University of Zurich — UZH — Universität Zürich
Faculty of Science — MNF — Math.-naturwiss. Fakultät
Institute of Mathematics — I·Math — Institut für Mathematik

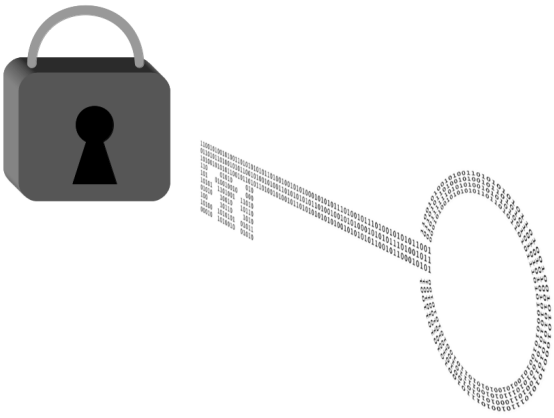
Master Thesis

McEliece-type cryptosystems:
costs, security and an attack to a recent variant

Andrea De Giorgi

under supervision of
Prof. Dr. Joachim Rosenthal

6th August 2012



Acknowledgements

I would like to express my sincerest gratitude to Prof. Dr. J. Rosenthal for having given me the possibility of writing my master thesis under his supervision, enabling me to broaden my knowledge in this very interesting topic.

A big thanks goes also to Felix Fontein, who kindly answered all my questions and helped me in programming with `Magma`.

Contents

Lists

List of Algorithms

List of Figures

List of Tables

Introduction

In our all day life we can experience at first hand the advantages that the modern technology gives us: since the invention of the phone, the telegraph and the radio we can talk, communicate, interact with someone located in almost any part of the world. The price to pay for having these comforts is the possible loss of security in the information exchange. Moreover, the continuous and inexorable technological development provides the means for intercepting information during communications from a user to another. Luckily, this same natural process should also enable the access to sufficient scientific knowledge allowing the prevention of such risks.

Elementary forms of cryptography were already known in the ancient times, but in this thesis we restrict our attention to the recent concept of *asymmetric* or *public-key cryptography*. Its strength relies in the fact that, differently from symmetric or private-key cryptography, the two parties do not have to exchange the encryption/decryption key to have some cryptosystem working. Though, in 1973, few years before the development of this new sector of cryptography, the British Government Communications Headquarters (GCHQ) discovered what we usually name Diffie-Hellman key exchange[1] but kept this algorithm secret for the sake of national interest.

The idea to use two different keys to encrypt and decrypt a plaintext was intensely exploited in the late Seventies, during which ciphers such as RSA, the Merkle-Hellman knapsack and the McEliece cryptosystem were discovered. In this work we will focus particularly on the latter. It was proposed by R. J. McEliece in 1978 [25] and has the particularity of being based on error-correcting codes: for encryption, a plaintext is multiplied by a matrix which is a modified generator matrix of some linear code, then some components of the obtained vector have to be corrupted. The receiver (i.e. the code designer) should then be able to recover the original message by treating the ciphertext as if it were a codeword which did not emerge unscathed from the transmission channel.

Few years later, H. Niederreiter [30] proposed another cryptosystem based on error-correcting codes. In this case, the ciphertext is a syndrome which the legal user has to convert into the plaintext by decoding. The two cryptosystems present many similarities, so that it is not infrequent that some authors try to apply the same idea to both cryptosystems. Many publications even treat the Niederreiter cipher as a particular case of McEliece, but ironically, in his own paper the Austrian mathematician does not even cite the work of his American colleague.

Apart from being both based on coding theory, an important point they share is that they have the same security level [22]. Sadly, in spite of the advantage of having fast encryption and decryption algorithms if compared for example to RSA [24, p. 588], the McEliece and

[1] See `http://www.gchq.gov.uk/History/Pages/PKE.aspx`.

the Niederreiter cryptosystems have an important drawback making their implementation unattractive, that is the size of the keys. On the other side, it has been shown that, differently from RSA itself or from the elliptic curve digital signature algorithm, these two ciphers are among the few ciphers able to resist to quantum-computing cryptanalysis [5, p. 1], even if much bigger keys are needed [5, p. 2]. This is an important (if not the main) reason why the McEliece and the Niederreiter cryptosystems are so intensely investigated, trying to find solutions allowing a reduction of the keys sizes, often by changing the underlying code class.

In his paper, McEliece bases his result exclusively on one kind of codes, namely Goppa codes, while Niederreiter just requires the employment of some linear block code classes able of correcting efficiently a number of errors which should be as large as possible. Nowadays we know that many types of codes, even recently discovered, are not suitable for cryptographic purposes [2, 28]. An important fact is that, while the original McEliece implemented with Goppa codes is still unbroken [2, 3, 49] with exception of the result obtained by Bernstein, Lange and Peters [6], thanks to Sidel'nikov and Šestakov it turned out that the use of Niederreiter in combination with generalized Reed-Solomon codes is insecure [41].

The present thesis begins with a chapter dedicated to a background in coding theory. Here we introduce various code classes which found an application in code-based cryptography, such as Goppa codes, considered by McEliece to construct his cryptosystem, or RS codes, suggested by Niederreiter.

In the second chapter we expose some concepts which will be useful in the rest of the thesis, such as the number of irreducible polynomials of a given degree, a useful information for evaluating the safety level of Goppa codes. The section dedicated to the constant weight encoding contains an algorithm allowing the transformation of binary vectors into encodable messages. This, originally proposed by Guillot, has been found in two sources [12, 32], both containing some mistakes; we give a corrected version.

In the third chapter we describe more in detail the McEliece and the Niederreiter cryptosystems. Since their publication, uncountable variants of these two cryptosystems were proposed (see e.g. [2, 8, 24, 36, 44]). We will also present one of them, which takes the name of Sidel'nikov cryptosystem [40] and is based on Reed-Muller codes, chosen by the Russian mathematician thanks to their great error-correction capability, as he affirms in his own paper. Sadly, it was found out by Minder and Shokrollahi that the use of RM codes is not safe either [28].

Chapter 4 contains a discussion about the running time and the storage needs for implementing each of the three cryptosystems and their security against some attacks. In particular, the computational complexities of McEliece and Niederreiter have also been explicitly computed for some parameter values and inserted into a table, in order to evaluate for which steps and with which of these numbers one seems to be more efficient than the other. We will also insert an example for each of the two ways a cryptanalyst has for trying to break the McEliece or the Niederreiter system: the *decoding attack*, with which the attacker aims at decrypting a plaintext from an intercepted ciphertext, and the *structural attack*, whose goal is to recover the private code [3, 46]. For the first case we will present the message-resend attack [7], which only works against the McEliece cryptosystem, and for the second the well-known Sidel'nikov-Šestakov attack [41], able to compute a valid private key starting from the public key of a Niederreiter cryptosystem.

The last chapter is dedicated to one of the new variants of McEliece and Niederreiter. It was proposed in 2011 by Baldi, Bianchi, Chiaraluce, Rosenthal and Schipani. The underlying idea is to substitute the permutation matrix P, present in both cryptosystems, with another

matrix Q obtained by summing a dense matrix R and a sparse matrix T. The former must make the research of low-weight codewords difficult, while the latter, actually a sum of permutation matrices itself, has to be chosen so that the intentional errors do not exceed the limit imposed by the code. For a successful ciphertext decoding, the legal user (Bob) has to ask the sender (Alice) to use just error vectors respecting some constraints, given in form of a matrix; in this way, the dense R can be eliminated in the decoding phase. The purpose of this new variant is to allow the use of GRS codes after that it has been proved that they are unsafe if the original forms of the cryptosystems are used. However, having to respect some conditions has a drawback regarding both versions of the cipher, but which is more evident with the Niederreiter form: the combination of the public parity-check with the constraints matrix defines a public subcode that could permit the cryptanalysis of the system. To avoid this possibility, we propose not to publish the constraints matrix but directly the error vectors respecting it, or a part of them, trying to give an indication about the amount of binary vectors available in function of the code length. Unfortunately, also this attempt to reconsider GRS codes for cryptographic purposes turned out to be inapplicable. Very recently, in spring 2012, Gauthier, Otmani and Tillich found an attack to the new variant. We propose it at the end of the thesis.

Notation

We conclude this introduction with a little remark about the notation used throughout this thesis. The bold upper-case and lower-case letters ($\mathbf{A}$, $\mathbf{a}$) represent row vectors; also matrices are denoted by upper-case and lower-case letters, but with a different font (A, a). For better readability, the numbers above 10^5 are grouped with apostrophes every three digits and the fractional part is preceded by a comma (e.g. 123'456,789).

Chapter 1

Preliminaries about codes

We start this introductory chapter by giving some background information about coding theory, presenting at the same time the notation which will be used throughout the whole thesis. In the following sections we introduce some families of codes which are needed in the next chapters: Goppa and Reed-Solomon codes in relation to the McEliece cryptosystem and Reed-Muller codes in connection with the Sidel'nikov cryptosystem.

1.1 Basic concepts

The idea behind the employment of error correcting codes relies in the need of transmitting information from a sender to a receiver. By knowing that some errors could happen while the message runs through a noisy channel, some redundancy is added. A good code should be able of correcting as many errors as possible, without hindering a fast data transmission and efficient encoding and decoding phases.

