the case of algebraic codes, LDPC codes do not suffer from critical attacks arising from the additional geometrical structure. Indeed, the only known weaknesses due to the geometrical

structure can either be easily avoided, or do not represent an issue at all (de facto, providing a very limited improvement in known cryptanalysis techniques).

A first attack based on the QC structure has appeared in [57]; such a procedure exploits a particular automorphism group which exists when the circulant size is a number whose factorization contains powers of 2, and essentially provides a polynomial speed up to ISD algorithms. In any case, to counter these attacks, it is enough to choose an odd circulant size or, to be more conservative, to choose it as a prime, to avoid any possible attack based on factorization. Furthermore, it is known that ISD can receive a speed-up due to the QC structure: this technique, which we have already described in Chapter 5, is known as DOOM [55] and yields a polynomial advantage in ISD. In any case, this advantage is non critical and can be countered by increasing the code density (and the number of errors) by a small number.

There are however some other potential issues that appear when QC-LDPC codes are used. First of all, differently from standard coding problems such as SDP and MLDP, there is no NP-completeness proof on their QC versions. In other words, security of these schemes is based on the assumption that the QC structure does not change the hardness of the problem. It must be said that, despite more than 10 years of cryptanalysis, no efficient attack has been found, apart from DOOM: thus, this assumption seems very reliable. The second problem, which at the moment is the true Achille's heel of these schemes, is represented by the decoding procedure. Indeed, as we have described in Chapter 3, efficient decoders for LDPC codes are intrinsically probabilistic. The operations that are performed depend on the relation between the actual syndrome and the parity-check matrix; furthermore, these decoders are characterized by a usually non negligible decoding failure probability. An adversary may then mount *statistical attacks*, which are based on observing information such as the decryption duration or events of decoding failure. In a CCA2 attack model, these procedures may lead to the full recovery of the secret key. As we describe in this chapter, these attacks can be avoided at the cost of i) constant time and power implementations, and ii) code lengths able to guarantee negligible DFR values.

In this section we give a complete description of the LEDA cryptographic suite, by presenting its design rationale. We describe the Q-decoder, a variant of the BF decoder which adapts to the specific code structure employed in LEDA. We describe the whole range of attack avenues against LEDA, describing both decoding and key recovery attacks. We focus on statistical attacks based on decoding failures, and show that their applicability is not due to the QC structure, but mostly depend on the innate LDPC properties. Finally, we describe a technique to bound the decoding failure probability, and use it to derive parameters reaching different security levels.

6.1 The secret key in LEDA

We will use $\mathcal{R}$ to denote the ring of binary circulant matrices of size p. In a circulant matrix all rows and columns have the same weight: thus, with some abuse of notation, we will use circulant weight to denote the weight of any row or column in a circulant. We can easily define an isomorphism ψ between $\mathcal{R}$ and the polynomial ring $\mathbb{F}_2[x]/(x^p+1)$: indeed, we can write $\psi(\mathbf{A}) = a(x)$, where $a(x)$ is the polynomial whose coefficients correspond to the first row of $\mathbf{A}$. For any three circulants $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, the following properties then hold:

i) if $\mathbf{A}+\mathbf{B}=\mathbf{C}$, then $\psi(\mathbf{A})+\psi(\mathbf{B})=\psi(\mathbf{C})$;

ii) if $\mathbf{AB}=\mathbf{C}$, then $\psi(\mathbf{A})\psi(\mathbf{B})=\psi(\mathbf{C})$;

iii) if $\mathbf{AB}=\mathbf{I}$, then $\psi(\mathbf{A})\psi(\mathbf{B})=1$.

For a polynomial $a(x)$, we define its Hamming weight as the weight of $\psi^{-1}\big(a(x)\big)$, that is, as the number of its set entries. We recall the following theorem, which is employed in both BIKE [53] and LEDAcrypt [20] to guarantee efficient sampling of invertible elements from $\mathcal{R}$.

Theorem 12 *Let p be a prime number such that $\mathtt{ord}_p(2) = p-1$. Let $g(x)$ be a binary polynomial in $\mathbb{F}_2[x]/\langle x^p+1\rangle$, with degree >0. Then, $g(x)$ has a multiplicative inverse in $\mathbb{F}_2[x]/(x^p+1)$ if and only if it contains an odd number of terms and $g(x)\neq\Phi(x)$, with $\Phi(x)=x^{p-1}+x^{p-2}+\cdots+x+1$.*

Because of the isomorphism between circulants and polynomial, the previous theorem (of which, for the sake of brevity, we omit the proof) is useful to sample invertible elements in $\mathcal{R}$. As we show in the following, this theorem will be crucial to guarantee an efficient key generation in LEDA cryptosystems.

In LEDAcrypt, the secret key is parameterized by a small integer n_0, the circulant size p, the column weight v of the generated parity-check matrix and another integer m, whose meaning will be clarified in the following. In particular, the secret key is constituted by two matrices:

i) the parity-check matrix of a QC-LDPC code $\mathcal{C}$, in the form

$$\mathbf{H} = [\mathbf{H}_0, \mathbf{H}_1, \cdots, \mathbf{H}_{n_0-1}] \in \mathcal{R}^{n_0},$$

where each $\mathbf{H}_i$ has weight $d_v \ll p$. Then, the secret code is a $(n_0 d_v, d_v)$-regular LDPC code;

ii) a transformation matrix $\mathbf{Q} \in \mathcal{R}^{n_0 \times n_0}$, that is

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{0,0} & \mathbf{Q}_{0,1} & \cdots & \mathbf{Q}_{0,n_0-1} \\ \mathbf{Q}_{1,0} & \mathbf{Q}_{1,1} & \cdots & \mathbf{Q}_{1,n_0-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{n_0-1,0} & \mathbf{Q}_{n_0-1,1} & \cdots & \mathbf{Q}_{n_0-1,n_0-1} \end{bmatrix} \tag{6.1}$$

where each row and column has constant weight $m \ll n_0 p$.

Starting from the secret key, we obtain a parity-check matrix of the public code $\mathcal{C}'$ as

$$\mathbf{L} = \mathbf{H}\mathbf{Q} = [\mathbf{L}_0, \mathbf{L}_1, \cdots, \mathbf{L}_{n_0-1}] \in \mathcal{R}^{n_0}. \tag{6.2}$$

Then, depending on which representation of $\mathcal{C}'$ is used as the public key, we may build a McEliece type scheme or a Niederreiter type one.

To guarantee full rank of $\mathbf{L}$, we consider the following Theorem, which poses some constraints on the scheme parameters.

Theorem 13 *Let $p > 2$ be a prime such that $ord_p(2) = p - 1$ and $\mathbf{Q} \in \mathcal{R}^{n_0 \times n_0}$ as in Eq.* (6.1)*. Let $\mathbf{W_Q} \in \mathbb{N}^{n_0 \times n_0}$ such that its element in position (i, j) is equal to the weight of $\mathbf{Q}_{i,j}$. Let $\mathbf{\Pi}(\mathbf{W_Q})$ be the permanent of $\mathbf{W_Q}$. Then, if $\mathbf{\Pi}(\mathbf{W_Q})$ is odd and $\mathbf{\Pi}(\mathbf{W_Q})$, then $\mathbf{Q}$ is non singular.*

Proof. Since each block $\mathbf{Q}_{ij}$ is isomorphic to a polynomial $q_{ij}(x) \in \mathbb{F}_2[x]/(x^p + 1)$, the determinant of the matrix Q is represented as an element of $\mathbb{F}_2[x]/(x^p + 1)$, too, which we denote as $d(x)$. If the inverse of $d(x)$ exists, then $\mathbf{Q}$ is non singular. According to Theorem 12, showing that $d(x)$ has odd weight and $d(x) \neq \Phi(x) = x^{p-1} + x^{p-2} + \cdots + 1$ is enough to guarantee that it is invertible. In general, when we are considering two polynomials $a(x)$ and $b(x)$, with $\mathrm{wt}\,(a(x)) = w_a$ and $\mathrm{wt}\,(b(x)) = w_b$, the following statements hold:

1. $\mathrm{wt}\,(a(x)b(x)) = w_a w_b - 2c_1$, where c_1 is the number of cancellations of pairs of monomials with the same exponent resulting from multiplication;

2. $\mathrm{wt}\,(a(x) + b(x)) = w_a + w_b - 2c_2$, where c_2 is the number of cancellations due to monomials with the same exponent appearing in both polynomials.

The determinant $d(x)$ is obtained through multiplications and sums of the elements $q_{ij}(x)$ and, in case of no cancellations, has weight equal to $\mathbf{\Pi}(\mathbf{W_Q})$. If some cancellations occur, considering statements 1) and 2) above, we have that $\mathrm{wt}\,(d(x)) = \mathbf{\Pi}(\mathbf{W_Q}) - 2c$, where c is the overall number of cancellations. So, even when cancellations occur, $d(x)$ has odd weight only if $\mathbf{\Pi}(\mathbf{W_Q})$ is odd. In addition, the condition $\mathbf{\Pi}(\mathbf{W_Q}) < p$ guarantees that $d(x) \neq \Phi(x)$, since $\mathrm{wt}\,(\Phi(x)) = p$. ■

In particular, the matrix $\mathbf{Q}$ in LEDAcrypt is chosen such that the distribution of its weights is such that the matrix $\mathbf{W_Q}$ is circulant. When d_v is odd, this guarantees that $\mathbf{L}$ has full rank and that each one of its circulant blocks is non singular.

We point out that a particular case of the framework we describe in this chapter is that proposed by Misozscky et al. in [14], that is an ancestor of the BIKE submission to the NIST competition, in which $\mathbf{Q} = \mathbf{I}$ (and, thus, $m = 1$). In this case, the code $\mathcal{C}$ is typically referred to as a Moderate-Density Parity-Check (MDPC) code and $\mathcal{C}' = \mathcal{C}$. Apart from this slight difference and a different nomenclature, there is really no meaningful distinction in the code properties: MDPC codes can be decoded with the same algorithm that we use for LDPC codes and share the same pros and cons.

When $m > 1$, the knowledge of the matrix $\mathbf{Q}$ can be exploited to improve the decoding process, through a decoding technique which we define as *Q-decoder*. When $m = 1$, the Q-decoder collapses to the classical BF-decoder.

We now proceed in describing the two algorithms which compose the LEDAcrypt suites.

6.1.1 LEDAkem

LEDAkem is built upon the Niederreiter framework; the public key is the systematic parity-check matrix for $\mathcal{C}'$, that is

$$\mathbf{L}' = \mathbf{L}_{n_0-1}^{-1}\mathbf{L} = [\mathbf{M}, \mathbf{I}_p],$$

where $\mathbf{M} \in \mathcal{R}^{n_0-1}$.
Given a weight-t vector $\mathbf{e} \in \mathbb{F}_2^n$, encryption is performed as

$$\mathbf{s} = \mathbf{L}\mathbf{e}^\top.$$

To decrypt, one first computes

$$\begin{aligned}
\mathbf{s} &= \mathbf{L}_{n_0-1}\mathbf{s} \\
&= \mathbf{L}_{n_0-1}\mathbf{L}_{n_0-1}^{-1}\mathbf{L}\mathbf{e}^\top \\
&= \mathbf{L}\mathbf{e}^\top \\
&= \mathbf{H}\mathbf{Q}\mathbf{e}^\top \\
&= \mathbf{H}(\mathbf{e}\mathbf{Q}^\top)^\top \\
&= \mathbf{H}\tilde{\mathbf{e}}^\top,
\end{aligned}$$

where $\tilde{\mathbf{e}} = \mathbf{e}\mathbf{Q}^\top$ is called the *expanded error vector* and has weight $\leq mt$. Then, the Q-decoder is used to recover $\mathbf{e}$.

6.1.2 LEDApkc

LEDApkc is built upon the McEliece framework; the public key is the systematic generator matrix for $\mathcal{C}'$, that is

$$\mathbf{G} = \left[\begin{array}{c|c} \mathbf{I}_{(n_0-1)p} & \begin{array}{c} (\mathbf{L}_{n_0-1}^{-1}\mathbf{L}_0^\top)^\top \\ (\mathbf{L}_{n_0-1}^{-1}\mathbf{L}_1^\top)^\top \\ \vdots \\ (\mathbf{L}_{n_0-1}^{-1}\mathbf{L}_{n_0-2}^\top)^\top \end{array} \end{array}\right].$$

The ciphertext is obtained as

$$\mathbf{c} = \mathbf{m}\mathbf{G} + \mathbf{e},$$

where $\mathrm{wt}(\mathbf{e}) = t$.
Decryption starts with the computation of

$$\begin{aligned} \mathbf{s} &= \mathbf{L}\mathbf{c}^\top \\ &= \mathbf{H}\mathbf{Q}\mathbf{e}^\top \\ &= \mathbf{H}\tilde{\mathbf{e}}^\top, \end{aligned}$$

where, as in the previous section, $\tilde{\mathbf{e}} = \mathbf{e}\mathbf{Q}^\top$.
Q-decoding is then applied to recover $\mathbf{e}$, from which $\mathbf{m}$ can be easily recovered by considering the first $(n_0 - 1)p$ bits of $\mathbf{c} + \mathbf{e}$.

6.1.3 Achieving IND-CCA2 security

To achieve IND-CCA2 security, LEDApkc implements the Kobara-Imai conversion, which is used to avoid issues such as malleability, i.e., the possibility for an adversary to produce a valid ciphertext by modifying an intercepted one. At the current state-of-the-art, we are not aware of attacks of this kind against cryptosystems based on QC-LDPC codes; in any case, using IND-CCA2 conversion completely removes potential security issues. We describe the idea of malleability attacks by describing the case of the original McEliece scheme.

Given an intercepted ciphertext $\mathbf{c}$, an adversary may proceed as follows:

1. produce ciphertexts $\mathbf{c}^{(i)}$, for $i \in [0; n-1]$, where $\mathbf{c}^{(i)}$ is obtained from $\mathbf{c}$ by flipping the i-th bit;
2. query a decryption oracle with the ciphertexts $\mathbf{c}^{(i)}$;
3. for the indexes i corresponding to decryption failures, conclude $e_i = 0$.