We begin with a general definition of a code:

Definition 1.1.1 ([18, 23, 38]). Let $n, q \in \mathbb{N} = \{1, 2, 3, \ldots\}$ be natural numbers different from 0. A q-ary *alphabet* is a set of q distinct symbols $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$. A *$q$-ary block code of length n* is a set $\mathcal{C}$ of sequences of length n (called *codewords*) of elements from Λ, i.e.:

$$\mathcal{C} \subseteq \Big\{ (c_1 c_2 \ldots c_n) \mid c_i \in \Lambda \Big\}.$$

Example 1.1.2.

i. Setting $q = 27$ and $\Lambda_1 = \{a, b, \ldots, z, \text{-}\}$, "-" representing the hyphen, we can define as a code the set of all words in English (or in any language not including special characters). Assign to n the value of the longest word and complete the others by adding e.g. enough hyphens;

ii. Set $q = 2$ and $\Lambda_2 = \{0, 1\}$. Then $\mathcal{C}$ is called *binary code*. The words are sequences of 0 and 1.

For the rest of this thesis, we will concentrate on the case where q is a prime power, i.e. of the form $q = p^a$ for some prime p and natural number a. When talking about natural numbers, 0 will not be included, unless otherwise specified. We also introduce the notation $\mathbb{F}_q$ to indicate the finite field with q elements and we set $\Lambda = \mathbb{F}_q$.

Example 1.1.3. As $27 = 3^3$, we can define a code using an alphabet made of 3 symbols to encode the elements of Λ_1 from Ex. 1.1.2(i). In the case $\Lambda_3 = \{0, 1, 2\} = \mathbb{F}_3$, a possible *ternary* code would look like:

$$\mathcal{C} = \begin{cases} \text{-} & \longmapsto & (000) \\ a & \longmapsto & (001) \\ b & \longmapsto & (002) \\ & \vdots & \\ z & \longmapsto & (222). \end{cases}$$

Definition 1.1.4. Let $\mathbf{x} = (x_1 \dots x_n)$ and $\mathbf{y} = (y_1 \dots y_n)$ be words of a code $\mathcal{C}$.

- The *Hamming distance* between $\mathbf{x}$ and $\mathbf{y}$ is defined as $d_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) := \#\{i \mid x_i \neq y_i\}$;
- The *weight* of $\mathbf{x}$ is defined as $\mathrm{wt}(\mathbf{x}) := \#\{i \mid x_i \neq 0\} = d_{\mathrm{H}}(\mathbf{x}, \mathbf{0})$, where $\mathbf{0} := (0 \dots 0)$;
- The *(minimum) distance* of $\mathcal{C}$ is defined as $d(\mathcal{C}) := \min\{d_{\mathrm{H}}(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in \mathcal{C},\ \mathbf{x} \neq \mathbf{y}\}$.

Codewords are denoted as row vectors throughout this work. As such, the sum of two codewords will be denoted through the usual "+"-sign; even in the very frequent binary case where bitwise summation corresponds to the XOR-operation, for simplicity we will generally renounce the use of the symbol "$\oplus$". Wherever ordinary sum of two numbers in binary representation is meant, it will be explicitly indicated.

Definition 1.1.5. A linear subspace $\mathcal{C} \subseteq \mathbb{F}_q^n$ is called *linear block code* or, shortly, *linear code.* If $\mathcal{C}$ has dimension k and minimum distance d, then it is referred to as an $[n, k, d]$ code. The number $r := n - k$ of added coordinates is called *redundancy* or *codimension.*

Since an $[n, k, d]$ code $\mathcal{C}$ is a vector space, it is possible to find a basis. If G is a $k \times n$ matrix whose rows are basis vectors of $\mathcal{C}$, this code can be described through multiplication of all elements of $\mathbb{F}_q^k$ by G. Such a matrix is called *generator matrix* of $\mathcal{C}$ and we denote $\mathcal{C} = \langle \mathsf{G} \rangle$. On the other side, let H be a matrix such that $\mathcal{C} = \{\mathbf{x} \in \mathbb{F}_q^n \mid \mathsf{H}\mathbf{x}^\top = \mathbf{0}^\top\} = \ker(\mathsf{H})$. Then H is an $r \times n$ matrix and is called *parity-check matrix* of $\mathcal{C}$. If G is of the form $\mathsf{G} = \big(\mathsf{I}_k \mid \mathsf{A}\big)$, where I_j represents the $j \times j$ identity matrix and A is a $k \times r$ matrix, then $\big(-\mathsf{A}^\top \mid \mathsf{I}_r\big)$ is a valid parity-check matrix for $\langle \mathsf{G} \rangle$. In this case, G and H are said to be in *systematic form.*

Example 1.1.6. The code from Ex. 1.1.3 is a $[3, 3, 1]$ linear code with generator matrix $\mathsf{G} = \mathsf{I}_3$. It is: $\mathcal{C} = \mathbb{F}_3^3$.

1.2 Cyclic codes

We introduce here the concept of cyclic codes, which we will need in the next sections to present other classes of codes, such as BCH, RS and GRS codes.

Definition 1.2.1 (Cyclic code [26, 29]). A linear code $\mathcal{C}$ having length n is said to be *cyclic* if the presence of a codeword $\mathbf{c} = (c_0 c_1 \dots c_{n-1})$ in it implies that also the right shift $\vec{\mathbf{c}} = (c_{n-1} c_0 c_1 \dots c_{n-2}) \in \mathcal{C}\ \forall \mathbf{c}$.

It is possible to put every word of a cyclic code into one-to-one correspondence with a polynomial of degree $\leqslant n-1$. Let $\boldsymbol{R}_n := \mathbb{F}_q[x] \big/ \langle x^n - 1 \rangle$. The following mapping defines an isomorphism of vector spaces:

$$\begin{array}{ccc} \mathbb{F}_q^n & \xrightarrow{\sim} & \boldsymbol{R}_n \\ \mathbf{c} = (c_0 c_1 \ldots c_{n-1}) & \longmapsto & c(x) = c_0 + c_1 x + \ldots + c_{n-1} x^{n-1}. \end{array} \tag{1.1}$$

The two operations are the usual addition and multiplication. For simplicity, we will generally renounce to the $[\cdot]$-notation for the equivalence class of an element of $\boldsymbol{R}_n$ and will choose the polynomial of lowest degree as a representative, identifying $\boldsymbol{R}_n$ with the set $(\mathbb{F}_q)_{<n}[x]$ of polynomials over $\mathbb{F}_q$ of degree less than n. The above isomorphism also allows us to identify $(\mathbb{F}_q)_{<n}[x]$ with $\mathbb{F}_q^n$. In particular, notice that for some $c(x)$ (i.e. $\mathbf{c}$) as above it is:

$$xc(x) = c_0 x + c_1 x^2 + \ldots + c_{n-1} x^n \mod (x^n - 1) = c_{n-1} + c_0 x + \ldots + c_{n-2} x^{n-1} = \vec{\mathbf{c}}.$$

From now on we will hence use the vector and the polynomial notation interchangeably. This allows us to enunciate the following

Lemma 1.2.2 ([29, 38]). *A code $\mathcal{C}$ is cyclic $\iff$ $\mathcal{C}$ is an ideal of $\boldsymbol{R}_n$.*

Proof. "$\Longleftarrow$": Follows immediately from the fact that a multiplication by x corresponds to a cyclic shift.

"$\Longrightarrow$": Let $\mathbf{c} \in \mathcal{C}$ and $i_1, \ldots, i_s \in \mathbb{N} \cup \{0\}$. $\mathcal{C}$ is linear, hence any linear combination $x^{i_1}\mathbf{c} + x^{i_2}\mathbf{c} + \ldots + x^{i_s}\mathbf{c}$ of cyclic shifts of $\mathbf{c}$ is still a codeword. Since $\sum_{\sigma=1}^{s} x^{i_\sigma}$ describes an arbitrary element of $\boldsymbol{R}_n$, the claim follows. □

In the next steps we show the existence of so-called generator and check elements for any cyclic code.

Theorem 1.2.3 ([26, 29]). *Let $\mathcal{C} \subseteq \boldsymbol{R}_n$ be a cyclic code. Then:*

i. $\exists!$ *monic polynomial $g \in \mathbb{F}_q[x]$ of minimal degree such that $[g(x)] = g(x) + \langle x^n - 1 \rangle \in \mathcal{C}$;*

ii. $\mathcal{C} = \langle g(x) \rangle$*, i.e. $\mathcal{C}$ is a principal ideal;*

iii. $g(x) \mid (x^n - 1)$.

Proof. (i) Assume in $\mathcal{C}$ there are two different such polynomials g_1 and g_2, both of the same degree. Then $\deg(g_1 - g_2) < \deg(g_1) = \deg(g_2)$. $\lightning$

(ii) "$\supseteq$": $\mathcal{C}$ is an ideal and $g(x) \in \mathcal{C}$, hence $\langle g(x) \rangle \subseteq \mathcal{C}$.

"$\subseteq$": Assume $\mathcal{C} \ni c(x) = p(x)g(x) + s(x)$ for some polynomials $p, s \in \mathbb{F}_q[x]$ with $\deg s < \deg g$. Then, being $\mathcal{C}$ an ideal, $s = c - pg \in \mathcal{C}$. By working over $\mathcal{C}$ (i.e. over $\boldsymbol{R}_n$), from (i) it follows $s = 0$; thus, $\mathcal{C} \subseteq \langle g(x) \rangle$.

(iii) Let p and s be as before and assume $x^n - 1 = pg + s$. This implies that in $\boldsymbol{R}_n$, $s = -pg \in \mathcal{C}$, hence also $pg \in \mathcal{C}$. Since $\deg s < \deg g < n$, $s = 0$. □

Definition 1.2.4.

i. The polynomial g introduced in Thm. 1.2.3 is given the name of *generator polynomial* of the code $\mathcal{C}$;

ii. If $\mathcal{C}$ is generated by g, then the *check polynomial* of $\mathcal{C}$ is the polynomial $h \in \boldsymbol{R}_n$ such that $g(x)h(x) = x^n - 1$.

Lemma 1.2.5 ([26, 37]). *Assume some cyclic code $\mathcal{C}$ is generated by $g(x) = g_0 + g_1 x + \ldots + g_r x^r$ ($g_r \neq 0$) and has check polynomial $h(x)$. Let $r := \deg g$ and $k := \deg h = n - r$. Then, $\dim(\mathcal{C}) = k$ and*

$$\mathsf{G} = \begin{pmatrix} g_0 & g_1 & \cdots & g_r & & & \\ & g_0 & g_1 & \cdots & g_r & & 0 \\ & & \ddots & \ddots & \ddots & \ddots & \\ 0 & & & g_0 & g_1 & \cdots & g_r \end{pmatrix}$$

is a generator matrix for $\mathcal{C}$.

Proof. By Thm. 1.2.3(ii), we know that for any $c \in \mathcal{C}$ it is $c(x) = a(x)g(x)$ for some $a \in \boldsymbol{R}_n$. If we divide a by h, we get

$$c(x) = a(x)g(x) = (p(x)h(x) + s(x))g(x),$$

where $\deg s < k$. As a consequence, since $g(x)h(x) = 0$ in $\boldsymbol{R}_n$, we have $c(x) = s(x)g(x)$. Hence, $\deg c < n$. This means that every codeword can be constructed by multiplying $g(x)$ by a polynomial of degree less than k. This also tells us that the k-dimensional space spanned by the multiplication of $\{1, x, x^2, \ldots, x^{k-1}\}$ by $g(x)$ is a basis of $\mathcal{C}$, which justifies the form of the k rows of G; in symbols: $\mathcal{C} = g(x) \cdot (\mathbb{F}_q)_{<k}[x]$ over $\mathbb{F}_q$. □

It is also important to note that the converse of Thm. 1.2.3(iii) holds too.

Theorem 1.2.6 ([37, p. 323]). *If $f \in \mathbb{F}_q[x]$ is a monic polynomial dividing $x^n - 1$, then it is the generator of a cyclic code.*

Proof. Assume $g(x) \neq f(x)$ is the polynomial of minimal degree generating the same code as $f(x)$. By Thm. 1.2.3(i), $\deg f > \deg g$. Since $g(x) \in \langle f(x) \rangle$, $g(x) = a(x)f(x)$ for some $a \in \boldsymbol{R}_n$. Let now $h_f(x)$ be the check polynomial of $\langle f(x) \rangle$ corresponding to $f(x)$:

$$g(x) = a(x)f(x) \Longrightarrow g(x)h_f(x) = a(x)f(x)h_f(x) \overset{f \cdot h_f = 0}{\overset{\downarrow}{=}} 0 \mod x^n - 1. \tag{1.2}$$

Since $\deg h_f = n - \deg f \Longrightarrow \deg(g \cdot h_f) < n$, contradicting (1.2); thus, $g(x) = f(x)$. □

The next step is the definition of the concept of roots of unity. From now on, let $\mathbb{F}_{q^s}$ denote the splitting field of the polynomial $x^n - 1$ over $\mathbb{F}_q$ and assume $\gcd(n, q) = 1$.

Definition 1.2.7 (Root of unity [26, 37]). The fields $\mathbb{F}_q$ and $\mathbb{F}_{q^s}$ as above are given. Then any element $\alpha \in \mathbb{F}_{q^s}$ satisfying $\alpha^n = 1$ is called *n^{th} root of unity over* $\mathbb{F}_q$. All n^{th} roots of unity are collected into a set denoted $E^{(n)}$.

The reason why we need $\gcd(n, q)$ to be 1 is the following: the derivative of $x^n - 1$ is nx^{n-1} (see also Def. 1.4.9); if n and q are coprime, these two functions have no common roots, implying the existence of n distinct roots of unity in $\mathbb{F}_{q^s}$. On the contrary, if $\gcd(n, q) > 1$, there would be a smaller number of roots. Furthermore, the couple $(E^{(n)}, \cdot)$ forms a cyclic subgroup of $\mathbb{F}_{q^s}^*$ because it is a subgroup of an integral domain. A straightforward consequence is the next

Definition 1.2.8 ([37, 38]). If the group $E^{(n)}$ is generated by some n^{th} root of unity ω (in symbols: $E^{(n)} = \langle\omega\rangle$), then ω is called *primitive n^{th} root of unity.*

Remark 1.2.9.

i. $\mathbb{F}_{q^s} = \mathbb{F}_q(\omega)$;

ii. If we denote $E^{(n)} = \{\alpha_1, \ldots, \alpha_n\}$ and let ω be a primitive root of unity over $\mathbb{F}_q$, we have:

$$x^n - 1 = \prod_{i=1}^{n}(x - \alpha_i) = \prod_{i=0}^{n-1}(x - \omega^i).$$

Clearly, from (ii) it follows that the minimal polynomials of the roots of unity must divide $x^n - 1$. Before going on, we need a property of the roots of irreducible polynomials.

Theorem 1.2.10 ([38]). *Let $f \in \mathbb{F}_q[x]$ be an irreducible polynomial of degree d and α be a root of f in some extension field; then the other roots are $\alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{d-1}}$. Moreover, $\mathbb{F}_{q^d}$ is the splitting field of f.*

Proof. Define $\mathbb{F}_{q^d} \cong \mathbb{F}_q[x]/\langle f(x)\rangle$; hence, $\alpha \in \mathbb{F}_{q^d}$. We also have $f(\alpha^{q^j}) = f(\alpha)^{q^j} = 0 \ \forall j$. Because we are working over finite fields and with a polynomial of given degree, the cycle is bounded at $j = d - 1$. □

Per definition, a minimal polynomial has to be irreducible. Given some primitive root of unity ω, the conjugated elements of ω^i (i.e. the ones having the same minimal polynomial) are

$$\omega^{iq}, \omega^{iq^2}, \ldots, \omega^{iq^{d_i-1}},$$

where d_i is minimal with $iq^{d_i} = i \mod n$ [26, 37]. Being this valid $\forall i \in \mathbb{Z}_n = \{0, \ldots, n-1\}$, the exponents of the elements conjugated with any ω^i can be collected into *cyclotomic cosets* of the form:

$$C_i = \{i, iq, iq^2, \ldots, iq^{d_i-1}\},$$

with d_i as above. It is obvious that the cyclotomic cosets are disjunct and that $C_i = C_{iq} = \ldots = C_{iq^{d_i-1}}$. We resume this result in the following

Corollary 1.2.11 ([26, 38]). *Let ω be a primitive root of unity over $\mathbb{F}_q$. Then the minimal polynomial of ω^i is given by*

$$M_i(x) = \prod_{j \in C_i}(x - \omega^j) \in \mathbb{F}_q[x]$$

and consequently $x^n - 1$ is equal to the product of all distinct $M_j(x)$.

Remark 1.2.12. For every cyclotomic coset it is: $\#C_i = \deg M_i(x)$.

Thus, with the minimal polynomials being monic and irreducible, we have described an easy way to find the degree of the factors of $x^n - 1$. This can be exploited to introduce a different manner to define cyclic codes.

Remark 1.2.13. From Thm. 1.2.6 and Cor. 1.2.11 it follows that any generator polynomial of a cyclic code can be represented as a product of the form

$$g(x) = \prod_{j \in U} (x - \omega^j),$$

where $U := C_{i_1} \cup \ldots \cup C_{i_\ell} \subseteq \mathbb{Z}_n$. This corresponds to the product of the distinct minimal polynomials whose roots are the ω^j appearing in g.

We have seen that the minimal polynomial is the same for all elements of a cyclotomic coset; so, to have any M_i completely defined, it is enough to choose a representative exponent for each of the corresponding C_i. Let S be a set containing exactly one representative for every cyclotomic coset and $\tilde{S} \subseteq S$. Then, the result obtained in Rk. 1.2.13 can be rewritten as:

$$g(x) = \prod_{j \in \tilde{S}} M_j(x). \tag{1.3}$$

Now let $i \in \tilde{S}$, i.e. $M_i(x) \mid g(x)$; hence, if ω is primitive, M_i is the minimal polynomial of ω^i, implying $g(\omega^i) = 0$. More generally, for any polynomial $c \in \mathbb{F}_q[x]$, we have that $c(\omega^i) = 0 \iff M_i(x) \mid c(x)$, which by restricting our attention to $c \in \boldsymbol{R}_n$ becomes $c(\omega^i) = 0 \iff c(x) \in \langle M_i(x) \rangle$. If this is valid $\forall i \in \tilde{S}$, then it is also $c(x) \in \langle g(x) \rangle$. This enables us to reach the main result of this section:

Theorem 1.2.14. *Let $\tilde{S}$ be as above and ω^i be a root of the minimal polynomial $M_i(x)$ $\forall i \in \tilde{S}$. Then the cyclic code generated by $g(x) = \prod_{j \in \tilde{S}} M_j(x)$ is completely defined by ω^i and can be also written as:*

$$\langle g(x) \rangle = \left\{ c(x) \in \boldsymbol{R}_n \;\middle|\; c(\omega^j) = 0 \ \forall j \in \tilde{S} \right\}.$$

Until now we have always assumed a generator g to be monic, irreducible and with the lowest possible degree. Though, it is possible to generalize a bit this result by choosing as a code generator a polynomial $f \in \boldsymbol{R}_n$ instead of a product of minimal polynomials. Let us see which consequences this has.

Lemma 1.2.15 ([26, p. 199]**).** *Let g be as in* (1.3) *and $p \in \boldsymbol{R}_n$ such that all the roots of unity which are zeros of p are also zeros of g. Then $\langle pg \rangle = \langle g \rangle$.*

Proof. "$\subseteq$": Clear.