This attack rationale is based on the fact that, when $i \in \mathrm{S}(\mathbf{e})$, then $\mathbf{c}^{(i)}$ corresponds to a codeword of the public code corrupted by an error vector of weight $t - 2$, and can

thus be corrected; in the other cases, the weight of the associated error vector is $t+1$, and it cannot be corrected.

To avoid attacks of this kind, LEDApkc employs the conversion described in [58], in which the vectors $\mathbf{m}$ and $\mathbf{e}$ are obtained through hash functions involving the plaintext and some randomness, which is sent along with the ciphertext. After decryption, the validity of the ciphertext is verified through a check on the hash functions. We refer the reader to [58] for the full details of the conversion. To consider a simplified version of the conversion, which comes with the same flavour, we analyze the BIKE-1 algorithm, in which $\mathbf{m}$ is obtained as the hash of $\mathbf{e}$, that is $\mathbf{m} = \mathcal{H}(\mathbf{e})$. Decryption fails if, after having recovered $\mathbf{e}$, we have

$$\mathbf{c} + \mathcal{H}(\mathbf{e}) \cdot \mathbf{G} \neq \mathbf{e}.$$

This guarantees that intercepted ciphertext are not malleable, since modifying $\mathbf{e}$ would also require to sastify the condition on its hash (which is a one-way function).

The principle of malleability attacks applies in analogous way to the Niederreiter cryptosystem. For this reason, LEDAkem comes in two different versions, whose main difference is represented by the life of a key-pair. Indeed, when *ephemeral keys* are used, each key-pair is used for just once key encapsulation. Thus, the aforementioned cryptanalysis techniques cannot be applied, because the adversary does not have possibility of submitting multiple oracle queries. When long term keys are used, however, this kind of attacks must be taken into account. To this end, the current implementation of LEDAcrypt employes a conversion which comes from an adaptation of the one we depict in Algorithms 2 and 3. The proposed conversion can be applied to any Niederreiter scheme; for this reason, we have used Niederreiter.Encrypt and Niederreiter.Decrypt to denote the encryption and decryption functions, respectively. As we see from the algorithm, the conversion works as a sort of XOR cipher, in which the plaintext is summed with the syndrome. In the Algorithm, $\mathcal{H}_t$ denotes an hash function which outputs vectors of weight t; here the plaintext is represented by a binary message $\mathbf{m}$ of size n_t. The proposed conversion has a really low complexity, since it requires only computation of XORs and hash functions but, has a drawback, increases the ciphertext size by n_t bits.

Algorithm 2 Conversion.Encrypt

Input: public key $\mathbf{H}'$, plaintext $\mathbf{m} \in \mathbb{F}_2^{n_t}$.
Output: $\mathbf{y}^{(0)} \in \mathbb{F}_2^r$, $\mathbf{y}^{(1)} \in \mathbb{F}_2^{n_t}$.

1: $\mathbf{e} \leftarrow \mathcal{H}_t(\mathbf{m})$
2: $\mathbf{y}^{(0)} \leftarrow \mathbf{H}\mathbf{e}^\top$ ▷ Syndrome of the error vector
3: $\mathbf{y}^{(1)} \leftarrow \mathcal{H}(\mathbf{e}) \oplus \mathbf{m}$
4: **return** $\{\mathbf{y}^{(0)}, \mathbf{y}^{(1)}\}$

Algorithm 3 Conversion.Decrypt

Input: $\mathbf{y}^{(0)} \in \mathbb{F}_2^r$, $\mathbf{y}^{(1)} \in \mathbb{F}_2^{n_t}$.
Output: plaintext $\in \mathbb{F}_2^{n_t}$ or decryption failure $\perp$.

if Niederreiter.Decrypt $(\mathbf{y}^{(0)}) == \perp$ **then**
  return $\{\perp\}$ ▷ Syndrome decoding has failed
else
  $\mathbf{e} \leftarrow$ Niederreiter.Decrypt $(\mathbf{y}^{(0)})$
  $\mathbf{m} \leftarrow \mathbf{y}^{(1)} \oplus \mathcal{H}(\mathbf{e})$
  if $\mathcal{H}_t(\mathbf{m}) == \mathbf{e}$ **then**
    return $\{\mathbf{m}\}$ ▷ Integrity check is fine
  else
    return $\{\perp\}$ ▷ Integrity check has failed
  end if
end if

6.2 Q-decoder

We briefly recall the functioning of the classical BF decoder which we have presented in Chapter 4 works as follows. At each iteration, for each codeword bit position, the number of unsatisfied parity-check equations is computed: if this number equals or exceeds a given threshold, then the bit is flipped. The decision threshold can be chosen in many ways: for instance, it may be constant for the whole decoding process, or it may vary during iterations. In any case, its value and the way it is eventually updated strongly affect the decoder performances. A choice that usually turns out to be one returning the lowest failure probability is that of setting the threshold, at each iteration, as the maximum number of unsatisfied parity-check equations in which any codeword bit is involved. In fact, a codeword bit participating in a higher number of unsatisfied parity-check equations can be considered less reliable than a codeword bit participating in a smaller number of unsatisfied parity-check equations. So, if the threshold is chosen in this way, the bits that get flipped are those that are most likely to be affected by errors.

In LEDA schemes, the expanded error vector $\tilde{\mathbf{e}}$ to be corrected is influenced by the value of $\mathbf{Q}^\top$: $\tilde{\mathbf{e}}$ is equivalent to a random error vector $\mathbf{e}$ with weight t multiplied by $\mathbf{Q}^\top$. The improved decoder that we describe takes into account the multiplication by $\mathbf{Q}^\top$, to estimate with high efficiency the locations of erred bits in $\mathbf{e}$; for this reason, we call it a *Q-decoder*.

Inputs of the decoder are the syndrome $\mathbf{s}$ and the matrices $\mathbf{H}$ and $\mathbf{Q}$. The output of the decoder is a $1 \times n$ vector $\hat{\mathbf{e}}$ or a decoding failure, where $\hat{\mathbf{e}}$ represents the decoder estimate of the weight-t error vector which has been used during encryption. A threshold criterion is adopted to compute the positions in $\hat{\mathbf{e}}$ that must be changed.
The decoder initialization is performed by setting $\hat{\mathbf{e}} = \mathbf{0}_n$. The decoder runs a maxi-

mum number of iterations which we denote as $\mathtt{i_{max}}$. By denoting with b the decoding threshold, a generic i-th iteration is described as follows

1. Compute the counters as in a traditional BF decoder: the i-th counter, which we denote as σ_i, is obtained as

$$\sigma_j = \sum_{l=0}^{r-1} s_l h_{l,j}.$$

   Store the counters in a vector $\mathbf{\Sigma} = [\sigma_0, \sigma_1, \cdots, \sigma_{n-1}] \in [0; d_v]^n$.

2. Compute $\mathbf{R} = [\rho_0, \rho_1, \cdots, \rho_{n-1}] = \mathbf{\Sigma} * \mathbf{Q}$, where $*$ denotes the integer product.

3. Define $F = \{j \in [0, n-1] |\ \rho_j \geq b\}$.

4. Update $\hat{\mathbf{e}}$ as

$$\hat{\mathbf{e}} = \hat{\mathbf{e}} + \mathbf{1}_F$$

   where $\mathbf{1}_F$ is a length-n binary vector with all-zero entries, except those indexed by F.

5. Update the syndrome as

$$\mathbf{s} = \mathbf{s} + \textstyle\sum_{j \in F} \mathbf{q}_j \mathbf{H}^\top,$$

   where $\mathbf{q}_j$ is the j-th row of $\mathbf{Q}^\top$.

6. If the weight of $\mathbf{s}$ is zero then stop decoding and return $\hat{\mathbf{e}}$.

7. If $i < \mathtt{i_{max}}$ then increment i and go back to step 1, otherwise stop decoding and return a decoding failure.

As in classical BF, the first step of this algorithm computes the vector $\mathbf{\Sigma}$. Each entry of this vector counts the number of unsatisfied parity-check equations corresponding to that bit position, and takes values in $[0; d_v]$. This evaluates the likelihood that the binary element of $\tilde{\mathbf{e}}$ at the same position is equal to one. Differently from classical BF, in step ii) the correlation $\mathbf{r}$ between these likelihoods and the rows of $\mathbf{Q}^\top$ is computed. In fact, the expanded error vector $\tilde{\mathbf{e}} = \mathbf{e}\mathbf{Q}^\top$ can be written as the sum of the rows of $\mathbf{Q}^\top$ indexed by the support of $\mathbf{e}$, that is

$$\tilde{\mathbf{e}} = \sum_{j \in \mathrm{S}(\mathbf{e})} \mathbf{q}_j.$$

Since both $\mathbf{Q}$ and $\mathbf{e}$ are sparse (that is, $m, t \ll n$), cancellations between ones in the sum are very unlikely. When the correlation between $\mathbf{\Sigma}$ and a generic row $\mathbf{q}_i$ of $\mathbf{Q}^\top$ is computed, two cases may occur:

- if $i \notin \mathrm{S}(\mathbf{e})$, then it is very likely that $\mathbf{q}_i$ has a very small number of common ones with all the rows of $\mathbf{Q}^\top$ forming $\tilde{\mathbf{e}}$, hence the expected correlation is small;

- if $i \in \mathrm{S}(\mathbf{e})$, then $\mathbf{q}_i$ is one of the rows of $\mathbf{Q}^\top$ forming $\tilde{\mathbf{e}}$, hence the expected correlation is large.

The main difference with classical BF is that, while in the latter all error positions are considered as independent, the Q-decoder exploits the correlation among expanded errors, since their positions are influenced by $\mathbf{Q}^\top$. This allows achieving important reductions in the number of decoding iterations.

6.2.1 Choice of the Q-decoder decision thresholds

One important aspect affecting performance of the Q-decoder is the choice of the threshold values against which the correlation is compared at each iteration. A natural choice is to set the threshold, for each iteration, equal to the maximum correlation value. This strategy ensures that only those few bits that have maximum likelihood of being affected by errors are flipped during each iteration, thus achieving the lowest DFR. However, such an approach has some drawbacks in terms of complexity, since the computation of the maximum correlation requires additional computations with respect to a fixed threshold.

Therefore, as in [59], we consider a different strategy, which allows computing the threshold values on the basis of the syndrome weight at each iteration. According to this approach, during an iteration it is sufficient to compute the syndrome weight and read the corresponding threshold value from a look-up table. This strategy still allows to achieve a sufficiently low DFR, while employing a significantly smaller number of decoding iterations.

Let us consider the l-th iteration of the Q-decoder, with syndrome $\mathbf{s}$. Let $\tilde{\mathbf{e}}^{(l)}$ denote the corresponding error vector, of weight $\tilde{t}_l$, and let $\mathbf{e}^{(l)}$ be the vector of weight t_l such that $\tilde{\mathbf{e}}^{(l)} = \mathbf{e}^{(l)}\mathbf{Q}^\top$.
The probability that a syndrome bit is set can be obtained as

$$\beta = \frac{\sum_{j=1,\ j \text{ odd}}^{\min\{n_0 d_v, \tilde{t}_l\}} \binom{n_0 d_v}{j}\binom{n_0 p - n_0 d_v}{\tilde{t}_l - j}}{\binom{n_0 p}{\tilde{t}_l}}.$$

The average syndrome weight at iteration l results then in

$$w_s^{(l)} = E\left[wt\left\{s^{(l)}\right\}\right] = p \cdot \beta. \tag{6.3}$$

Since both the parity-check matrix and the error vector are sparse, the probability that the syndrome has a weight which is significantly different from $w_s^{(l)}$ is negligible.

So, (6.3) allows predicting the average syndrome weight starting from $\tilde{t}_l$. In order to predict how $\tilde{t}_l$ varies during iterations, let us consider the i-th codeword bit and the corresponding correlation value $\rho_i^{(l)}$ at the l-th iteration. The probability that such a codeword bit is affected by an error can be written as

$$\Pr\left[e_i = 1|\rho_i^{(l)}\right] = \frac{\Pr\left[e_i = 1, \rho_i^{(l)}\right]}{\Pr\left[\rho_i^{(l)}\right]} = \left(1 + \frac{\Pr\left[e_i = 0, \rho_i^{(l)}\right]}{\Pr\left[e_i = 1, \rho_i^{(l)}\right]}\right)^{-1}$$

where e_i is the i-th bit of the error vector used during encryption. After some calculations, we obtain

$$\Pr\left[e_i = 1|\rho_i^{(l)}\right] = \frac{1}{1 + \frac{n-t_l}{t_l}\left(\frac{p_{ci}(t_l)}{p_{ic}(t_l)}\right)^{\rho_i^{(l)}}\left(\frac{1-p_{ci}(t_l)}{1-p_{ic}(t_l)}\right)^{md_v-\rho_i^{(l)}}}, \tag{6.4}$$

where $p_{ci}(t_l)$ and $p_{ic}(t_l)$ are defined as in [60], that is

$$p_{ci}(t_l) = \sum_{j\,=\,0,\,j\text{ odd}}^{\min[n_0d_v-1,t_l]} \frac{\binom{n_0d_v-1}{j}\binom{n-n_0d_v}{t_l-j}}{\binom{n-1}{t_l}},$$

$$p_{ic}(t_l) = \sum_{j\,=\,0,\,j\text{ even}}^{\min[n_0d_v-1,t_l-1]} \frac{\binom{n_0d_v-1}{j}\binom{n-n_0d_v}{t_l-j-1}}{\binom{n-1}{t_l-1}},$$

Adding the i-th row of $\mathbf{Q}^\top$ to the expanded error vector $\tilde{\mathbf{e}}$ is the same as flipping the i-th bit of the error vector $\mathbf{e}$. Hence, we can focus on $\mathbf{e}$ and on how its weight t_l changes during decoding iterations. The values of $\tilde{t}_l$ can be estimated using (6.3), while, due to sparsity, those of t_l can be estimated as $\tilde{t}_l/m$.

The decision to flip the i-th codeword bit is taken when the following condition is fulfilled

$$\Pr\left[e_i = 1|\rho_i^{(l)}\right] > (1+\Delta)\Pr\left[e_i = 0|\rho_i^{(l)}\right], \tag{6.5}$$

where $\Delta \geq 0$ represents a margin that must be chosen taking into account the DFR and complexity: increasing Δ decreases the DFR but increases the number of decoding iterations. So, a trade-off value of Δ can be found that allows achieving a low DFR while avoiding unnecessary large numbers of iterations.

Since $\Pr\left[e_i = 0|\rho_i^{(l)}\right] = 1 - \Pr\left[e_i = 1|\rho_i^{(l)}\right]$, (6.5) can be rewritten as

$$\Pr\left[e_i = 1|\rho_i^{(l)}\right] > \frac{1+\Delta}{2+\Delta}. \tag{6.6}$$

$\Pr\left[e_i = 1|\rho_i^{(l)}\right]$ is an increasing function of $\rho_i^{(l)}$, hence the minimum value of $\rho_i^{(l)}$

such that (6.6) is satisfied can be computed as

$$b^{(l)} = \min\left\{\rho_i^{(l)} \in [0, md_v], \quad \text{s.t. } \Pr\left[e_i = 1|\rho_i^{(l)}\right] > \frac{1+\Delta}{2+\Delta}\right\}, \tag{6.7}$$

and used as the decision threshold at iteration l.

Based on the above considerations, the procedure to compute the decision threshold value per each iteration as a function of the syndrome weight can be summarized as follows:

1. The syndrome weights corresponding to $t'_l = 0, m, 2m, \cdots, mt$ (which are all the possible values of t'_l neglecting cancellations) are computed according to (6.3). These values are denoted as $\{w_s(0), w_s(m), \cdots, w_s(mt)\}$.

2. At iteration l, given the syndrome weight $\bar{w_s}^{(l)}$, the integer $j \in [0, t]$ such that $w_s(jm)$ is as close as possible to $\bar{w_s}^{(l)}$ is computed.

3. Consider $t_l = j$ and compute $b^{(l)}$ according to (6.7) and (6.4). The value of $b^{(l)}$, so obtained, is used as the decoding threshold for iteration l.

The above procedure can be implemented efficiently by populating a look-up table with the pairs $\{w_j, b_j\}$, sequentially ordered. During an iteration, it is enough to compute $\bar{w_s}^{(l)}$, search the largest w_j in the look-up table such that $w_j < \bar{w_s}^{(l)}$ and set $b^{(l)} = b_j$.

We have observed that, moving from large to smalle values of w_j, the thresholds computed this way firstly exhibit a decreasing trend, then start to increase. According to numerical simulations, neglecting the final increase is beneficial from the performance standpoint. Therefore, in the look-up table we replace the threshold values after the minimum with a constant value equal to the minimum itself.

6.2.2 Relations with QC-MDPC code-based systems

In LEDA cryptosystems, the public code is a QC-MDPC code that admits $\mathbf{L} = \mathbf{HQ}$ as a valid parity-check matrix. However, differently from QC-MDPC code-based schemes, the private code is a QC-LDPC code, which facilitates decoding. In fact, decoding directly the public QC-MDPC code through classical BF decoders would be a possibility, but the approach we follow is different. By using the Q-decoder, we decode the private QC-LDPC code, taking into account the correlation introduced in the private error vector due to multiplication by $\mathbf{Q}^\top$.

Besides working over different matrices, the main difference between these two decoding algorithms is in the use of integer multiplications in our decoder, while all multiplications are performed over $\mathbb{F}_2$ in classical BF decoders. In fact, in our decoder

we perform the following operation to compute the correlation vector $\mathbf{r}$

$$\mathbf{r} = \mathbf{s} * \mathbf{H} * \mathbf{Q} = \mathbf{e}\mathbf{Q}^\top\mathbf{H}^\top * \mathbf{H} * \mathbf{Q} \approx \mathbf{e}\mathbf{L}\top * \mathbf{L},$$

where the last approximation comes from the fact that, for two sparse matrices $\mathbf{A}$ and $\mathbf{B}$, we have $\mathbf{A} \cdot \mathbf{B} \approx \mathbf{A} * \mathbf{B}$. Thus, we can say that $\mathbf{HQ} \approx \mathbf{H} * \mathbf{Q}$. So, if we consider classical BF decoding working over the matrix $\mathbf{L} = \mathbf{HQ}$, the counter vector is computed as

$$\mathbf{\Sigma} = \mathbf{s} * \mathbf{L} = \mathbf{e}\mathbf{L}^\top * \mathbf{L}.$$

In the Q-decoder, the error vector is updated by summing rows of $\mathbf{Q}^\top$, which is equivalent to flipping bits of the public error vector. Hence, there is a clear analogy between decoding of the private QC-LDPC code through the Q-decoder and decoding of the public QC-MDPC code through a classical BF decoder. Through numerical simulations we have verified that the two approaches yield comparable performance in the waterfall region.

6.3 Security analysis

In this section we consider possible attack avenues against LEDA cryptosystems; in particular, in this section we focus on classical attacks such as key enumeration and ISD, and leave statistical attacks to the last part of this chapter.

6.3.1 Attacks based on exhaustive key search

Enumerating all the possible values for the secret key is, in principle, applicable to any cryptosystem. While the cost of performing an exhaustive key search is dominated by less computational demanding key recovery strategies in LEDA cryptosystems, we consider partial key enumeration attacks, aiming at scanning through all the possible $\mathbf{H}$ or $\mathbf{Q}$ sparse matrices as a support for other strategies. While there is no standing attack benefiting from an enumeration of possible $\mathbf{H}$, from the enumeration of possible $\mathbf{Q}$ matrices or from the enumeration of both of them, we deem reasonable adding such a constraint to the design of the parameter sets as a peace-of-mind measure and obtain a system for which the said enumerations are computationally unfeasible.

We recall that $\mathbf{H}$ is a block circulant binary matrix constituted by $1 \times n_0$ circulant-blocks, each of which having size equal to p bits, while $n_0 \in \{2, 3, 4\}$ and p is a prime such that $\mathrm{ord}_2(p) = p - 1$ (i.e., $2^{p-1} \bmod p = 1 \bmod p$). $\mathbf{Q}$ is a binary circulant-block matrix constituted by $n_0 \times n_0$ binary circulant-blocks with size p.

Considering that each row of a circulant-block of $\mathbf{H}$ has Hamming weight d_v, a straightforward counting argument yields $\sharp\mathbf{H} = \binom{p}{d_v}^{n_0}$ as the number of possible

choices for $\mathbf{H}$. The number of possible choices for $\mathbf{Q}$, denoted as $\sharp\mathbf{Q}$, can be derived starting from the consideration that the weights of a row of each circulant block in a block-row of $\mathbf{Q}$ are equal for all the rows up to a rotation of the weights of the blocks. Such weights, denoted as $\{m_0, \ldots, m_{n_0-1}\}$, allow to write the number of possible choices for $\mathbf{Q}$ as $\sharp\mathbf{Q} = \left[\prod_{i \in \{m_0, \ldots, m_{n_0-1}\}} \binom{p}{i} \right]^{n_0}$.

We also consider the possibility that an attacker performs an exhaustive key search employing a quantum computer. In such a case, the best scenario for the attacker is that it is possible to exploit Grover's algorithm to compute either $\mathbf{H}$ or $\mathbf{Q}$, and test its correctness in deriving the other matrix and the corresponding public key. Assuming conservatively that the test can be implemented on a quantum computer, we consider the resistance against exhaustive key search with a quantum computer to be the minimum between $\sqrt{\sharp\mathbf{H}}$ and $\sqrt{\sharp\mathbf{Q}}$ for the search over $\mathbf{H}$ and $\mathbf{Q}$, respectively.

For all parameters of practical interest, brute search attacks never represent an issue.

6.3.2 Attacks based on Information Set Decoding

The most computationally effective technique known to attack LEDA schemes is represented by ISD. In particular, when QC-LDPC cryptosystems are used, ISD attacks can be used to perform both decoding attacks, aimed at decrypting an intercepted ciphertext, and key recovery attacks.

ISD decoding attacks

When a message recovery attack of this kind is performed against a cryptosystem exploiting quasi cyclic codes, such as the case of LEDA, it is known that a speedup equal to the square root of the circulant block size can be achieved [55]. For the case of LEDAcrypt, the speed up is measured as $\sqrt{p}$.

ISD key recovery attacks

In the case of McEliece and Niederreiter cryptosystems instantiated with public codes characterized by sparse parity-check matrices, ISD algorithms can also be used to mount key recovery attacks. In the following, we describe three different attack strategies; we will denote with $C_{\texttt{ISD}}(n, r, t)$ the complexity of finding a codeword of weight t in the code with length n and redundancy r.

1. In LEDA, the public key is the representation of a code $\mathcal{C}'$ whose parity-check

matrix $\mathbf{L}$ is in the form

$$\mathbf{L} = \left[\mathbf{L}_0, \mathbf{L}_1, \cdots, \mathbf{L}_{n_0-1}\right],$$

where each block $\mathbf{L}_i$ is a $p \times p$ circulant matrix with weight $d'_v \leq d_v m$.
It can be shown that $\mathcal{C}'$ has minimum distance $\leq 2d'_v$; indeed, let us consider two distinct integers $0 \leq i_0, i_1 \leq n_0 - 1$, and define the $p \times n_0 p$ matrix $\mathbf{C}$ as follows

$$\mathbf{C} = [\mathbf{C}_0, \cdots, \mathbf{C}_{n_0-1}], \text{ with } \mathbf{C}_i \in \mathbb{F}_2^{p\times p}, \ \mathbf{C}_i = \begin{cases} 0 & \text{if } i \neq i_0, i_1 \\ \mathbf{L}_{i_1}^\top & \text{if } i = i_0 \\ \mathbf{L}_{i_0}^\top & \text{if } i = i_1 \end{cases}. \tag{6.8}$$

It is easy to see that the rows of $\mathbf{C}$ are codewords of $\mathcal{C}'$, as

$$\mathbf{C}\mathbf{L}^\top = \mathbf{C}_{i_0}\mathbf{L}_{i_0}^\top + \mathbf{C}_{i_1}\mathbf{L}_{i_1}^\top = \mathbf{L}_{i_1}^\top L_{i_0}^\top + \mathbf{L}_{i_0}^\top \mathbf{L}_{i_1}^\top = 0,$$

where the last equality is justified by the fact that multiplication between circulant matrices is commutative.

If an attacker succeeds in determining one of the rows of $\mathbf{C}$, which are codewords of the public code $\mathcal{C}'$, he will be able to recover two blocks $\mathbf{L}_i, \mathbf{L}_j$ of secret parity check matrix $\mathbf{L}$. From such values, the attacker will be able to determine the value of $\mathbf{L}_{n_0-1}^{-1}$ from the blocks in the public key, and consequentially derive the full secret parity matrix $\mathbf{L}$.

Each one of the rows $\mathbf{c}$ of $\mathbf{C}$ has weight $2d'_v$, and can thus be searched by exploiting an ISD algorithm with a cost $C_{\texttt{ISD}}(n_0 p, p, 2d'_v)$. We note that more than a codeword with weight $2d'_v$ is present, thus resulting in a speedup of the codeword finding attack. To quantify the number of codewords, consider that we have $\binom{n_0}{2}$ possible matrices $\mathbf{C}$ as in Eq. (6.8), each one containing p rows: thus, the number of codewords in $\mathcal{C}'$ with weight $2d'_v$ (and the subsequent speedup in ISD) is obtained as $p\binom{n_0}{2}$.

2. Another attack strategy consists in applying ISD after a puncturation of the public code. Consider the systematic generator matrix for $\mathcal{C}'$, with the form

$$\mathbf{G}_{\texttt{sys}} = \left[\begin{array}{cccc|c} \mathbf{I}_p & & & & \mathbf{G}_0 \\ & \mathbf{I}_p & & & \mathbf{G}_1 \\ & & \ddots & & \vdots \\ & & & \mathbf{I}_p & \mathbf{G}_{n_0-2} \end{array}\right] = \left[\begin{array}{cccc|c} \mathbf{I}_p & & & & \left(\mathbf{L}_0\mathbf{L}_{n_0-1}^{-1}\right)^\top \\ & \mathbf{I}_p & & & \left(\mathbf{L}_1\mathbf{L}_{n_0-1}^{-1}\right)^\top \\ & & \ddots & & \vdots \\ & & & \mathbf{I}_p & \left(\mathbf{L}_{n_0-2}\mathbf{L}_{n_0-1}^{-1}\right)^\top \end{array}\right].$$

Pick a a block $\mathbf{G}_i$ and constitute the matrix $\mathbf{G}^* = [\mathbf{I}_p, \mathbf{G}_i]$. $\mathbf{G}^*$ can be thought

as the generator matrix of a code $\mathcal{C}^*$, with length $2p$ and dimension p, which, we show, contains codewords of weight $\leq 2d'_v$. Indeed, we have

$$\mathbf{L}_{n_0-1}^{\top}\mathbf{G}^* = [\mathbf{L}_{n_0-1}^{\top}; \mathbf{L}_{n_0-1}^{\top}\left(\mathbf{L}_i\mathbf{L}_{n_0-1}^{T}\right)^{\top}] = [\mathbf{L}_{n_0-1}^{\top}; \mathbf{L}_i^{\top}].$$

We thus have that the low weight codewords of $\mathcal{C}^*$ reveal enough information to reconstruct the secret key from the public key. It is thus possible to apply an ISD technique to solve the find such codewords, with complexity $C_{\texttt{ISD}}(2p, p, 2d'_v)$. Note that there are p codewords of weight $2d'_v$ for each block $\mathbf{G}_i$ employed to build $\mathbf{G}^*$, therefore resulting in a speedup for the ISD equal to $n_0 p$

3. A third way to exploit ISD consists in searching for low weight codewords in the dual of the public code, which we denote $\mathcal{C}'^{\perp}$. A valid generator matrix for such a code is thus the $\mathbf{L}$ matrix, which has, by construction, low weight codewords. Indeed, the rows of $\mathbf{L}$ have weight $\leq n_0 d'_v$ It is thus possible to solve the codeword finding problem on $\mathcal{C}'^{\perp}$ with a cost $C_{\texttt{ISD}}(n_0 p, (n_0-1)p, n_0 d'_v)$. Due to the quasi cyclic nature of $\mathbf{L}$, the ISD cost will be reduced by a factor of of p.