"$\supseteq$": Let h be the check polynomial of the code generated by g. We know that $\gcd(p(x), h(x)) = 1$, so we can apply the Euclidean algorithm. Let a and b be some polynomials over $\mathbb{F}_q$ such that $a(x)p(x) + b(x)h(x) = 1$; then:

$$ap + bh = 1 \Longrightarrow gap + gbh = g \quad \overset{gh=0 \text{ in } \boldsymbol{R}_n}{\Longrightarrow} \quad gap = g \mod (x^n - 1).$$

Hence, $\langle g \rangle \subseteq \langle pg \rangle$. □

Thus, whenever a polynomial $f \in \boldsymbol{R}_n$ can be written as $f = pg$ with p as above, the codes generated by g and f are the same (other generators for a code are for example the idempotents). This can be resumed in the following

Corollary 1.2.16. *Let $f \in \boldsymbol{R}_n$ have the roots $\alpha_1, \ldots, \alpha_\sigma \in E^{(n)}$ over $\mathbb{F}_{q^s}$. If their minimal polynomials are $M^{(1)}(x), \ldots, M^{(\sigma)}(x)$ and their* distinct *minimal polynomials are $\tilde{M}^{(1)}, \ldots, \tilde{M}^{(\tilde{\sigma})}$, then for the code generated by f it is:*

$$\langle f(x) \rangle = \left\{ c(x) \in \boldsymbol{R}_n \;\middle|\; c(\alpha_j) = 0 \ \forall j \right\} = \left\langle \prod_{j=1}^{\tilde{\sigma}} \tilde{M}^{(i)}(x) \right\rangle = \langle \mathrm{lcm}\{M^{(1)}(x), \ldots, M^{(\sigma)}(x)\} \rangle.$$

What we have seen up to now — Thm. 1.2.14 in particular — suggests us a way to build a parity-check matrix for a cyclic code. Let $\mathcal{C} = \left\{ c(x) \in \boldsymbol{R}_n \;\middle|\; c(\alpha_1) = \ldots = c(\alpha_\sigma) = 0 \right\}$, with α_j some roots of unity over $\mathbb{F}_q$. Then $c(x)$ is a codeword of $\mathcal{C} \iff \mathsf{H}\mathbf{c}^\top = \mathbf{0}$, where $\mathbf{c}$ is the row vector form of $c(x)$ as in (1.1) and

$$\mathsf{H} = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_\sigma & \alpha_\sigma^2 & \cdots & \alpha_\sigma^{n-1} \end{pmatrix}. \tag{1.4}$$

If the splitting field of $x^n - 1$ is $\mathbb{F}_{q^s}$, then we can exploit the bijection between $\mathbb{F}_{q^s}$ and $\mathbb{F}_q^s$ to substitute the entries of H through the corresponding length-s column vectors in $\mathbb{F}_q$, in order to get a bigger matrix H' over $\mathbb{F}_q$ of size $s\sigma \times n$. For simplicity, it will be convenient to delete the linearly dependent rows from the new matrix.

1.3 BCH, RS and GRS codes

After some theory about cyclic codes and related concepts like roots of unity and minimal polynomials, we are now almost ready to define BCH codes. They are named after R. C. Bose and D. K. Ray-Chaudhuri, who discovered them in 1960, and A. Hocquenghem, who found them independently one year earlier. BCH codes also serve as a basis for introducing two other classes of codes, called Reed-Solomon and generalized Reed-Solomon codes. Before that, we just need the following

Theorem 1.3.1 (BCH bound [26, 37]). *Let ω be a primitive n^{th} root of unity over $\mathbb{F}_q$, $b \in \mathbb{N} \cup \{0\}$ and $\delta \in \mathbb{N}$. Consider the consecutive powers $\omega^b, \omega^{b+1}, \ldots, \omega^{b+\delta-2}$ and the code $\mathcal{C} = \langle g(x) \rangle$, where $g(x)$ is the monic polynomial over $\mathbb{F}_q$ of smallest degree having the ω^i among its zeros. Then, $d(\mathcal{C}) \geqslant \delta$.*

Proof. Analogously to (1.4), it is possible to express the parity-check matrix of $\mathcal{C}$ as

$$\mathsf{H} = \begin{pmatrix} 1 & \omega^b & \omega^{2b} & \cdots & \omega^{(n-1)b} \\ 1 & \omega^{b+1} & \omega^{2(b+1)} & \cdots & \omega^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{b+\delta-2} & \omega^{2(b+\delta-2)} & \cdots & \omega^{(n-1)(b+\delta-2)} \end{pmatrix}. \tag{1.5}$$

As by assumption the ω^i, $i \in \{b, \ldots, b+\delta-2\}$, just have to be *among* the zeros of g, it means that $\mathcal{C} \subseteq \left\{ \mathbf{c} \in \mathbb{F}_q^n \;\middle|\; \mathsf{H}\mathbf{c}^\top = 0 \right\}$. We know that the distance of a linear code is given by the lowest

number of linearly dependent columns of a parity-check matrix [38], hence we try to compute the determinants of the biggest square submatrices of H, i.e. the $(\delta-1)\times(\delta-1)$ minors. Let $n_1, n_2, \ldots, n_{\delta-1}$ be distinct elements of $\mathbb{Z}_n$ and let A have the requested properties. Then:

$$\det \mathsf{A} = \det \begin{pmatrix} \omega^{n_1 b} & \omega^{n_2 b} & \cdots & \omega^{n_{\delta-1} b} \\ \omega^{n_1(b+1)} & \omega^{n_2(b+1)} & \cdots & \omega^{n_{\delta-1}(b+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{n_1(b+\delta-2)} & \omega^{n_2(b+\delta-2)} & \cdots & \omega^{n_{\delta-1}(b+\delta-2)} \end{pmatrix} \overset{\substack{\text{divide the } i^{\text{th}} \\ \text{column by } \omega^{n_i b}}}{\overset{\downarrow}{=}}$$

$$= \omega^{n_1 b}\omega^{n_2 b}\cdots\omega^{n_{\delta-1} b} \det \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \omega^{n_1} & \omega^{n_2} & \cdots & \omega^{n_{\delta-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{n_1(\delta-2)} & \omega^{n_2(\delta-2)} & \cdots & \omega^{n_{\delta-1}(\delta-2)} \end{pmatrix}.$$

Being $\omega \in E^{(n)}$, $\omega^{(n_1+\ldots+n_{\delta-1})b} \neq 0$. Moreover, because ω is primitive, $\omega^{n_i} \neq \omega^{n_j}$ for $i \neq j$; hence, the resulting matrix is a Vandermonde matrix, i.e. a matrix where every element appears in all its powers in ascending order. Any minor of maximum size of such a matrix is different from zero; thus, $\det \mathsf{A} \neq 0 \Longrightarrow d(\mathcal{C}) \geqslant \delta$. □

Definition 1.3.2 (BCH code [23, 37]). Let ω, b and δ be as in Thm. 1.3.1. Consider also ω^i with $i = b, \ldots, b+\delta-2$. Then, the code

$$\mathsf{BCH}_q(n, \delta, \omega, b) := \langle g(x) \rangle$$

where $g(x) = \operatorname{lcm}\{M_b(x), \ldots, M_{b+\delta-2}(x)\} \in \mathbb{F}_q[x]$, is called *$q$-ary BCH code of length n and designed distance δ.*

Lemma 1.3.3. *BCH codes have dimension $k = n - \#\{C_b \cup C_{b+1} \cup \ldots \cup C_{b+\delta-2}\}$.*

Proof. Let $\mathcal{C} = \langle g(x) \rangle$ be a BCH code such that g and all other values are as before. In Rk. 1.2.12 we have seen that the cardinality of a cyclotomic coset is equal to the degree of the related minimal polynomial. Thus, by Lemma 1.2.5 we have that

$$k = n - \deg g = n - \deg(\operatorname{lcm}\{M_b(x), \ldots, M_{b+\delta-2}(x)\}),$$

from which the claim follows. □

There exist some special cases of BCH codes which deserve to be mentioned:

Definition 1.3.4 ([23, 37]).

i. If $b = 1$, the code is called *narrow sense* BCH code;

ii. If for the primitive root of unity ω it is also true that $\langle \omega \rangle = \mathbb{F}_{q^s}^*$, i.e. $n = q^s - 1$, then we say that the code is a *primitive* BCH code;

iii. When $n = q - 1 \geqslant 2$, then the code is referred to as a *Reed-Solomon* code, shorter RS code.[2]

Remark 1.3.5.

i. A RS code is also a primitive BCH code with $s = 1$;

ii. A RS code is generated when the primitive root of unity ω is such that $\langle\omega\rangle = \mathbb{F}_q^*$, i.e. $\omega \in \mathbb{F}_q^*$, because $x^n - 1 = x^{q-1} - 1 = \prod_{a\in\mathbb{F}_q^*}(x - a)$. This means that the splitting field of $x^n - 1$ is $\mathbb{F}_q$ itself, so the minimal polynomial of any unit $a \in \mathbb{F}_q^*$ is $M_i(x) = x - a = x - \omega^i$ for some i;

iii. By (1.3), a RS code $\mathcal{C}$ is generated by a product of linear factors; in fact, note that $S = \mathbb{Z}_n$, since the cyclotomic cosets all contain just one element.

This is confirmed by Def. 1.3.2, which tells us that:

Corollary 1.3.6. *A RS code $\mathcal{C}$ of designed distance δ is given by:*

$$\mathcal{C} = \mathsf{BCH}_q(q-1, \delta, \omega, b) = \langle g(x)\rangle = (x - \omega^b)(x - \omega^{b+1})\cdots(x - \omega^{b+\delta-2}),$$

where ω is such that $\langle\omega\rangle = \mathbb{F}_q^*$ and $b \in \mathbb{N} \cup \{0\}$.