Taking into account ISD attacks is necessary, in order to derive proper parameters for the values of t, d_v and m. Essentially, using ISD complexity estimations for the decoding message attack will return the minimum value of t which guarantees security, while considering key recovery attacks will return the minimum value of d'_v which guarantees security. Then, the values of d_v and m can be properly chosen. However, as we describe in the following section, the particular structure of the matrix $\mathbf{L}$ should be taken into account, in order to avoid weak keys, i.e., secret keys for which ISD attacks are facilitated.

6.3.3 Weak keys in LEDA cryptosystems

Recall Eq. (6.2): for each circulant block in $\mathbf{L}$, we have

$$\mathbf{L}_i = \sum_{j=0}^{n_0-1} \mathbf{H}_j \mathbf{Q}_{j,i}.$$

It can clearly happen that the weight of some $\mathbf{L}_i$ is lower than md_v: if such cases, rows of $\mathbf{L}$ will have weight $< n_0 m d_v$. Since the complexity of ISD key recovery attacks grows with the weight of rows of $\mathbf{L}$, it is then clear that, for secret keys with this property, such attacks are facilitated. To derive a detailed analysis on the number of actually weak keys, we should first compute the threshold row weight $\bar{w}$, i.e., the highest row weight for which the complexity of key recovery attacks is below the target security level. Then, to ensure security of the system, we should add a rejection sampling step in the key generation algorithm. For each secret key, we compute the

row weight of the corresponding $\mathbf{L}$: if this weight is $\leq \bar{w}$, then the secret key is discarded and a new one is generated.

In LEDA cryptosystems, with a conservative choice, this threshold value is set as md_v: each time a secret key produces a cancellation in the computation of $\mathbf{L}$, the key is discarded. It is clear that this rejection sampling step reduces the secret key cardinality, so the rejection ratio should be taken into account. Indeed, if this ratio is too high, the key generation algorithm becomes inefficient (since, on average, a huge number of keys are discarded before a valid one is generated) and, furthermore, brute force attacks are facilitated. However, despite the conservative choice of $\bar{w} = md_v$, for practical parameters of LEDA cryptosystems the fraction of rejected keys is particularly low. Thus, this ensures that the aforementioned issues of efficiency and security do not appear. In the following we provide a theoretical estimate on the rejection ratio.

First of all, we define $\mathrm{Pr}_{\oplus}(p, v_1, v_2, v_x)$ as the probability that the sum of two random length-p vectors with weights v_1 and v_2 results in a vector with weight v_x; we have

$$\mathrm{Pr}_{\oplus}(p, v_1, v_2, v_x) = \begin{cases} \frac{\binom{v_1}{\frac{v_1+v_2-v_x}{2}}\binom{p-v_1}{v_2-\frac{v_1+v_2-v_x}{2}}}{\binom{p}{v_2}}, & \text{if } \begin{cases} |v_1 - v_2| \leq v_x \\ v_x \leq v_1 + v_2, \\ v_x \equiv v_1 + v_2 \mod 2, \end{cases} \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathrm{Pr}_{\oplus}^{(N)}(p, v, v_x)$ be the probability that the sum of N random length-p vectors with weight v results in a vector with weight v_x. This probability can be recursively defined as

$$\mathrm{Pr}_{\oplus}^{(N)}(p, v, v_x) \begin{cases} \begin{cases} 0 & \text{if } v^{(0)} \neq v, \\ 1 & \text{if } v^{(0)} = v, \end{cases} & \text{if } N = 1, \\ \sum_{v^{(N-1)}=0}^{p} \mathrm{Pr}_{\oplus}^{(N-1)}(p, v, v^{(N-1)})\mathrm{Pr}_{\oplus}(p, v, v^{(N-1)}, v_x), & \text{if } N \geq 2. \end{cases} \tag{6.9}$$

The above probabilities can be used to estimate the row weight distribution of $\mathbf{L}$. We remind that all the blocks $\mathbf{H}_j$ have row and column weight d_v, while each block $\mathbf{Q}_{j,i}$ has row and column weight which we denote with $m_{j,i}$. Then, each product $\mathbf{H}_j\mathbf{Q}_{j,i}$ can be described as the sum of $m_{j,i}$ random circulant blocks with row weight d_v, and thus its weight distribution can be estimated through Eq. (6.9), with $N = m_{j,i}$ and $v = d_v$. The weight distribution of $\mathbf{L}_i$ can then be computed as $\mathrm{Pr}_i^{(n_0-1)}(v_x)$, where the function Pr_i is defined through the following recursive expression

$$\mathrm{Pr}_i^{(j)}(v_x) = \begin{cases} \mathrm{Pr}_{\oplus}^{(m_{0,i})}(p, d_v, v_x), & \text{if } j = 0, \\ \sum_{v_x=0}^{p}\sum_{v_1=0}^{p}\sum_{v_2=0}^{p} \mathrm{Pr}_i^{(j-1)}(v_1)\mathrm{Pr}_{\oplus}^{(m_{j,i})}(p, d_v, v_2)\mathrm{Pr}_{\oplus}(p, v_1, v_2, v_x) & \text{otherwise.} \end{cases}$$

We finally obtain the weight distribution of a row of $\mathbf{L}$ as

$$\Pr_{\mathbf{L}}(w_x) = \prod_{i=0}^{n_0-1} \Pr_i^{(n_0-1)}(v_i), \quad \forall v_0, \cdots, v_{n_0-1} \text{ s.t. } \sum_{i=0}^{n_0-1} v_i = w_x.$$

The rejection rate is hence estimated as

$$\eta = 1 - \Pr_{\mathbf{L}}(n_0 m d_v),$$

and, for practical parameters of LEDA cryptosystems, is always close to $1/2$.

6.4 Statistical attacks

In this section we describe a family of attacks based on statistical analyses, namely *statistical attacks*. This family includes reaction attacks, in which data is collected through the observation of Bob's reactions, and side-channel attacks. Historically, this kind of attacks on QC-LDPC and QC-MDPC code based cryptosystems has been introduced by Guo et al. in 2016 [16], by showing how to use decryption failures to reconstruct the secret key of QC-MDPC code based cryptosystems; we will refer to this attack as GJS (the name is the acronym of the authors last names). In 2017 the attack has been generalized to the case of LEDA cryptosystems [18,19], by modifying the GJS attack to take into account the presence of matrix a $\mathbf{Q}$ with $m > 1$. Finally, these procedures have been improved, by considering the information leakage due to other quantities, such as the number of decoding iterations [17,61]. We point out that all the aforementioned attacks are strictly based on quasi-cyclicity and thus, cannot be applied to codes with a different geometrical structure.

In this section we recall the results of [62] and present a general framework which embeds all the aforementioned attacks; furthermore, we show that the these attacks do not strictly depend on the QC structure. Furthermore our proposed technique, with respect to the aforementioned attacks, allows for recovering a larger amount of information about the secret key.

One final remark regards the applicability of the results we describe in this section to LEDA cryptosystems. As we have already said, Q-decoding on $\mathbf{H}$ and $\mathbf{Q}$ approximates BF-decoding on $\mathbf{L}$. As observed in [63], statistical attacks designed for QC-MDPC schemes (i.e., for the case $m = 1$) can equivalently be applied to LEDA cryptosystems. Because of the relation between Q-decoding and BF-decoding, the attacks will be successful and will return the matrix $\mathbf{L}$. Thus, from now on, for the sake of simplicity, we will just consider the case of QC-MDPC cods decoded through BF.

6.4.1 A general model for statistical attacks

Let us consider a public-key cryptosystem with private and public keys sk and pk, respectively, and security parameter λ. We denote with $\mathsf{Decrypt}(sk, \mathbf{x})$ a generic decryption algorithm that, given a ciphertext $\mathbf{x}$ and sk as inputs, returns either the plaintext $\mathbf{m}$ or a decryption failure. We define $\mathcal{D}(sk, \mathbf{x})$ as an oracle that, queried with a ciphertext $\mathbf{x}$, runs $\mathsf{Decrypt}(sk, \mathbf{x})$ and returns some metrics that describe the execution of the decryption algorithm. More details about the typology of the oracle's replies are provided next.
An adversary, which is given pk, queries the oracle with N ciphertexts $\{\mathbf{x}^{(i)} \,|i = 1, \cdots, N\}$; we denote as $y^{(i)}$ the oracle's reply to the i-th query $\mathbf{x}^{(i)}$. The adversary then runs an algorithm $\mathcal{A}(K_P, \{\mathbf{x}^{(0)}, y^{(0)}\}, \cdots, \{\mathbf{x}^{(N-1)}, y^{(N-1)}\})$ that takes as inputs pk and the pairs of oracle queries and replies, and returns sk'. The algorithm $\mathcal{A}$ models the procedure that performs a statistical analysis of the gathered data and reconstructs the secret key, or an equivalent version of it. The time complexity of this whole procedure can be approximated as

$$C = \alpha N + C_{\mathcal{A}},$$

where α corresponds to the average number of operations performed for each query and $C_{\mathcal{A}}$ is the complexity of executing the algorithm $\mathcal{A}$. The adversary is then challenged with a randomly generated ciphertext $\mathbf{x}^*$, corresponding to a plaintext $\mathbf{m}^*$. We consider the attack successful if $C < 2^{\lambda}$ and the probability of $\mathbf{m} = \mathsf{Decrypt}(sk', \mathbf{x}^*)$ being equal to $\mathbf{m}^*$ is not negligible (i.e., larger than $2^{-\lambda}$).

We point out that this formulation is general, since it does not distinguish between the McEliece and Niederreiter cases. In the same way the private and public keys might be generic. Furthermore, the above model allows for taking into account many kinds of attacks, depending on the oracle's reply. For instance, when considering attacks based on decryption failures, the oracle's reply is a boolean value which is true in case of a failure and false otherwise. When considering timing attacks based on the number of iterations, then the oracle's reply corresponds to the number of iterations run by the decoding algorithm.

6.4.2 The GJS attack

In this section we describe the GJS attack [16], in the case of a scheme employing a CCA conversion like that in [58]. This means that the error vectors which are used during encryption are obtained as the result of an hash function; this corresponds to assuming that the oracle queries are all randomly generated, i.e., the error vectors used during encryption can be seen as randomly picked elements from the ensemble of all n-uples with weight t.

As we have already stated, state-of-the-art statistical attacks [17–19, 64, 65] are specific to the sole case of QC codes defined through a secret parity-check matrix in the

form

$$\mathbf{H} = \left[\mathbf{H}_0, \mathbf{H}_1, \ldots, \mathbf{H}_{n_0-1}\right], \tag{6.10}$$

where each $\mathbf{H}_i$ is a circulant matrix of weight d'_v and size p, and n_0 is a small integer. All such attacks are focused on gathering information about the existence (or absence) of some cyclic distances between symbols 1 in $\mathbf{H}$. In particular, an adversary aims at recovering the following quantities, which were introduced in [64].

Distance spectrum: *Given a vector* $\mathbf{a}$*, with support* $\mathrm{S}(\mathbf{a})$ *and length* p*, its* distance spectrum *is defined as*

$$\mathrm{DS}(\mathbf{a}) = \{\min\{\pm(i-j) \mod p\} | \, i, j \in \mathrm{S}(\mathbf{a}), \;\; i \neq j\}.$$

Multiplicity: *We say that a distance* $d \in \mathrm{DS}(\mathbf{a})$ *has* multiplicity μ_d *if there are* μ_d *distinct pairs in* $\mathrm{S}(\mathbf{a})$ *which produce the same distance* d*.*

Basically, the distance spectrum is the set of all distances with multiplicity larger than 0. It can be easily shown that all the rows of a circulant matrix are characterized by the same distance spectrum; thus, given a circulant matrix $\mathbf{M}$, we denote the distance spectrum of any of its rows (say, the first one) as $\mathrm{DS}(\mathbf{M})$.

Statistical attacks proposed in the literature aim at estimating the distance spectrum of the circulant blocks in the secret $\mathbf{H}$, and are based on the observation that some quantities that are typical of the decryption procedure depend on the number of common distances between the error vector and the rows of the parity-check matrix. In particular, the generic procedure of a statistical attack on a cryptosystem whose secret key is in the form (6.10) is described in Algorithm 4; we have called the algorithm Ex-GJS in order to emphasize the fact that it is an extended version of the original GJS attack [64], which was only focused on a single circulant block in $\mathbf{H}$. Our algorithm, which is inspired by that of [18], is a generalization of the procedure in [64], in which all the circulant blocks in $\mathbf{H}$ are taken into account. We present this algorithm in order to show the maximum amount of information that state-of-the-art statistical attacks allow to recover.

The error vector used for the i-th query is expressed as $\mathbf{e}^{(i)} = [\mathbf{e}_0^{(i)}, \ldots, \mathbf{e}_{n_0-1}^{(i)}]$, where each $\mathbf{e}_j^{(i)}$ has length p. The estimates $\mathbf{a}^{(0)}, \ldots, \mathbf{a}^{(n_0-1)}$ and $\mathbf{b}^{(0)}, \ldots, \mathbf{b}^{(n_0-1)}$ are then used by the adversary to guess the distance spectra of the blocks in the secret key. Indeed, let us define $\mathcal{E}^{(d,j)}(n,t)$ as the ensemble of all error vectors having length n, weight t and such that they exhibit a distance d in the distance spectrum of the j-th length-p block. Then, depending on the meaning of the oracle's reply, the ratios $a_d^{(j)}/b_d^{(j)}$ correspond to the estimate of the average value of some quantity, when the error vector belongs to $\mathcal{E}^{(d,j)}(n,t)$. For instance, when considering attacks based on decryption failures, the oracle's reply is either 0 or 1, depending on whether the

Algorithm 4 Ex-GJS

Input: public key pk, number of queries $N \in \mathbb{N}$
Output: estimates $\mathbf{a}^{(0)}, \cdots, \mathbf{a}^{(n_0-1)}, \mathbf{b}^{(0)}, \cdots, \mathbf{b}^{(n_0-1)} \in \mathbb{N}_{\lfloor p/2 \rfloor}$.

Initialize $\mathbf{a}^{(0)}, \cdots, \mathbf{a}^{(n_0-1)}, \mathbf{b}^{(0)}, \cdots, \mathbf{b}^{(n_0-1)}$ as null arrays of length $\lfloor p/2 \rfloor$
for $i \leftarrow 1$ **to** N **do**
  $\mathbf{x}^{(i)} \leftarrow$ ciphertext obtained through pk, using the error vector $\mathbf{e}^{(i)}$
  $y^{(i)} \leftarrow \mathcal{D}(sk, \mathbf{x}^{(i)})$
  for $j \leftarrow 0$ **to** $n_0 - 1$ **do**
    $\Delta_j \leftarrow \mathrm{DS}(\mathbf{e}_j^{(i)})$
    for $d \in \Delta_j$ **do**
      $a_d^{(j)} \leftarrow a_d^{(j)} + y^{(i)}$
      $b_d^{(j)} \leftarrow b_d^{(j)} + 1$
    end for
  end for
end for
return $\{\mathbf{a}^{(0)}, \cdots, \mathbf{a}^{(n_0-1)}, \mathbf{b}^{(0)}, \cdots, \mathbf{b}^{(n_0-1)}\}$

decryption was successful or failed. In such a case, the ratio $a_d^{(j)}/b_d^{(j)}$ corresponds to an empirical measurement of the DFR, conditioned to the event that the error vector belongs to $\mathcal{E}^{(j,d)}(n,t)$. In general, statistical attacks are successful because many quantities that are typical of the decoding procedure depend on the multiplicity of the distances in $\mathrm{DS}(\mathbf{H}_j)$, $\forall j \in [0, \ldots, n_0 - 1]$. As we show next, this property is not strictly due to the existence of common distances, but more generally depends on the number of overlapping ones between columns in $\mathbf{H}$.

6.4.3 General statistical attacks

In this section we generalize the Ex-GJS procedure, and describe an algorithm which can be used to recover information about any regular code. In particular, our analysis shows that events of decoding failure i) do not strictly depend on the QC structure of the adopted code, and ii) permit to retrieve a quantity that is more general than distance spectra.

We first show that, for generic regular codes, there is a connection between the syndrome weight and the DFR. This statement is validated by numerical simulations on (w, v)-regular codes, obtained through Gallager construction [15], in which $\frac{v}{w} = \frac{r}{n}$. We consider a BF decoder running for a maximum of $\mathtt{i}_{\mathtt{max}}$ iterations, with unique decoding threshold which we denote with b. In particular, we have considered two codes with length $n = 5,000$, redundancy $r = 2,500$ and different pairs (w, v), decoded with BF Algorithm; their DFR vs. syndrome weight is shown in Fig. 6.1.

We notice from Fig. 6.1 that there is a strong dependence between the initial syndrome weight and the DFR and that different pairs (v, w) can lead to two different

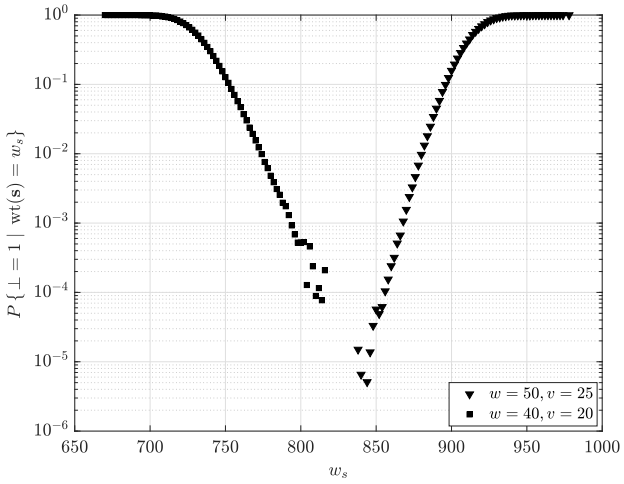


Figure 6.1: Distribution of the DFR as a function of the syndrome weight, for two regular (w, v)-regular LDPC codes, decoded through BF with $\mathtt{i}_{\mathtt{max}} = 5$ and $b = 15$. The weight of the error vectors is $t = 58$; for each code, 10^7 decoding instances have been considered.

trends in the DFR evolution.

Let us now define $\mathcal{E}(n, t, i_0, i_1)$ as the ensemble of all vectors having length n, weight t and whose support contains elements i_0 and i_1. Let $\mathbf{s}$ be the syndrome of an error vector $\mathbf{e} \in \mathcal{E}(n, t, i_0, i_1)$: we have

$$\mathbf{s} = \mathbf{h}_{i_0} + \mathbf{h}_{i_1} + \sum_{j \in \mathrm{S}(\mathbf{e}) \setminus \{i_0, i_1\}} \mathbf{h}_j.$$

The syndrome weight has a probability distribution that depends on the interplay between $\mathbf{h}_{i_0}$ and $\mathbf{h}_{i_1}$: basically, when these two columns overlap in a small (large) number of ones, then the average syndrome weight gets larger (lower). Moreover, motivated by the empirical evidence of Fig. 6.1, one can expect that the DFR experienced over error vectors belonging to different ensembles $\mathcal{E}(n, t, i_0, i_1)$ depends on the number of overlapping ones between columns $\mathbf{h}_{i_0}$ and $\mathbf{h}_{i_1}$. Then, a statistical attack against a generic regular code can be mounted, as described in Algorithm 5, which we denote as General Statistical Attack (GSA). The output of the algorithm is represented by the matrices $\mathbf{A}$ and $\mathbf{B}$, which are used by the adversary to estimate the average value of the oracle's replies, as a function of the pair (i_0, i_1). Notice that in Algorithm 5 the oracle's reply is denoted as $y^{(i)}$ and does not need to be better specified. We will indeed show in the next section that the same procedure can be used to exploit other information sources than the success (or failure) of decryption. We now focus on the case of $y^{(i)}$ being 0 or 1, depending on whether decryption was successful

or not. Then, each ratio $a_{j,l}/b_{j,l}$ represents an empirical estimate of the probability of encountering a decryption failure, when the error vector contains both j and l in its support. One might expect that the ratios $a_{j,l}/b_{j,l}$ are distributed on the basis of

Algorithm 5 GSA

Input: public key pk, number of queries $N \in \mathbb{N}$
Output: estimates $\mathbf{A}, \mathbf{B} \in \mathbb{N}^{n \times n}$.

$\mathbf{A} \leftarrow \mathbf{0}_{n \times n}$
$\mathbf{B} \leftarrow \mathbf{0}_{n \times n}$
for $i \leftarrow 1$ **to** N **do**
  $\mathbf{x}^{(i)} \leftarrow$ ciphertexts obtained through the error vector $\mathbf{e}^{(i)}$
  $y^{(i)} \leftarrow \mathcal{D}(sk, \mathbf{x}^{(i)})$
  $\mathrm{S}\left(\mathbf{e}^{(i)}\right) \leftarrow$ support of $\mathbf{e}^{(i)}$
  for $j \in \mathrm{S}\left(\mathbf{e}^{(i)}\right)$ **do**
    for $l \in \mathrm{S}\left(\mathbf{e}^{(i)}\right)$ **do**
      $a_{j,l} \leftarrow a_{j,l} + y^{(i)}$
      $b_{j,l} \leftarrow b_{j,l} + 1$
    end for
  end for
end for
return $\{\mathbf{A}, \mathbf{B}\}$

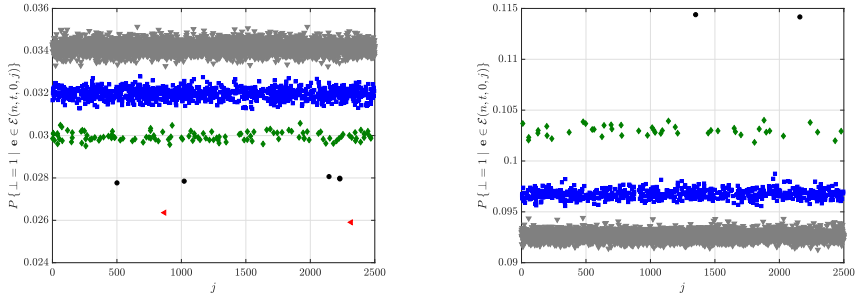
the number of overlapping ones between columns j and l in $\mathbf{H}$. We have verified this intuition by means of numerical simulations; the results we have obtained are shown in Fig. 6.2, for the case of error vectors belonging to ensembles $\mathcal{E}(n, t, 0, j)$, with $j \in [1, \ldots, n-1]$. The figure clearly shows that the ratios $a_{j,l}/b_{j,l}$ can be used to guess the number of overlapping ones between any pair of columns in $\mathbf{H}$.

These empirical results confirm the conjecture that the DFR corresponding to error vectors in $\mathcal{E}(n, t, i_0, i_1)$ depends on the number of overlapping ones between the columns i_0 and i_1. Moreover, these results show that the same idea of [64], with some generalization, can be applied to whichever kind of code.

We now show that even when QC codes are considered, our algorithm recovers more information than that which can be obtained through the Ex-GJS procedure. For such a purpose, let us consider a parity-check matrix in the form (6.10), and let $\gamma_{i,j}$ be the number of overlapping ones between columns i and j. Now, because of the QC structure, we have

$$\left|\mathrm{S}\left(\mathbf{h}_i\right) \cap \mathrm{S}\left(\mathbf{h}_j\right)\right| = \left|\mathrm{S}\left(\mathbf{h}_{p\lfloor i/p \rfloor + [i+z \mod p]}\right) \cap \mathrm{S}\left(\mathbf{h}_{p\lfloor j/p \rfloor + [j+z \mod p]}\right)\right|, \quad \forall z. \tag{6.11}$$

We now consider two columns that belong to the same circulant block in $\mathbf{H}$, i.e., $i = pi_p + i'$, $j = pi_p + j'$, with $0 \leq i_p \leq n_0 - 1$ and $0 \leq i', j' \leq p - 1$; then, (6.11)



(a) (b)

Figure 6.2: Simulation results for (v, w)-regular codes with $n = 5000$, $k = 2500$, for $t = 58$ and for error vector belonging to ensembles $\mathcal{E}(n, t, 0, j)$, for $j \in [1, \ldots, n-1]$. The parameters of the codes are $v = 25$, $w = 50$ for Figure (a), $v = 20$, $w = 40$ for Figure (b); the decoder settings are $i_{\mathtt{max}} = 5$ and $b = 15$. The results have been obtained through the simulation of 10^9 decoding instances. Grey, blue, green, black and red markers are referred to pairs of columns with number of intersections equal to $0, 1, 2, 3, 4$, respectively.

can be rewritten as

$$|\mathrm{S}(\mathbf{h}_i) \cap \mathrm{S}(\mathbf{h}_j)| = \left|\mathrm{S}\left(\mathbf{h}_{pi_p+[i'+z \mod p]}\right) \cap \mathrm{S}\left(\mathbf{h}_{pi_p+[j'+z \mod p]}\right)\right|, \quad \forall z.$$

With some simple computations, we finally obtain

$$\left|\mathrm{S}(\mathbf{h}_{pi_p+i'}) \cap \mathrm{S}(\mathbf{h}_{pi_p+j'})\right| = \begin{cases} \left|\mathrm{S}(\mathbf{h}_{pi_p}) \cap \mathrm{S}(\mathbf{h}_{pi_p+p-(i'-j')})\right| & \text{if } j' < i' \\ \left|\mathrm{S}(\mathbf{h}_{pi_p}) \cap \mathrm{S}(\mathbf{h}_{pi_p+j'-i'})\right| & \text{if } j' > i' \end{cases}. \quad (6.12)$$

Similar considerations can be carried out if the two columns do not belong to the same circulant block. So, (6.12) shows that the whole information about overlapping ones between columns in $\mathbf{H}$ is actually represented by a subset of all the possible values of $\gamma_{i,j}$. This means that the execution of Algorithm 5 can be sped up by taking the QC structure into account.

In particular, the values of $\gamma_{i,j}$ can be used to obtain the distance spectra of the blocks in $\mathbf{H}$ in a straightforward way. Let us refer to (6.12), and look at two columns $\mathbf{h}_{pi_p}$ and $\mathbf{h}_j$, with $j = pi_p + j'$, where $j' \in [0, 1, \ldots, p-1]$. The support of $\mathbf{h}_{pi_p}$ is $\phi(\mathbf{h}_{pi_p}) = \{c_0^{(pi_p)}, \cdots, c_{v-1}^{(pi_p)}\}$. The support of $\mathbf{h}_j$ can be expressed as

$$\mathrm{S}(\mathbf{h}_j) = \{c_l^{(j)} \,\big|\, c_l^{(j)} = c_l^{(pi_p)} + j' \mod p, \;\; l \in [0, \ldots, v-1], \;\; c_l^{(pi_p)} \in \mathrm{S}(\mathbf{h}_{pi_p})\}.$$

Then, we have $\left|\mathrm{S}(\mathbf{h}_{pi_p}) \cap \mathrm{S}(\mathbf{h}_j)\right| = \gamma_{pi_p,j}$; this means that there are $\gamma_{pi_p,j}$ pairs

$\{c, c'\} \in \mathrm{S}(\mathbf{h}_{pi_p}) \times \mathrm{S}(\mathbf{h}_j)$ such that

$$c' = c + d \mod p, \;\; d \in \{j', p - j'\}. \tag{6.13}$$

It is easy to see that (6.13) corresponds to the definition of the distance spectrum of the blocks in $\mathbf{H}$; then, (6.13) can be turned into the following rule

$$\left|\mathrm{S}(\mathbf{h}_{pi_p}) \cap \mathrm{S}(\mathbf{h}_{pi_p+j'})\right| = \gamma \leftrightarrow d \in \mathrm{DS}(\mathbf{H}_{i_p}), \;\; \mu_d = \gamma,$$

with $d = \min\{\pm j' \mod p\}$.
This proves that the procedure described by Algorithm 5 allows obtaining at least the same amount of information recovered through the Ex-GJS algorithm, which is specific to the QC case and guarantees a complete cryptanalysis of the system [64]. In other words, our analysis confirms that Algorithm 5 is applicable and successful in at least all the scenarios in which the attack from [64] works. Moreover, our procedure allows for recovering a larger amount of information about the secret key, and thus defines a broader perimeter of information retrieval, which encompasses existing and future attacks.