Example 1.3.7. We want a narrow sense RS code $\mathcal{C}$ over $\mathbb{F}_7$ of distance at least 3. We note that 3 is a primitive field element. Then a generator and a check polynomial of $\mathcal{C}$ are respectively

$$g(x) = (x - 3)(x - 3^2) = x^2 + 2x + 6 \quad \text{and} \quad h(x) = x^4 + 5x^3 + 5x^2 + 2x + 1,$$

because $g(x)h(x) = x^6 - 1$.

We now introduce a class of codes generalizing what we have seen until now.

Definition 1.3.8 (Generalized RS code [26, p. 303]). Let $\mathbf{a} := (a_1, \ldots, a_n) \in \mathbb{F}_q^n$ be a vector containing n *distinct* coordinates and $\mathbf{v} := (v_1, \ldots, v_n) \in (\mathbb{F}_q^*)^n$ be another vector whose coordinates are non-zero but not necessarily distinct. Then the set

$$\mathsf{GRS}_k(\mathbf{a}; \mathbf{v}) := \left\{ \big(v_1 f(a_1), v_2 f(a_2), \ldots, v_n f(a_n)\big) \;\middle|\; f(x) \in (\mathbb{F}_q)_{<k}[x] \right\},$$

where $(\mathbb{F}_q)_{<k}[x]$ stays for the polynomials over $\mathbb{F}_q$ of degree less than $k \leqslant n$, is called *generalized Reed-Solomon code of length n and dimension k.*

Proposition 1.3.9. *A valid generator matrix for a GRS code is given by*

$$\mathsf{G} := \mathsf{G}(\mathbf{a}; \mathbf{v}) = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ v_1 a_1 & v_2 a_2 & \cdots & v_n a_n \\ \vdots & \vdots & \ddots & \vdots \\ v_1 a_1^{k-1} & v_2 a_2^{k-1} & \cdots & v_n a_n^{k-1} \end{pmatrix}. \tag{1.6}$$

[2] Many sources (e.g. [17, 38]) prefer to give a more general definition of RS codes and just require n to be less (or even equal) than q.

Proof. Follows directly from the definition. □

Proposition 1.3.10. *GRS codes are $[n, k, n-k+1]$ codes, i.e. belong to the class of MDS (maximum distance separable) codes.*

Proof. Length, dimension and linearity are clear. Regarding distance: note that a polynomial of degree $< k$ cannot have more than $k-1$ zeros, hence, for the v_j being elements of $\mathbb{F}_q^*$, the codewords have at most $k-1$ components equal to 0. This means that the code has distance $d(\mathcal{C}) \geqslant n-k+1$. The Singleton bound ($d+k \leqslant n+1$ [38]) implies equality. □

Lemma 1.3.11 ([26, pp. 304 and 334]**).** *GRS codes have a parity-check matrix of the form*

$$\mathsf{H} := \mathsf{H}(\mathbf{a};\mathbf{z}) = \mathsf{X}(\mathbf{a}) \cdot \mathsf{Y}(\mathbf{z}) := \begin{pmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ a_1^2 & a_2^2 & \cdots & a_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{r-1} & a_2^{r-1} & \cdots & a_n^{r-1} \end{pmatrix} \begin{pmatrix} z_1 & & & \\ & z_2 & & 0 \\ & & \ddots & \\ 0 & & & z_n \end{pmatrix} \tag{1.7}$$

for some $\mathbf{z} := (z_1, \ldots, z_n) \in (\mathbb{F}_q^*)^n$, r representing the redundancy ($r = n-k$). Note that X is a Vandermonde matrix, hence H is sometimes called generalized Vandermonde matrix *[15]. Moreover, if $\mathbf{a}$ and $\mathbf{v}$ are as above and H defines the code $\mathsf{GRS}_k(\mathbf{a};\mathbf{v})$, then $\mathbf{z}$ is such that $\mathsf{GRS}_r(\mathbf{a};\mathbf{z}) = \mathsf{GRS}_k(\mathbf{a};\mathbf{v})^\perp$, that is the dual code of $\mathsf{GRS}_r(\mathbf{a};\mathbf{z})$.*

Remark 1.3.12. As we have seen in Def. 1.3.4, RS codes are special cases of BCH codes. Moreover, as the name itself suggests, they are also particular cases of GRS codes. Hence, it must be possible to give a parity-check matrix for RS codes respecting both definitions. In (1.5) we encountered the form of a parity-check matrix for a BCH code. If, by using the same notation as there, we assume that $a_j = \omega^{j-1}$ and $z_j = \omega^{(j-1)b}$, we get

$$\mathsf{H}(\mathbf{a};\mathbf{z}) = \left(z_j a_j^{i-1}\right)_{\substack{i=1,\ldots,r,\\ j=1,\ldots,n}} = \left(\omega^{(j-1)b}\omega^{(i-1)(j-1)}\right)_{\substack{i=1,\ldots,\delta-1,\\ j=1,\ldots,n}} = \left(\omega^{(j-1)(b+i-1)}\right)_{i,j}, \tag{1.8}$$

which is exactly the matrix found in equation (1.5).

Before going over to the next section, we need to define another important class of codes.

Definition 1.3.13 (Alternant code [26, p. 334]**).** Let $\mathbf{a}$ and $\mathbf{v}$ have the same characteristics as before over some field $\mathbb{F}_{q^m}$ and let $\mathsf{GRS}_k(\mathbf{a};\mathbf{v})$ be the GRS code over $\mathbb{F}_{q^m}$ generated by them. Then the *alternant code* $\mathsf{Alt}(\mathbf{a};\mathbf{v})$ is the code over $\mathbb{F}_q$ such that

$$\mathsf{Alt}(\mathbf{a};\mathbf{v}) := \mathsf{GRS}_k(\mathbf{a};\mathbf{v}) \cap \mathbb{F}_q^n.$$

Because it is a subcode of $\mathsf{GRS}_k(\mathbf{a};\mathbf{v})$, the matrix H given in (1.7) is also a parity-check matrix of $\mathsf{Alt}(\mathbf{a};\mathbf{v})$.

1.4 Goppa codes

Goppa codes were discovered by the Russian mathematician V. D. Goppa. They can be seen as a generalization of BCH codes and, like them, constitute a subclass of the alternant codes [23, 26, 45].

We define for simplicity $\boldsymbol{S}_m := \mathbb{F}_{q^m}[x]/\langle G(x)\rangle$ for some polynomial G.

Definition 1.4.1 (Goppa code [23, 39]). Let q be a prime power and n, m be natural numbers such that $n \leqslant q^m$. Let G be a polynomial of degree τ over $\mathbb{F}_{q^m}$ and $L := \{\gamma_1, \dots, \gamma_n\}$ a subset of $\mathbb{F}_{q^m}$ such that none of its elements is a zero of G. We define as

$$\Gamma(L, G) := \left\{ \mathbf{c} = (c_1 c_2 \dots c_n) \in \mathbb{F}_q^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{x - \gamma_i} = 0 \in \boldsymbol{S}_m \right\}$$

the q-ary *Goppa code of length n, Goppa polynomial G and support L*. The code is called *irreducible* if G is irreducible.

Remark 1.4.2. If a Goppa code $\Gamma(L, G)$ is irreducible, then one can choose for L the biggest possible size, that is $L = \mathbb{F}_{q^m}$.

It is clear that the condition saying that $G(\gamma_j) \neq 0 \ \forall j = 1, \dots, n$ is needed to ensure the existence of the summation. Actually, as we will see in a while, the fraction contained in it is a polynomial; furthermore, this polynomial will lead us to a second way for defining Goppa codes. This alternative definition will be helpful in one of the main results of the present section.

Lemma 1.4.3 ([23, 38]). *Let G be as above and $\gamma \in \mathbb{F}_{q^m}$ be such that $G(\gamma) \neq 0$. Then the element $(x - \gamma)$ is invertible in $\boldsymbol{S}_m$ and it is*

$$(x - \gamma)^{-1} = \frac{-1}{G(\gamma)} \frac{G(x) - G(\gamma)}{x - \gamma}.$$

Proof. Obviously, $\boldsymbol{S}_m$ is a field $\iff$ the polynomial G is irreducible. Though, even in the case G is not irreducible and $\boldsymbol{S}_m$ is just a ring, $x - \gamma$ is a unit of $\boldsymbol{S}_m$, i.e. it has an inverse element. Division with remainder yields $G(x) = q(x)(x - \gamma) + G(\gamma)$, from which we get

$$-G(\gamma)^{-1} q(x)(x - \gamma) = 1 \mod G(x) \quad \text{and} \quad q(x) = \frac{G(x) - G(\gamma)}{x - \gamma};$$

thus, $(x - \gamma) \mid (G(x) - G(\gamma))$. By substituting $q(x)$ in the left equality, we get the claim. □

Corollary 1.4.4. *The expression $\frac{G(x)-G(\gamma)}{x-\gamma}$, where G and γ are as above, is a polynomial.*

Proposition 1.4.5 (Alternative definition of a Goppa code). *Let all parameters be as in* Def. 1.4.1. *Then a q-ary Goppa code of length n can be seen as the set*

$$\Gamma(L, G) = \left\{ \mathbf{c} = (c_1 c_2 \dots c_n) \in \mathbb{F}_q^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{G(\gamma_i)} \frac{G(x) - G(\gamma_i)}{x - \gamma_i} = 0 \in \mathbb{F}_{q^m}[x] \right\}.$$

By Cor. 1.4.4, $\deg\left(\frac{G(x)-G(\gamma_i)}{x-\gamma_i}\right) = \deg(G) - 1$, *hence here it is possible to omit the modulo.*

Theorem 1.4.6 ([23, 37, 38]). *The distance of a Goppa code having Goppa polynomial $G(x)$ is bigger than its degree τ.*