One final remark is about the kind of information that can be used to mount statistical attacks. As already observed, events of decryption failure and the number of decoding iterations leak information about the secret and, thus, can be used to mount such attacks. However, due to the intrinsically probabilistic nature of the decoder, there are other quantitites that may be considered by an adversary. In [62] Santini et al. have shown that, for instance, the number of performed flips in the first iteration, as well as the syndrome weight after the first iteration, can be used to successfully perform statistical attacks. It is very likely that these results can be extended to i) other quantitities, and ii) all decoder iterations.

6.5 Countering statistical attacks

In this section, which concludes this chapter, we propose a methodology to design LEDA instances which can withstand statistical attacks. The results we describe are a brief summary of the work in [66].

6.5.1 Ephemeral keys

As we have already stated, a variant of LEDAkem uses ephemeral keys, i.e., employs key-pairs that are used for only one key encapsulation/decapsulation. This choice completely thwarts statistical attacks, since it removes the possibility of collecting a sufficiently large amount of information about the secret key. Thus, even non negligible DFR values do not represent a security issue.

Table 6.1: Parameter sizes for LEDAkem instances with ephemeral keys.

λ	n_0	p	t	d_v	m	errors out of decodes
128	2	$14,939$	136	11	$[4,3]$	14 out of $1.2 \cdot 10^9$
	3	$7,853$	86	9	$[4,3,2]$	0 out of $1 \cdot 10^9$
	4	$7,547$	69	13	$[2,2,2,1]$	0 out of $1 \cdot 10^9$
192	2	$25,693$	199	13	$[5,3]$	2 out of $1 \cdot 10^9$
	3	$16,067$	127	11	$[4,4,3]$	0 out of $1 \cdot 10^9$
	4	$14,341$	101	15	$[3,2,2,2]$	0 out of $1 \cdot 10^9$
256	2	$36,877$	267	11	$[7,6]$	0 out of $1 \cdot 10^9$
	3	$27,437$	169	15	$[4,4,3]$	0 out of $1 \cdot 10^9$
	4	$22,691$	134	13	$[4,3,3,3]$	0 out of $1 \cdot 10^9$

Then, by taking into account all the ISD attacks we have described in the previous sections, secure LEDAkem parameters can be devised. In Tab. 6.5.1 we propose instances for security parameter $\lambda \in \{128, 192, 256\}$ and number of circulant blocks $n_0 \in \{2, 3, 4\}$. The decoder that we propose for such instances is the Q-decoder in which, at each iteration, the threshold is chosen through the rule described in Section 6.2.1. The DFR of these instances has been estimated through numerical simulations and, for the proposed instances, is in worst case in the order of 10^{-9}.

In Tables 6.2 6.3 we show key sizes and running times with the actual implementation of LEDAkem with ephemeral keys, which can downloaded from the official website [20], for the instances proposed in Table 6.5.1. As we see from the Table, the parameter n_0 can be adjusted to modify the trade-off between key size and running time. For instance, $n_0 = 2$ is the optimal choice for the public key size, while choosing $n_0 = 3$ leads to improvements in the scheme running time.

Table 6.2: LEDAkem with ephemeral keys – Sizes in bytes of the key pair (at rest and in memory), of the encapsulated secret and of the shared secret, as a function of the security parameter and the number of circulant blocks n_0

Security Parameter	$\mathbf{n_0}$	**Private key (B)**		**Public key (B)**	**Encapsulated secret size (B)**	**Shared secret size (B)**
		At rest	**In memory**			
128	2	24	452	1,872	1,872	32
	3	24	540	2,080	1,040	32
	4	24	684	2,832	944	32
192	2	32	644	3,216	3,216	48
	3	32	748	4,032	2,016	48
	4	32	924	5,400	1,800	48
256	2	40	764	4,616	4,616	64
	3	40	972	6,864	3,432	64
	4	40	1,092	8,520	2,840	64

Table 6.3: LEDAkem with ephemeral keys – Running times for key generation, key encapsulation and key decapsulation, followed by the total time needed for a key exchange without considering transmission times, as a function of the security parameter and the number of circulant blocks n_0, on an Intel Skylake i5-6600 at 3.6 GHz.
The figures are taken employing the completely portable reference implementation in ISO C11, compiled with `GCC` 6.3.0, employing `-march=native -O3` as optimization parameters

Security Parameter	n_0	KeyGen (ms)	Encap. (ms)	Decap. (ms)	Total exec. time (ms)
128	2	1.374 (± 0.130)	0.046 (± 0.009)	0.340 (± 0.072)	1.759
	3	0.569 (± 0.089)	0.038 (± 0.012)	0.424 (± 0.057)	1.031
	4	0.884 (± 0.510)	0.043 (± 0.006)	1.305 (± 0.136)	2.233
192	2	3.725 (± 0.188)	0.092 (± 0.018)	0.950 (± 0.103)	4.768
	3	1.793 (± 0.271)	0.088 (± 0.023)	1.112 (± 0.075)	2.992
	4	2.759 (± 1.170)	0.112 (± 0.014)	2.065 (± 0.176)	4.936
256	2	7.644 (± 0.261)	0.176 (± 0.022)	1.276 (± 0.105)	9.095
	3	4.964 (± 0.602)	0.177 (± 0.013)	1.623 (± 0.122)	6.764
	4	5.649 (± 1.846)	0.217 (± 0.018)	2.752 (± 0.176)	8.618

6.6 Long term keys

When we want to use long term keys, we necessarily have to deal with statistical attacks. As we have already said, this requires i) a constant time and power implementation, and ii) negligible DFR values. In this section we describe a methodology to design a 2-iterations Q-decoder with negligible failure rate. Our approach is based on the following considerations:

i) when $\mathbf{L}$ has full weight, then a BF-decoder operating in $\mathbf{L}$ is equivalent to a Q-decoder operating on $\mathbf{H}$ and $\mathbf{Q}$; this property can be easily proven by repeating the computations we have shown in Section 6.2.2 and by assuming that

$$\mathbf{H} * \mathbf{Q} = \mathbf{HQ},$$

i.e., that the integer product between $\mathbf{H}$ and $\mathbf{Q}$ returns a matrix with entries in $\{0; 1\}$;

ii) adopting the results of section 4.1.2, we can compute the number of errors that can be corrected by a single iteration of BF decoder operating on $\mathbf{L}$. Because of the aforementioned equivalence, this number corresponds to the amount of errors that can be corrected by a single iteration of Q-decoder operating on $\mathbf{H}$ and $\mathbf{Q}$.

Let $\bar{t}$ denote the error correction capability of a particular secret key sk. Let $\mathbf{s} = \mathbf{L}\mathbf{e}^\top$ be the input syndrome, with $\mathrm{wt}(\mathbf{e}) = t$; furthermore, let $\hat{\mathbf{e}}$ denote the error vector estimate after the first Q-decoder iteration. We define the number of residual errors as $t' = \mathrm{wt}(\mathbf{e} + \hat{\mathbf{e}})$. When the $t' \leq \bar{t}$, then the second iteration will always correct this amount of residual errors and will output a vector identical to $\mathbf{e}$. When $t' > \bar{t}$, there is some probability of failure which, however, can be hardly be estimated through theoretical arguments. Thus, we conservatively assume that $t' > \bar{t}$ always results in a decoding failure; this assumption allows us to derive a simple upper bound on this two iterations decoder as

$$\mathrm{DFR} \leq 1 - \Pr\left[t' \leq \bar{t}\right]. \tag{6.14}$$

We now describe how the distribution of the number of residual errors t' can be predicted; we denote with b the employed decoding threshold in the first iteration, and denote the vertical weight of $\mathbf{L}$ as $v = md_v$.

We use N_{ic} to denote the number of bits that are affected by errors and get flipped in the first iteration of the decoder, while N_{ci} denotes the number of bits that, on the contrary, are error free and get flipped. For the sake of simplicity, we refer to N_{ic} as the number of *right flips* and to N_{ci} as the number of *wrong flips*. It is clear that $t' = N_{ic} + N_{ci}$. We recall the following probabilities

$$\rho_{cc} = \sum_{j\,=\,0,\,j\text{ even}}^{\min[w-1,t]} \frac{\binom{w-1}{j}\binom{n-w}{t-j}}{\binom{n-1}{t}},$$
$$\rho_{ic} = \sum_{j\,=\,0,\,j\text{ even}}^{\min[w-1,t-1]} \frac{\binom{w-1}{j}\binom{n-w}{t-j-1}}{\binom{n-1}{t-1}},$$

where ρ_{cc} is the probability that a bit is error free and a parity-check equation involving it evaluates it correctly, that is, it is satisfied, whereas ρ_{ic} is the probability that a bit is in error and a parity-check equation involving it evaluates it correctly, that is, it is unsatisfied. As in [15, Section 4.2], [49, Section 3], we assume that in the first iteration the parity-check equations are not correlated. An erred bit, say the i-th, is flipped and, eventually, correctly evaluated if its counter $\sigma_i^{(1)} \geq b$, where b denotes the employed decoding threshold. which happens with probability

$$\mathrm{Pr}_{ic} = \sum_{j=b_0}^{v} \binom{v}{j} \rho_{ic}^j \left(1 - \rho_{ic}\right)^{v-j}.$$

An error free bit, say the j-th, is wrongly estimated if $\sigma_j^{(0)} \geq b$, which happens with probability

$$\mathrm{Pr}_{ci} = \sum_{j=b_0}^{v} \binom{v}{j} \left(1 - \rho_{cc}\right)^j \rho_{cc}^{v-j}.$$

In other words, Pr_{ic} is the probability of a right flip, while Pr_{ci} is the probability of a wrong flip. Hence we have

$$\Pr\left[N_{ic} = z\right] = \binom{t}{z} P_{ic}^{z} \left(1 - P_{ic}\right)^{t-z},$$

$$\Pr\left[N_{ci} = u\right] = \binom{n-t}{u} P_{ci}^{u} \left(1 - P_{ci}\right)^{n-t-u}.$$

Then, the probability that $\mathbf{e}'$ has weight equal to a given value t' can be obtained as follows

$$\Pr\left[\mathrm{wt}(\mathbf{e}') = t' \mid \mathrm{wt}(\mathbf{e}) = t\right] = \sum_{x=0}^{t'} \Pr\left[N_{ic} = t' - x\right] \Pr\left[N_{ci} = x\right].$$

Then, Eq. (6.14) can be rewritten as

$$\begin{aligned} \mathrm{DFR} &\leq 1 - \Pr\left[\mathrm{wt}\left(e'\right) \leq \bar{t}\right] \\ &= 1 - \sum_{t'=0}^{\bar{t}} \Pr\left[\mathrm{wt}\left(e'\right) = t'\right] \\ &= 1 - \sum_{t'=0}^{\bar{t}} \sum_{x=0}^{t'} \Pr\left[N_{ic} = t' - x\right] \Pr\left[N_{ci} = x\right]. \end{aligned}$$

Using this method, the DFR on this two iterations simple decoder can be upper bounded.

In particular, this method can be used to design LEDA instances. Indeed, given n_0, d_v, m, t and a target DFR ϵ, it is enough to choose $\bar{t}$ and p such that

$$\Pr\left[\mathrm{wt}\left(e'\right) \leq \bar{t}\right] > 1 - \epsilon.$$

To guarantee the desired DFR, we propose to test the error correction capability of each generated key: if it is lower than $\bar{t}$, then the key gets discarded and a new one is tested. With this simple additional rejection sampling step, we can ensure that the selected keys actually yields the desired DFR. In Table 6.4 we report some instances achieving different DFR values. We additionally report the number of key that we have rejected in our experiments.

In Tables 6.5 and 6.6 we have reported key sizes and running times for the LEDApkc instances proposed in Table 6.4.

Table 6.4: Parameters for the LEDAkem and LEDApkc employing a two-iteration Q-decoder matching a DFR equal to 2^{-64} and a DFR equal to $2^{-\lambda}$, where λ equals 128, 192, 256.

λ	n_0	DFR	$\mathbf{p}$	$\mathbf{t}$	$\mathbf{d_v}$	$\mathbf{m}$	$\bar{\mathbf{t}}$	**No. of keys out of 100 providing the guaranteed DFR**	$\mathbf{b}$
128	2	2^{-64}	35,899	136	9	[5, 4]	4	95	44
	2	2^{-128}	52,147	136	9	[5, 4]	4	95	43
192	2	2^{-64}	57,899	199	11	[6, 5]	5	92	64
	2	2^{-192}	96,221	199	11	[6, 5]	5	92	64
256	2	2^{-64}	89,051	267	13	[7, 6]	6	93	89
	2	2^{-256}	152,267	267	13	[7, 6]	6	93	88

Table 6.5: LEDApkc – Sizes in bytes of the key pair, and the minimum and maximum ciphertext expansion overhead, as a function of the security parameter and of the decryption failure rate provided by the choice of the parameters of the underlying QC-LDPC code

Security parameter	$\mathbf{n_0}$	**DFR**	**Private key (B)**		**Public key (B)**	**Min. ciphertext overhead (B)**	**Max. ciphertext overhead (B)**
			At rest	**In memory**			
128	2	2^{-64}	25	468	4,488	4,521	8,976
	2	2^{-128}	25	468	6,520	6,554	13,040
192	2	2^{-64}	33	660	7,240	7,283	14,480
	2	2^{-192}	33	660	12,032	12,077	24,064
256	2	2^{-64}	41	884	11,136	11,189	22,272
	2	2^{-256}	41	884	19,040	19,095	38,080

Table 6.6: LEDApkc – Running times for key generation, encryption and decryption assuming a plaintext message to be encrypted with size 1KiB.
Execution times on an Intel Skylake i5-6600 at 3.6 GHz are reported as a function of the NIST category and of the decryption failure rate provided by the choice of the parameters of the underlying QC-LDPC code.
The figures are taken employing the completely portable reference implementation in ISO C11, compiled with `GCC` 6.3.0, employing `-march=native -O3` as optimization parameters

Security parameter	n_0	DFR	KeyGen (s)	Encryption (ms)	Decryption (ms)
128	2	2^{-64}	0.290 ($\pm$ 0.008)	0.29 ($\pm$ 0.00)	0.76 ($\pm$ 0.00)
	2	2^{-128}	0.422 ($\pm$ 0.014)	0.42 ($\pm$ 0.03)	1.18 ($\pm$ 0.12)
192	2	2^{-64}	1.187 ($\pm$ 0.483)	0.56 ($\pm$ 0.11)	1.70 ($\pm$ 0.21)
	2	2^{-192}	1.538 ($\pm$ 0.043)	1.10 ($\pm$ 0.11)	2.39 ($\pm$ 0.07)
256	2	2^{-64}	2.543 ($\pm$ 0.037)	1.02 ($\pm$ 0.09)	3.26 ($\pm$ 0.44)
	2	2^{-256}	4.240 ($\pm$ 0.069)	1.53 ($\pm$ 0.07)	4.16 ($\pm$ 0.09)

Chapter 7

Conclusions

In this thesis we have studied post-quantum public-key cryptosystems based on hard problems arising from coding theory arguments. We have described classical solutions based on Goppa codes, and have introduced a new scheme based on Generalized Reed-Solomon codes. This new cryptosystem, which we have called BCRSS, is a modification of the BBCRS scheme, which comes as a countermeasure to a recent cryptanalysis that can efficiently recover the secret code. We have described how a proper parameters choice, together with a little tweak to avoid high decryption complexity, is enough to thwart such an attack. We have compared the performances of the BCRSS with those of competing schemes based on algebraic codes; our results show that this new cryptosystem represents a valid alternative to these solutions.

We have then considered the family of Low-Density Parity-Check (LDPC) codes which, differently from algebraic codes, can efficiently be decoded only with techniques that are intrinsically characterized by some failure probability. We have then addressed the open problem of deriving reliable and easy-to-obtain estimations for the Decoding Failure Rate (DFR) of such codes. In particular, we have analyzed one iteration of the simple Bit Flipping (BF) decoder, and have derived precise and theoretical bound on its error correction capability. We have proposed a method to estimate the number of errors which can be corrected by a single decoder iteration; this methodology takes into account the parity-check matrix structure and, through the additional decoding threshold optimization, improves upon known bounds on the error correction capability. We have then described how, with analogous tools, a mathematical upper bound on the DFR of a single iteration can be derived. The obtained bound does not require any assumption, is easy to compute and is sufficiently tight. We have specialized the general expression of the bound to Quasi-Cyclic (QC) codes and to codes with girth ≥ 6, and have justified its tightness by means of numerical simulations. Despite this approach takes into account only a single iteration, we believe that it may serve as the first step to derive reliable and assumption-free bounds which take into account more than one iteration.

We have introduced the notion of reproducible codes, i.e., that of codes admitting a compact representation, which encompasses well known families of codes, such

as those of QC and Quasi-Dyadic. We have derived conditions on the existence of such codes, which can be further specialized to the interesting case of reproducibility obtained via permutations. We have provided examples of codes constructions, and have described how to instantiate code-based cryptosystems based on such families of codes.

Finally, we have described LEDAcrypt, a suite of crytographic algorithms based on QC-LDPC codes which is currently under evaluation for standardization in the ongoing NIST competition. We have provided a complete security analysis of the scheme, by taking into account classical attacks, such as those based on Information Set Decoding, and the recent family of statistical attacks. In particular, we have described a general methodology to mount statistical attacks, and have shown how the proposed method outperforms existing techniques. Our results confirm that, for cryptosystems based on QC-LDPC and QC-MDPC codes, constant time and power implementations, with negligible decryption failure rate, are a necessity. We have then described a way to upper bound the DFR in LEDA cryptosystems, and have used this method to design practical LEDA instances.

Bibliography

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[2] Lily Chen, Yi-Kai Liu, Stephen Jordan, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone, "Report on post-quantum cryptography", Tech. Rep. NISTIR 8105, National Institute of Standards and Technology, 2016.

[3] National Institute of Standards and Technology, "Post-quantum crypto project", December 2016.

[4] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory", *Deep Space Network Progress Report*, vol. 44, pp. 114–116, January 1978.

[5] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems", vol. 24, no. 3, pp. 384–386, May 1978.

[6] Daniel J. Bernstein, *Grover vs. McEliece*, pp. 73–80, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] Rafael Misoczki and Paulo S. L. M. Barreto, "Compact McEliece keys from Goppa codes", in *Selected Areas in Cryptography*, vol. 5867 of *Lecture Notes in Computer Science*, pp. 376–392. Springer Verlag, 2009.

[8] P. Gaborit, "Shorter keys for code based cryptography", in *Proc. Int. Workshop on Coding and Cryptography (WCC 2005)*, Bergen, Norway, March 2005, pp. 81–90.

[9] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich, "Algebraic cryptanalysis of McEliece variants with compact keys", in *EUROCRYPT*, H. Gilbert, Ed. 2010, vol. 6110 of *Lecture Notes in Computer Science*, pp. 279–298, Springer.

[10] 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)*, Sorrento, Italy, 2000, p. 215, IEEE.

[11] 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)*, june 2007, pp. 951–956.

[12] 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)*, June 2007, pp. 2591–2595.

[13] 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)*, Berlin, Heidelberg, 2008, pp. 246–262, Springer-Verlag.

[14] 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*, Istambul, Turkey, 2013, pp. 2069–2073, IEEE.

[15] R. G. Gallager, *Low-Density Parity-Check Codes*, PhD thesis, M.I.T., 1963.

[16] Qian Guo, Thomas Johansson, and Paul Stankovski, *A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors*, pp. 789–815, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[17] Edward Eaton, Matthieu Lequesne, Alex Parent, and Nicolas Sendrier, "QC-MDPC: A timing attack and a CCA2 KEM", in *PQCrypto*, Tanja Lange and Rainer Steinwandt, Eds., Fort Lauderdale, FL, USA, April 2018, pp. 47–76, Springer International Publishing.

[18] Tomáš Fabšič, Viliam Hromada, Paul Stankovski, Pavol Zajac, Qian Guo, and Thomas Johansson, "A reaction attack on the QC-LDPC McEliece cryptosystem", in *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017*, Tanja Lange and Tsuyoshi Takagi, Eds., pp. 51–68. Springer, Utrecht, The Netherlands, June 2017.

[19] Tomás Fabšič, Viliam Hromada, and Pavol Zajac, "A reaction attack on LEDApkc", *IACR Cryptology ePrint Archive*, vol. 2018, pp. 140, 2018.

[20] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini, "LEDAkem: Low dEnsity coDe-bAsed key encapsulation mechanism", 2017.

[21] A. Vardy, "The intractability of computing the minimum distance of a code", *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1757–1766, Nov 1997.

[22] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory", *Problems of Control and Information Theory*, vol. 15, no. 2, pp. 159–166, 1986.

[23] Jean-Charles Faugere, Valérie Gauthier-Umana, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich, "A distinguisher for high-rate McEliece cryptosystems", *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6830–6844, 2013.

[24] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Ruben Niederhagen, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, and Wen Wang, "*https://classic.mceliece.org/*".

[25] E. Prange, "The use of information sets in decoding cyclic codes", *IRE Transactions*, vol. IT-8, pp. S5–S9, 1962.

[26] P. J. Lee and E. F. Brickell, "An observation on the security of mceliece's public-key cryptosystem", in *Advances in Cryptology — EUROCRYPT '88*, D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, Eds., Berlin, Heidelberg, 1988, pp. 275–280, Springer Berlin Heidelberg.

[27] J. S. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes", *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1354–1359, Sep. 1988.

[28] J. Stern, "A method for finding codewords of small weight", in *Coding Theory and Applications*, G. Cohen and J. Wolfmann, Eds., vol. 388 of *Lecture Notes in Computer Science*, pp. 106–113. Springer Verlag, 1989.

[29] Matthieu Finiasz and Nicolas Sendrier, "Security bounds for the design of code-based cryptosystems", in *Advances in Cryptology – ASIACRYPT 2009*, Mitsuru Matsui, Ed., Berlin, Heidelberg, 2009, pp. 88–105, Springer Berlin Heidelberg.

[30] Alexander May, Alexander Meurer, and Enrico Thomae, "Decoding random linear codes in $\tilde{O}(2^{0.054n})$", in *Advances in Cryptology – ASIACRYPT 2011*, Dong Hoon Lee and Xiaoyun Wang, Eds., Berlin, Heidelberg, 2011, pp. 107–124, Springer Berlin Heidelberg.

[31] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer, "Decoding random binary linear codes in $2^{n/20}$: How 1+1=0 improves information set decoding", in *Advances in Cryptology - EUROCRYPT 2012*, D. Pointcheval and T. Johansson, Eds. 2012, vol. 7237 of *LNCS*, pp. 520–536, Springer.

[32] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini, "A finite regime analysis of information set decoding algorithms", *Algorithms*, vol. 12, no. 10, 2019.

[33] Rodolfo Canto Torres and Nicolas Sendrier, "Analysis of information set decoding for a sub-linear error weight", in *PQCrypto*, 2016.

[34] Christiane Peters, "Information-set decoding for linear codes over fq", in *Post-Quantum Cryptography*, Nicolas Sendrier, Ed., Berlin, Heidelberg, 2010, pp. 81–94, Springer Berlin Heidelberg.

[35] Lov K. Grover, "A fast quantum mechanical algorithm for database search", in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 1996, STOC '96, pp. 212–219, ACM.

[36] D.J. Bernstein, T. Lange, C.P. Peters, and H.C.A. Tilborg, van, "Explicit bounds for generic decoding algorithms for code-based cryptography", in *International Workshop on Coding and Cryptography (WCC 2009, Ullensvang, Norway, May 10-15, 2009. Pre-proceedings)*. 2009, pp. 168–180, Selmer Center, University of Bergen.

[37] M. Baldi, F. Chiaraluce, J. Rosenthal, P. Santini, and D. Schipani, "Security of generalised reed–solomon code-based cryptosystems", *IET Information Security*, vol. 13, no. 4, pp. 404–410, 2019.

[38] V. M. SIDELNIKOV and S. O. SHESTAKOV, "On insecurity of cryptosystems based on generalized reed-solomon codes", *Discrete Mathematics and Applications*, vol. 2, no. 4, 1992.

[39] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani, "Enhanced public key security for the mceliece cryptosystem", *Journal of Cryptology*, vol. 29, no. 1, pp. 1–27, Jan 2016.

[40] Alain Couvreur, Ayoub Otmani, Jean-Pierre Tillich, and Valérie Gauthier-Umaña, "A polynomial-time attack on the bbcrs scheme", in *Public-Key Cryptography – PKC 2015*, Jonathan Katz, Ed., Berlin, Heidelberg, 2015, pp. 175–193, Springer Berlin Heidelberg.

[41] Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umaña, Ayoub Otmani, and Jean-Pierre Tillich, "Distinguisher-based attacks on public-key cryptosystems using reed–solomon codes", *Designs, Codes and Cryptography*, vol. 73, no. 2, pp. 641–666, Nov 2014.

[42] Thierry P. Berger and Pierre Loidreau, "How to mask the structure of codes for a cryptographic use", *Designs, Codes and Cryptography*, vol. 35, pp. 63–79, 2005.

[43] Christian Wieschebrink, "Cryptanalysis of the niederreiter public key scheme based on grs subcodes", in *Post-Quantum Cryptography*, Nicolas Sendrier, Ed., Berlin, Heidelberg, 2010, pp. 61–72, Springer Berlin Heidelberg.

[44] Irene Márquez-Corbella, Edgar Martínez-Moro, and Ruud Pellikaan, "The non-gap sequence of a subcode of a generalized reed–solomon code", *Designs, Codes and Cryptography*, vol. 66, no. 1, pp. 317–333, Jan 2013.

[45] A. Couvreur, I. Márquez-Corbella, and R. Pellikaan, "Cryptanalysis of mceliece cryptosystem based on algebraic geometry codes and their subcodes", *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 5404–5418, Aug 2017.

[46] Jessalyn Bolkema, Heide Gluesing-Luerssen, Christine Kelley, Kristin Lauter, Beth Malmskog, and Joachim Rosenthal, *Variations of the McEliece Cryptosystem*, pp. 129–150, 11 2017.

[47] Violetta Weger, "A Code-Based Cryptosystem using GRS Codes".

[48] S. K. Chilappagari, D. V. Nguyen, B. Vasic, and M. W. Marcellin, "On the guaranteed error correction capability of LDPC codes", in *Proc. IEEE International Symposium on Information Theory (ISIT 2008)*, Toronto, Canada, July 2008, pp. 434–438.

[49] Jean-Pierre Tillich, "The decoding failure probability of mdpc codes", 06 2018, pp. 941–945.

[50] P. Santini, M. Battaglioni, M. Baldi, and F. Chiaraluce, "Hard-decision iterative decoding of ldpc codes with bounded error rate", in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.