Proof. Through the form obtained in Prop. 1.4.5 and by performing some computations, it is possible to construct a parity-check matrix for Goppa codes (see e.g. [37, pp. 390–393]). Then, the so obtained parity-check matrix H can be decomposed into the product of three matrices $\mathsf{H} = \mathsf{WXY}$, where W, X and Y are respectively of the form

$$\begin{pmatrix} G_\tau & & & 0 \\ G_{\tau-1} & G_\tau & & \\ \vdots & \ddots & \ddots & \\ G_1 & \cdots & G_{\tau-1} & G_\tau \end{pmatrix}, \begin{pmatrix} 1 & \cdots & 1 \\ \gamma_1 & \cdots & \gamma_n \\ \vdots & \ddots & \vdots \\ \gamma_1^{\tau-1} & \cdots & \gamma_n^{\tau-1} \end{pmatrix} \text{ and } \begin{pmatrix} \frac{1}{G(\gamma_1)} & & & 0 \\ & \frac{1}{G(\gamma_2)} & & \\ & & \ddots & \\ 0 & & & \frac{1}{G(\gamma_n)} \end{pmatrix}$$

(compare this with (1.7)), in which the Goppa polynomial is defined as $G(x) = \sum_{i=0}^{\tau} G_i x^i$, $G_\tau \neq 0$. We know that the elements $\gamma_1, \ldots, \gamma_n \in L$ are pairwise distinct and that among them there are no roots of G, hence X is a Vandermonde matrix. It is also $\det(\mathsf{W}) = G_\tau^\tau \neq 0$. That is, the matrix H has full rank, implying that its kernel is a code of distance at least $\tau + 1$. □

Corollary 1.4.7. *As it is the case for BCH codes, also for Goppa codes it is possible to choose a designed distance δ, by taking $\tau = \delta - 1$.*

Remark 1.4.8 ([37, 38]).

i. A Goppa code is still completely defined even if the matrix W is omitted from the above product;

ii. The dimension k of a Goppa code can be bounded through $n - m\tau \leqslant k \leqslant n - \tau$.

Proof. (i) As we have seen in the proof of Thm. 1.4.6, $\det(\mathsf{W}) \neq 0$; then, given a Goppa code Γ defined by H it holds: $\mathbf{c} \in \Gamma \iff \mathsf{H}\mathbf{c}^\top = \mathsf{WXY}\mathbf{c}^\top = 0 \iff \mathsf{XY}\mathbf{c}^\top = 0$.

(ii) The right inequality is given by the Singleton bound and Thm. 1.4.6 ($d \geqslant \tau + 1$). Concerning the left hand side: construct a new parity-check matrix H'' by substituting the entries of H', which are elements of $\mathbb{F}_{q^m}$, through the corresponding strings in $\mathbb{F}_q^m$. Then H'' has size at most $m\tau \times n$ and its kernel is a vector space over $\mathbb{F}_q$ of dimension at least $n - m\tau$. □

Definition 1.4.9 (Formal derivative [21, p. 199]). Let $f(x)$ be a polynomial over an arbitrary ring. Then the *formal derivative* of $f(x) = f_0 + f_1 x + \ldots + f_n x^n$ is defined as

$$f'(x) := f_1 + 2f_2 x + \ldots + nf_n x^{n-1}.$$

Remark 1.4.10. The formal derivative respects the product rule, just as the usual derivative known from analysis; i.e. for two polynomials f and g it is

$$(fg)'(x) = f'(x)g(x) + f(x)g'(x)$$

and more generally, for s polynomials $f_1, \ldots, f_s$ we have

$$(f_1 f_2 \cdots f_s)'(x) = (f_1' f_2 \cdots f_s)(x) + (f_1 f_2' \cdots f_s)(x) + \ldots + (f_1 f_2 \cdots f_s')(x).$$

After having introduced the concept of formal derivative, we are now able to present an important result about minimum distance involving the family of *binary* Goppa codes, i.e. Goppa codes over a binary field. As we will see in the next chapters, such a subclass of codes was exploited by McEliece to formulate his cryptosystem.

Theorem 1.4.11 ([23, 39]). *Let $q = 2$ and let $\Gamma(L, G)$ be a binary Goppa code having support $L = \{\gamma_1, \dots, \gamma_n\} \subseteq \mathbb{F}_{2^m}$; then the Goppa polynomial G is an element of $\mathbb{F}_{2^m}[x]$. If G has degree τ and is square-free (which means that all roots are simple), then $d(\Gamma) \geqslant 2\tau + 1$.*

Proof. Let $\mathbf{c} := (c_1 \dots c_n) \in \Gamma \setminus \{\mathbf{0}\}$, i.e. a codeword with $\mathrm{wt}(\mathbf{c}) > 0$. We name the indices of the non-zero components of the word as $j_1, j_2, \dots, j_{\mathrm{wt}(\mathbf{c})}$ and collect them into a set, say J. Let $P(x) := \prod_{\sigma \in J}(x - \gamma_\sigma)$ be the polynomial having as roots exactly the elements of L with indices in J. Assuming all calculations are made modulo $G(x)$, for the word $\mathbf{c}$ we know that

$$0 = P(x) \sum_{j=1}^{n} \frac{c_j}{G(\gamma_j)} \frac{G(x) - G(\gamma_j)}{x - \gamma_j} = P(x) \sum_{s \in J} \overbrace{c_s}^{=1\ \forall s} \cdot \frac{1}{x + \gamma_s} \frac{G(\gamma_s)}{G(\gamma_s)} =$$

$$= \prod_{\sigma \in J}(x + \gamma_\sigma) \cdot \sum_{s \in J} \frac{1}{x + \gamma_s} = \sum_{s \in J} \prod_{\sigma \neq s}(x + \gamma_\sigma) \overset{\substack{\text{Rk.} \\ 1.4.10}}{=} P'(x).$$

This means that $P'(x) \in \langle G(x) \rangle$. Because we are working over characteristic 2, P' is composed only of monomials of even power:

$$P'(x) = \sum p_{2i} x^{2i} = \left(\sum p_{2i} x^i \right)^2.$$

We assumed G to be square-free, so $G(x) \mid P'(x) \Longrightarrow G(x)^2 \mid P'(x)$, i.e. P' is also a codeword of $\langle G(x)^2 \rangle$. This means that:

$$2\tau = \deg(G^2) \leqslant \deg(P') < \deg(P) = \mathrm{wt}(\mathbf{c}).$$

Applying the fact that $\mathbf{c}$ was chosen at random among all possible non-zero codewords of Γ, we get that $d(\Gamma) \geqslant 2\tau + 1$ or, in other words, a Goppa code having square-free Goppa polynomial is able to correct at least as many errors as its degree. □

Example 1.4.12. Assume we want to construct a Goppa code $\Gamma(L, G)$ with designed distance $\delta = 3$. For this, we need $\tau = 2$. Let the irreducible polynomial $G(x) = x^2 + x + 1$ be our Goppa polynomial and $L = \mathbb{F}_8 = \{0, 1, \omega, \dots, \omega^6\}$ the support of the code, where ω is a primitive element of the field $\mathbb{F}_8 \cong \mathbb{F}_2 / \langle x^3 + x + 1 \rangle$. Note that the $G(\omega^i) \neq 0\ \forall i$ since the roots of G are in $\mathbb{F}_{2^j}$ with j even. By Thm. 1.4.6 and Rk. 1.4.8, a parity-check matrix for Γ is

$$\mathsf{H} = \mathsf{XY} = \begin{pmatrix} \frac{1}{G(0)} & \frac{1}{G(1)} & \frac{1}{G(\omega)} & \frac{1}{G(\omega^2)} & \cdots & \frac{1}{G(\omega^6)} \\ 0 & \frac{1}{G(1)} & \frac{\omega}{G(\omega)} & \frac{\omega^2}{G(\omega^2)} & \cdots & \frac{\omega^6}{G(\omega^6)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \omega^2 & \omega^4 & \omega^2 & \omega & \omega & \omega^4 \\ 0 & 1 & \omega^3 & \omega^6 & \omega^5 & \omega^5 & \omega^6 & \omega^3 \end{pmatrix},$$

whose version over $\mathbb{F}_2$, by substituting 1 with $(100)^\top$, ω with $(010)^\top$ and ω^2 with $(001)^\top$, is

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

Having transformed Γ into a binary code, by applying THM. 1.4.11 we get $d(\Gamma) \geqslant 5$. Noticing that $(11001011) \in \Gamma$, we have equality.

1.5 Reed-Muller codes

Reed-Muller codes, shorter RM codes, analogously to Goppa codes, are a family of codes for which efficient decoding algorithms are known. Thanks to this property, RM codes were investigated for possible applications in public-key cryptography, an aspect which will be discussed in SECT. 3.3. For most concepts explained in this section we refer to [10] and [26].

Definition 1.5.1 (Boolean monomial and Boolean polynomial [10, 37]). A *Boolean monomial* b is an element of $\mathbb{F}_2[x_1, \ldots, x_m]$ of the form

$$b := x_1^{e_1} x_2^{e_2} \cdots x_m^{e_m},$$

where the e_i are some exponents in $\mathbb{N} \cup \{0\}$. A *Boolean polynomial* is a linear combination of Boolean monomials.

Definition 1.5.2 ([10]). Let $\mathbf{a}$ and $\mathbf{b}$ be two vectors of length ν over any set. Then the *vector multiplication* of $\mathbf{a}$ and $\mathbf{b}$ (which is not to be confused with the dot and the cross products) is defined as

$$\mathbf{a} \star \mathbf{b} = (a_1, a_2, \ldots, a_\nu) \star (b_1, b_2, \ldots, b_\nu) := (a_1 \cdot b_1, a_2 \cdot b_2, \ldots, a_\nu \cdot b_\nu).$$

We will also refer to it as *star product*. We also introduce the notation

$$\mathbf{a}^{\star 2} := \mathbf{a} \star \mathbf{a},$$

where the starred exponent is so as to distinguish the "star-square" from the "dot-square".