[51] Paolo Santini, Massimo Battaglioni, Marco Baldi, and Franco Chiaraluce, "A theoretical analysis of the error correction capability of ldpc and mdpc codes under parallel bit-flipping decoding", 2019.

[52] Paolo Santini, Edoardo Persichetti, and Marco Baldi, "Reproducible codes and cryptographic applications", Cryptology ePrint Archive, Report 2018/666, 2018, `https://eprint.iacr.org/2018/666`.

[53] Nicolas Aragon, Paulo S. L. M. Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor, "BIKE: Bit flipping key encapsulation", 2017.

[54] Alston S. Householder, "Unitary triangularization of a nonsymmetric matrix", *J. ACM*, vol. 5, pp. 339–342, 1958.

[55] N. Sendrier, "Decoding one out of many", in *PQCrypto 2011*, B.-Y. Yang, Ed. 2011, vol. 7071 of *LNCS*, pp. 51–67, Springer.

[56] Rodolfo Canto Torres and Nicolas Sendrier, "Analysis of information set decoding for a sub-linear error weight", in *PQCrypto 2016*, Tsuyoshi Takagi, Ed. 2016, vol. 9606 of *LNCS*, pp. 144–161, Springer.

[57] Mohammad Aref, Masoumeh Shooshtari, Thomas Johansson, and Mahmoud Ahmadian Attari, "Cryptanalysis of mceliece cryptosystem variants based on quasi-cyclic low-density parity check codes", *IET Information Security*, vol. 10, 01 2015.

[58] Kazukuni Kobara and Hideki Imai, "Semantically secure McEliece public-key cryptosystems — conversions for McEliece PKC", *Lecture Notes in Computer Science*, vol. 1992, pp. 19–35, 2001.

[59] Julia Chaulet and Nicolas Sendrier, "Worst case qc-mdpc decoder for mceliece cryptosystem", 07 2016, pp. 1366–1370.

[60] Marco Baldi, Marco Bianchi, and Franco Chiaraluce, "Security and complexity of the McEliece cryptosystem based on QC-LDPC codes", *IET Inf. Security*, vol. 7, no. 3, pp. 212–220, September 2012.

[61] Thales Paiva and Routo Terada, "Improving the efficiency of a reaction attack on the QC-MDPC McEliece", *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E101.A, pp. 1676–1686, Oct 2018.

[62] Paolo Santini, Massimo Battaglioni, Franco Chiaraluce, and Marco Baldi, "Analysis of reaction and timing attacks against cryptosystems based on sparse parity-check codes", in *Code-Based Cryptography*, Marco Baldi, Edoardo Persichetti, and Paolo Santini, Eds., Cham, 2019, pp. 115–136, Springer International Publishing.

[63] Paolo Santini, Marco Baldi, and Franco Chiaraluce, "Assessing and countering reaction attacks against post-quantum public-key cryptosystems based on QC-LDPC codes", in *Cryptology and Network Security - 17th International Conference, CANS 2018, Naples, Italy, September 30 - October 3, 2018, Proceedings*, 2018, pp. 323–343.

[64] Qian Guo, Thomas Johansson, and Paul Stankovski, "A key recovery attack on MDPC with CCA security using decoding errors", in *ASIACRYPT 2016*, Jung Hee Cheon and Tsuyoshi Takagi, Eds., vol. 10031 of *LNCS*, pp. 789–815. Springer Berlin Heidelberg, 2016.

[65] Alexander Nilsson, Thomas Johansson, and Paul Stankovski, "Error amplification in code-based cryptography", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 238–258, Nov. 2018.

[66] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini, "Ledacrypt: Qc-ldpc code-based cryptosystems with bounded decryption failure rate", in *Code-Based Cryptography*, Marco Baldi, Edoardo Persichetti, and Paolo Santini, Eds., Cham, 2019, pp. 11–43, Springer International Publishing.

Appendix A

In this Appendix we describe an efficient way to compute the cardinalities of the sets introduced in Definition 19. To this end, we first formalize the problem and then describe a method that, for the cases we are interested in, significantly improves upon the naive exhaustive search approach.

Problem 4 *Let* $\mathbf{a} \in \mathbb{N}^l$ *be a length-*l *vector of non negative integers, and let* $B \subseteq [0; l-1]$ *be a set of size* $m \leq l$*. Given* $\alpha \in \mathbb{N}$*,* $\alpha > 0$*, compute*

$$N_B = \left| \left\{ B \subseteq [0; l-1], \;\; |B| = m \;\; \textit{s.t.} \;\; \sum_{i \in B} a_i > \alpha \right\} \right|.$$

It is clear that an exhaustive search would require to generate all subsets of size m: thus, the corresponding complexity will be equal to $\binom{l}{m}$. As we show with combinatorial arguments, a simple algorithm can be devised, with a complexity that may be significantly lower.

In particular, we obtain the number of sets that are complementary to those defined in Problem 4, that is,

$$\bar{N}_B = \left| \left\{ B \subseteq [0; l-1], \;\; |B| = m \;\; \text{s.t.} \;\; \sum_{i \in B} a_j \leq \alpha \right\} \right|,$$

from which the value of N_B can be straightforwardly obtained as

$$N_B = \binom{l}{m} - \bar{N}_B.$$

For a set B, we denote with $\mathbf{a}^{(B)}$ the vector formed by the entries of $\mathbf{a}$ that are indexed by B; we define $\bar{N}_B^{(j)}$ as the number of subsets B for which the corresponding subvector $\mathbf{a}^{(B)}$ contains m elements, j of which are distinct, whose sum is smaller than or equal to α. We have

$$\bar{N}_B = \sum_{j=1}^{l} \bar{N}_B^{(j)}.$$

The values of $\bar{N}_B^{(j)}$ can be easily obtained, as we show next.

First of all, let ω be the number of distinct values in $\mathbf{a}$, with $Y = \{y_0, y_1, \cdots, y_{\omega-1}\}$ being the set of such values in ascending order. In the same way, we define $\lambda_u =$

$|\{i \text{ s.t. } a_i = y_u\}|$. As we show below, the computation of $\bar{N}_B$ depends only on these quantities.

Let Y_B be the set of distinct values that are contained in $\mathbf{a}^{(B)}$. When $j = 1$, we easily have

$$\bar{N}_B^{(1)} = \sum_{0 \leq i \leq \omega - 1 \,:\, y_i \leq \left\lfloor \frac{\alpha}{m} \right\rfloor} \binom{\lambda_i}{m}, \tag{1}$$

where, as usual, $\binom{\lambda_i}{m} = 0$ if $m > \lambda_i$. When $j > 1$, some further considerations must be taken into account. For a set B, let $y_{i_0}, y_{i_1}, \cdots, y_{i_{j-1}}$ be the distinct values assumed by the entries of $\mathbf{a}^{(B)}$, and denote the corresponding multiplicities as $m_0, m_1, \cdots, m_{j-1}$. If $B \in \bar{N}_B^{(j)}$, we must have

$$\sum_{u=0}^{j-1} m_u y_{i_u} \leq \alpha. \tag{2}$$

We clearly have $m = \sum_{u=0}^{j-1} m_u$, from which we obtain $m_0 = m - \sum_{u=1}^{j-1} m_u$; then, (2) can be rewritten as

$$m y_{i_0} + \sum_{u=1}^{j-1} m_u (y_{i_u} - y_{i_0}) \leq \alpha.$$

It is obvious that

$$m y_{i_0} + \sum_{u=1}^{j-1} m_u (y_{i_u} - y_{i_0}) \geq m y_{i_0} + \sum_{u=1}^{j-1} (y_{i_u} - y_{i_0}).$$

The above condition can be turned into the following criterion: a set B associated to the values $y_{i_0}, y_{i_1}, \cdots, y_{i_{j-1}}$ of $\mathbf{a}^{(B)}$, whose sum is smaller than or equal to α, exists if and only if

$$\sum_{u=1}^{j-1} y_{i_u} - y_{i_0} \leq \alpha - m y_{i_0}.$$

Let us now fix an index $q \in [1; j-2]$, and suppose that we are looking at all sets B such that $\mathbf{a}^{(B)}$ contains the values $y_{i_0}, \cdots, y_{i_{q-1}}$ with respective multiplicities $m_1, m_2, \cdots, m_{q-1}$. Then, imposing the constraint and summing over all subsets, we obtain

$$\begin{aligned} \alpha &\geq m y_{i_0} + \sum_{u=1}^{q-1} m_u (y_{i_u} - y_{i_0}) + m_q (y_{i_q} - y_{i_0}) + \sum_{z=q+1}^{j-1} m_z (y_{i_z} - y_{i_0}) \\ &\geq m y_{i_0} + \sum_{u=1}^{q-1} m_u (y_{i_u} - y_{i_0}) + m_q (y_{i_q} - y_{i_0}) + \sum_{z=q+1}^{j-1} (y_{i_z} - y_{i_0}). \end{aligned}$$

Then, the maximum value for m_q is obtained as

$$m_q^{(\max)} = \min\left\{\lambda_q, \left\lfloor \frac{\alpha - m y_{i_0} - \sum_{u=1}^{q-1} m_u (y_{i_u} - y_{i_0}) - \sum_{z=q+1}^{j-1} (y_{i_z} - y_{i_0})}{y_{i_q} - y_{i_0}} \right\rfloor \right\}.$$

Finally, $\bar{N}_B^{(j)}$ can be computed as

$$\bar{N}_B^{(j)} = \sum_{i_0=0}^{\omega-j} \sum_{i_1=i_0+1}^{\omega-j+1} \cdots \sum_{i_{j-1}=i_{j-2}+1}^{\omega-1} d(i_0, \cdots, i_{j-1}),$$

where

$$d(i_0, \cdots, i_{j-1}) = \begin{cases} 0 & \text{if } \sum_{u=1}^{j-1} y_{i_u} - y_{i_0} > \alpha - m y_{i_0} \\ \sum_{m_1=1}^{m_1^{(\max)}} \cdots \sum_{m_{j-1}=1}^{m_{j-1}^{(\max)}} \binom{\lambda_{i_0}}{m - \sum_{i=1}^{j-1} m_i} \prod_{u=1}^{j-1} \binom{\lambda_{i_u}}{m_u} & \text{otherwise} \end{cases}, \tag{3}$$

where, coherently with (1), $\binom{\lambda_{i_0}}{m - \sum_{i=1}^{j-1} m_i} = 0$ if $\lambda_{i_0} < m - \sum_{i=1}^{j-1} m_i$.

We point out that, when $\mathbf{a}$ contains a small number of distinct elements (i.e., $\omega \ll l$), this approach becomes significantly faster than the exhaustive search on all subsets. Indeed, first of all we clearly have $\bar{N}_B^{(j)} = 0$ when $j > \omega$; moreover, the number of configurations tested by using (3) is surely smaller than m^{j-1}. Then, for a specific value of j, the computation of $\bar{N}_B^{(j)}$ requires to test no more than $m^{j-1}\binom{\omega}{j}$ configurations. Thus, we can roughly upper bound the total number of configurations that are considered as

$$\sum_{j=1}^{\omega} m^{j-1} \binom{\omega}{j} \leq \sum_{j=1}^{\omega} m^{j-1} \left(\frac{\omega e}{j}\right)^j \leq \omega m^{\omega-1} e^{\omega},$$

where e is the basis of the natural logarithmic. It can be verified that, when $m, w \ll l$, the above upper bound is significantly smaller than $\binom{l}{m}$.

Acknowledgments

Giunto alla fine di questo percorso, ci sono numerose persone che meritano i miei ringraziamenti. Questo periodo mi è servito per conoscere il mondo della ricerca, per apprenderne i lati positivi (e negativi) e, probabilmente, per capire ciò che davvero voglio fare nella vita. Dopo questi tre anni, credo che il mio bagaglio culturale si sia notevolmente arricchito; non di teoremi e teorie matematiche, ma di esperienze vissute, di luoghi visitati e di persone incontrate. Mi è sempre stata concessa parecchia libertà, sia per le tematiche da affrontare, sia nel viaggiare il più possibile, libertà che spero di aver ripagato adeguatamente.

Quello della ricerca è un settore fantastico, in cui regnano la completa libertà e la totale uguaglianza. Veniamo giudicati per la qualità dei nostri lavori, non per estrazione sociale, etnia o colore della pelle. Partecipiamo a conferenze in cui persone provenienti da ogni parte del mondo si scambiano idee durante il giorno e alla sera si siedono davanti ad una birra; tutto questo, parlando due lingue universali come l'inglese e la matematica. Con umiltá accettiamo i nostri sbagli ed il lavoro fatto da altri, anche quando magari smentiscono o migliorano i risultati da noi ottenuti. Ad una conferenza scientifica giovani studenti ascoltano affermati ricercatori, e viceversa, perchè nella vita non si finisce mai di imparare e, soprattutto, perchè ogni persona ha qualcosa da insegnare agli altri, chiunque essi siano.

Sarebbe bello se il mondo fosse come una grande conferenza, in cui tutti quanti abbiamo le stesse opportunità, in cui ognuno ha possibilità di parola e, soprattutto, in cui ci si ascolta a vicenda. Purtroppo, il mondo è ben distante da questo ideale (forse utopico) e sempre più spesso, magari aizzati dalle parole di qualche politico in cerca di consenso, tendiamo a comportarci da egoisti e da menefreghisti. È facile discriminare chi ha meno di noi, chi non ha la voce per difendersi; è facile respingere il diverso ed evitare il dialogo. È facile fare tutto ciò, ma non ci rendiamo conto delle opportunità e capacità, nostre e non solo, che stiamo sprecando.

Diceva Albert Einstein, o forse qualcuno prima di lui, che *"la mente è come un paracadute: funziona solo se si apre"*. Aprire la mente: in questi anni ho sicuramente imparato a farlo, e mai smetterò. Non perchè mi servirà per scrivere articoli scientifici, ma perchè penso sia la cosa più importante che noi tutti dobbiamo cercare di fare per rendere il mondo un posto migliore.

Ancona, November 2019

Paolo Santini