Remark 1.5.3. If working over the binaries, the vector multiplication in the ith component of each vector corresponds to the logical operation AND.

Definition 1.5.4 ([10, 37]). A Boolean polynomial B is said to be in *reduced form* (in symbols: $\tilde{B}$) when to its monomials the following rules are applied:

i. $x_i x_j = x_j x_i$;

ii. $x_i^2 = x_i$

and adding the resulting *reduced monomials* until they are all distinct. Rule (i) is justified by the commutativity of the ring $\mathbb{F}_2[x_1, \ldots, x_m]$. In the second case, whenever working over the binary field we only face the multiplications $0^2 = 0$ and $1^2 = 1$; thus, for every vector $\mathbf{a} \in \mathbb{F}_2^n$ we have $\mathbf{a}^{\star 2} = \mathbf{a}$, implying that all the exponents $e_i > 0$ simply have to be set to 1. From this we can define the *degree of a Boolean monomial* as the degree of its reduced form, which corresponds to the number of variables appearing in it. Consequently, the *degree of a Boolean polynomial* is as usual defined as the highest degree among its monomials.

The two rules can be easily resumed in form of equivalence relations. Thus we are able to introduce the following notation:

Definition 1.5.5. Given the polynomial ring $\mathbb{F}_2[x_1, \ldots, x_m]$ and the equivalence relations $x_1^2 \sim x_1, \ldots, x_m^2 \sim x_m$, we define the quotient ring

$$\boldsymbol{B}_m := \mathbb{F}_2[x_1, \ldots, x_m] \Big/ \left\langle x_1^2 + x_1, \ldots, x_m^2 + x_m \right\rangle$$

as the set of Boolean polynomials in reduced form.

Some authors, instead of defining a general and a reduced form for Boolean polynomials, accept the denomination "polynomials" only for the elements of $\boldsymbol{B}_m$ and not for the ones in general form as presented in Def. 1.5.1. Anyway, from now on, each time Boolean monomials or polynomials will be mentioned, their reduced form as in Def. 1.5.5 will be implicitly assumed.

After having introduced the concept of Boolean polynomials, in order to build codes with their help it is necessary to associate a binary string to each of them. We notice that the ring $\boldsymbol{B}_m$ for any m contains 2^m distinct monomials, since this is equal to the cardinality of the power set of the m variables, in symbols $\#\mathcal{P}(\{x_1, \ldots, x_m\})$. For the same reason, one can easily say that $\#\mathcal{P}\big(\mathcal{P}(\{x_1, \ldots, x_m\})\big) = 2^{2^m}$ different polynomials can be built. Hence, one of the easiest ways to do that is to find a bijection between $\boldsymbol{B}_m$ and the set of binary strings of length 2^m. For this, we need a mapping between each monomial in $\boldsymbol{B}_m$ and a vector in $\mathbb{F}_2^{2^m}$.

Let ψ be a function mapping the variables x_i to a binary vector of size 2^m. We associate the elements 0 and 1 in $\boldsymbol{B}_m$ to the all-zero vector $\mathbf{0} = (0 \ldots 0)$ and to the all-one vector $\mathbf{1} = (1 \ldots 1)$ respectively, or in other words: $\psi(0) = \mathbf{0}$ and $\psi(1) = \mathbf{1}$. The variables (i.e. the degree-one monomials) $x_1, \ldots, x_m$ are mapped to alternate patterns of 0's and 1's of equal length, for example in the following way [37]:

$$\begin{aligned}
\psi(x_1) &:= (00 \ldots 011 \ldots 1) && \text{(two } 2^{m-1}\text{-patterns)} \\
\psi(x_2) &:= (0 \ldots 01 \ldots 10 \ldots 01 \ldots 1) && \text{(four } 2^{m-2}\text{-patterns)} \\
&\vdots \\
\psi(x_i) &:= (\underbrace{0 \ldots 0}_{2^{m-i}\times}\underbrace{1 \ldots 1}_{2^{m-i}\times} 0 \ldots 1 \underbrace{0 \ldots 0}_{2^{m-i}\times}\underbrace{1 \ldots 1}_{2^{m-i}\times}) && (2^i \ 2^{m-i}\text{-patterns}) \\
&\vdots \\
\psi(x_m) &:= (0101 \ldots 0101) && (2^m \text{ single bits})
\end{aligned} \tag{1.9}$$

Now we have enough material to define the general case involving polynomials. Let $\tilde{B}(\mathbf{x}) = \tilde{B}(x_1, \ldots, x_m) = b_1 + b_2 + \ldots + b_s$ be the Boolean polynomial in reduced form composed of some monomials b_i, $i = 1, \ldots, s$, $s \leqslant 2^m$ (otherwise $\tilde{B}$ would not be in reduced form). To compute the vectors associated with each of the s monomials, first substitute each variable appearing in them through the corresponding binary strings, following the mapping seen in (1.9); then, execute the star product as shown above. The vector representing B in the binary field is given as the sum modulo 2 of the resulting s binary strings.

Example 1.5.6. Let $m = 4$ and $B(x_1, x_2, x_3, x_4) = 1 + x_2^3 + x_2^6 x_4^2 + x_2 x_4^2 + x_1 x_2^5 x_3^3$. This means that the associated vectors will have length $2^4 = 16$. We begin by bringing the polynomial into reduced form:

$$\tilde{B} = 1 + x_2 + x_1 x_2 x_3.$$

Notice that both monomials containing x_4 disappeared, so we have to turn to vector multiplication only inside the rightmost monomial. From (1.9) we already know that 1 maps to $\mathbf{1}$

and x_2 is associated with (0000 1111 0000 1111). We now calculate the missing information:

$$\psi(x_1x_2x_3) = (\underbrace{0\ldots0}_{8\times}\underbrace{1\ldots1}_{8\times}) \star (0000\ 1111\ 0000\ 1111) \star (0011\ 0011\ 0011\ 0011) = (\underbrace{0\ldots0}_{14\times}11),$$

hence the vector associated with B is

$$\psi(B) = \psi(\tilde{B}) = \psi(1) + \psi(x_1) + \psi(x_1x_2x_3) = (1111\ 0000\ 1111\ 0011).$$

Instead of polynomials, it is also possible to associate a special class of functions, called Boolean functions, to the elements of $\mathbb{F}_2^{2^m}$. Therefore, we postpone the definition of RM codes for a few lines to introduce the concept of Boolean functions.

Definition 1.5.7 (Boolean function [37]). A *Boolean function* is a function $f\colon \mathbb{F}_2^\nu \longrightarrow \mathbb{F}_2$ for some natural number ν. We call the set of Boolean functions in ν variables $\mathcal{B}_\nu$. Together with the operations "+" and "·" (i.e. "XOR" and "AND"), where

$$(f+g)(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) \quad \text{and} \quad (f\cdot g)(\mathbf{x}) := f(\mathbf{x})\cdot g(\mathbf{x}), \quad f,g \in \mathcal{B}_\nu,\ \mathbf{x} \in \mathbb{F}_2^\nu,$$

it forms a ring.

Example 1.5.8. $f(x_1,x_2,x_3,x_4) = a_0 + a_1x_2 + a_2x_1x_2x_3 + a_3x_2x_4$ with $a_0,\ldots,a_3 \in \mathbb{F}_2$ is a Boolean function.

Boolean functions are based on finite sets, hence it is possible to give a list of all elements in the domain and their images. This can be easily done through a *truth table* [37], of which we give an example.

Example 1.5.9. Let $m = 3$ and $\mathbf{x} = (x_1,x_2,x_3)$; then the vectors associated with the three degree-one monomials have length 8. Here we set f and g to be respectively $f(\mathbf{x}) = x_1+x_2+x_3$ and $g(\mathbf{x}) = 1 + x_2 + x_1x_2 + x_1x_3$.

$\psi(x_1)$	0	0	0	0	1	1	1	1
$\psi(x_2)$	0	0	1	1	0	0	1	1
$\psi(x_3)$	0	1	0	1	0	1	0	1
$f(\mathbf{x})$	0	1	1	0	1	0	0	1
$g(\mathbf{x})$	1	1	0	0	1	0	1	0

Table 1 – Truth table of the Boolean functions f and g.

By listing the vectors $\psi(x_i)$ in the truth table, we see that if we read the components vertically, all possible 2^m combinations for m elements of $\mathbb{F}_2$ are indeed reached, i.e. the whole set $\mathbb{F}_2^m$ is represented. Moreover, thanks to the chosen sequence for $\psi(x_i)$, the length-m vectors are in lexicographic order.[3] The use of such tables gives us a hint about how to put $\mathcal{B}_m$ in one-to-one correspondence with the set of binary strings of length 2^m: it suffices to represent every $f \in \mathcal{B}_m$ through a list of its images in vector form, which will have the requested size. This leads to the next

[3] See Sect. 2.2 for the definition of this concept.

Theorem 1.5.10. *The mapping* $\varphi\colon \mathcal{B}_m \longrightarrow \mathbb{F}_2^{2^m}$ *is a ring isomorphism.*

Proof. Let $f, g \in \mathcal{B}_m$. Let $\psi(\mathbf{x})$ be a short cut to denote all binary vectors associated with the variables $x_1, \ldots, x_m$. With a slight abuse of notation, we represent the functions as vectors containing all their images, i.e.

$$\varphi(f) = f \circ \psi(\mathbf{x}) := f \left(\begin{array}{c|c|c|c|c|c|c|c|c} 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & \cdots & 0 & 1 & 0 & 1 \end{array}\right) \begin{array}{l} \leftarrow \psi(x_1) \\ \\ \leftarrow \psi(x_{m-1}) \\ \leftarrow \psi(x_m) \end{array} := \tag{1.10}$$

$$:= \big(f(\underbrace{00\ldots00}_{=(0)_2}), f(\underbrace{00\ldots01}_{=(1)_2}), \ldots, f(\underbrace{11\ldots11}_{=(2^m-1)_2})\big),$$

where $(\cdot)_2$ denotes the binary representation of the natural number in brackets.

(a) *Homomorphism:*

$$\mathcal{B}_m \ni \varphi(f+g) = (f+g)\circ\psi(\mathbf{x}) = f\circ\psi(\mathbf{x}) \oplus g\circ\psi(\mathbf{x}) \overset{(1.10)}{=} \begin{pmatrix} f(0\ldots00) \\ f(0\ldots01) \\ \vdots \\ f(1\ldots11) \end{pmatrix}^\top \oplus \begin{pmatrix} g(0\ldots00) \\ g(0\ldots01) \\ \vdots \\ g(1\ldots11) \end{pmatrix}^\top \in \mathbb{F}_2^{2^m}$$

and analogously

$$\mathcal{B}_m \ni (f \cdot g) \circ \psi(\mathbf{x}) = \big(f \circ \psi(\mathbf{x})\big) \cdot \big(g \circ \psi(\mathbf{x})\big) \overset{(1.10)}{=} \begin{pmatrix} f(0\ldots00) \\ f(0\ldots01) \\ \vdots \\ f(1\ldots11) \end{pmatrix}^\top \star \begin{pmatrix} g(0\ldots00) \\ g(0\ldots01) \\ \vdots \\ g(1\ldots11) \end{pmatrix}^\top \in \mathbb{F}_2^{2^m}.$$

(b) *Bijection:* Already from the right hand side of (1.10) we can determine the cardinality of $\mathcal{B}_m$. More formally: a Boolean function f in m variables maps $\mathbb{F}_2^m$ to $\mathbb{F}_2$; hence, by the well-known property of cardinalities of sets saying that there are $\#Y^{\#X}$ functions $F\colon X \to Y$, it is clear that the set $\mathcal{B}_m$ containing all such functions has 2^{2^m} elements, exactly like the set of binary strings of length 2^m.

From the fact that there exists exactly one function mapping each coordinate to 0, called the zero-function $0(\mathbf{x})$, follows that the kernel of φ contains only one element. This implies the injectivity of φ and hence, as both $\mathcal{B}_m$ and $\mathbb{F}_2^{2^m}$ are finite of the same cardinality, also its bijectivity. □

In addition, as it can be gathered from the first half of the current section, $\mathcal{B}_m$ and $\boldsymbol{B}_m$ are isomorphic [37] and consequently so are $\boldsymbol{B}_m$ and $\mathbb{F}_2^{2^m}$. This means that Boolean functions, reduced Boolean polynomials and binary strings can be used interchangeably, as we are able to move along of the isomorphisms φ and ψ. We get now to the main part of this section.

Definition 1.5.11 (Reed-Muller code [37, p. 270]**).** Let ρ be an integer such that $0 \leqslant \rho \leqslant m$. Then the ρth *order Reed-Muller code*, denoted $\mathsf{RM}(\rho, m)$, is defined as the set of all binary vectors of length $n = 2^m$ associated with Boolean functions $f(x_1, \ldots, x_m)$ — or equivalently with Boolean polynomials $B(x_1, \ldots, x_m)$ — having degree at most ρ.

Theorem 1.5.12. *RM codes are* $\left[2^m, \sum_{i=0}^{\rho}\binom{m}{i}, 2^{m-\rho}\right]$ *codes.*

Proof. (a) *Linearity:* Boolean polynomials of degree at most ρ are constructed with all possible linear combinations of monomials of degree at most ρ. Clearly, also the vectors $\mathbf{0}$ and $\mathbf{1}$ are included in each code. Thus, RM codes are linear codes.

(b) *Dimension:* From (a) we deduce a way to build a generator matrix for each $\mathsf{RM}(\rho, m)$ code. As all monomials $1, x_1, \ldots, x_m, x_1x_2, \ldots, x_{m-1}x_m, \ldots, x_1x_2\cdots x_m$ are linearly independent, we can take the vectors associated with them and list them as rows of a matrix, making sure to choose only the ones composed by a number of variables $\leqslant \rho$. In this way we get:

$$\mathsf{G} = \begin{pmatrix} \_\_\_\_\_ & \mathbf{1} & \_\_\_\_\_ \\ \_\_\_\_\_ & \psi(x_1) & \_\_\_\_\_ \\ \_\_\_\_\_ & \psi(x_2) & \_\_\_\_\_ \\ & \vdots & \\ \_\_\_\_\_ & \psi(x_m) & \_\_\_\_\_ \\ \_\_\_\_\_ & \psi(x_1x_2) & \_\_\_\_\_ \\ & \vdots & \\ \_\_\_\_\_ & \psi(x_{i_1}x_{i_2}\cdots x_{i_\rho}) & \_\_\_\_\_ \end{pmatrix},$$

where $1 \leqslant i_j \leqslant m$ and $1 \leqslant j \leqslant \rho$. This implies directly that the dimension of a $\mathsf{RM}(\rho, m)$ code is

$$k := k(\rho) = 1 + m + \binom{m}{2} + \ldots + \binom{m}{\rho} = \sum_{i=0}^{\rho}\binom{m}{i}.$$

(c) *Distance:* We can already exclude two special cases from the rest: $\rho = 0$ and $\rho = m$. When $\rho = 0$ we have the repetition code, which has distance equal to the length. When $\rho = m$, the code is all $\mathbb{F}_2^n$, implying that $d(\mathsf{RM}(m, m)) = 1$ (see also Ex. 1.5.14, parts (i) and (ii)). The proof of the general case requires some extra theory, hence we will just give an outline. The reasoning is based on the possibility of building recursively the $\mathsf{RM}(\rho + 1, m + 1)$ code starting from $\mathsf{RM}(\rho, m)$ and $\mathsf{RM}(\rho + 1, m)$ [29, 37]:

$$\mathsf{RM}(\rho + 1, m + 1) = \left\{(\mathbf{c} \mid \mathbf{c} + \mathbf{d}) \;\middle|\; \mathbf{c} \in \mathsf{RM}(\rho + 1, m) \wedge \mathbf{d} \in \mathsf{RM}(\rho, m)\right\},$$

where $(\cdot \mid \cdot)$ represents a single vector made of two half vectors. Then, the distance is proved by induction starting from the base case $\mathsf{RM}(0, 1)$. More intuitively, note that for any monomial b of degree δ the associated codeword $\psi(b)$ has weight $2^{m-\delta}$. Let $\delta = \rho$, hence the highest value possible in the code. The rows of G generated by monomials of the same degree and their linear combinations have the same weight, which is the lowest in the code. As $\mathrm{wt}(\psi(b)) = 2^{m-\rho}$, the claim follows. □

Remark 1.5.13. With the help of the form for a generator matrix given in part (b) of the proof of Thm. 1.5.12, one can also easily find out that every $\mathsf{RM}(\rho, m)$ code contains all $\mathsf{RM}(\tilde{\rho}, m)$ codes $\forall \tilde{\rho} \leqslant \rho$.

Example 1.5.14.

i. When $\rho = 0$, we have $\mathsf{RM}(0, m) = \{\mathbf{0}, \mathbf{1}\}$, which is the repetition code of length 2^m. Its generator matrix is $\mathsf{G} = (\mathbf{1})$.

ii. When $\rho = m$, we have all monomials of degree $\leqslant m$ represented in the basis of the code, corresponding to $\sum_{i=0}^{m} \binom{m}{i} = 2^m = n$ vectors in the generator matrix. Consequently, $\mathsf{RM}(m, m) = \mathbb{F}_2^n$ and G is composed of all vectors from $\mathbf{1}$ to $\psi(x_1 x_2 \cdots x_m)$.

iii. $\mathsf{RM}(0, 0) = \mathbb{F}_2$, which is consequent with the previous two examples. $\boldsymbol{B}_0 = \mathbb{F}_2$ either.

iv. Consider the following matrix:

$$\mathsf{G} = \left(\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \begin{array}{l} \leftarrow \psi(1) \\ \leftarrow \psi(x_1) \\ \leftarrow \psi(x_2) \\ \leftarrow \psi(x_3) \\ \leftarrow \psi(x_1 x_2) \\ \leftarrow \psi(x_1 x_3) \\ \leftarrow \psi(x_2 x_3) \\ \leftarrow \psi(x_1 x_2 x_3) \end{array}$$

The dashed lines are meant for dividing the different codes that can be constructed with $m = 3$: $\mathsf{RM}(0, 3)$ with only the first row, $\mathsf{RM}(1, 3)$ with the upper half of G, $\mathsf{RM}(2, 3)$ with rows 1–7 and $\mathsf{RM}(3, 3)$ with the whole matrix.

Chapter 2

Useful tools

In this chapter we investigate some topics which will be useful in the next chapters. First, we perform a reasoning which will lead us to a result showing one of the strengths of the McEliece cryptosystem, consisting in the big amount of polynomials one has available to define a Goppa code (see also SECT. 4.2). After that, we present a corrected version of an algorithm enabling the use of the Niederreiter cryptosystem. The last section is dedicated to a simple procedure for finding the systematic form of a matrix.

2.1 Counting irreducible polynomials

The goal of this section is to discover whether there exists a relationship between the number of monic irreducible polynomials of some degree t over a finite field $\mathbb{F}_q$ and the total number of monic polynomials over the same field. We start with some preliminary steps involving properties of irreducible polynomials.

Lemma 2.1.1 ([23, p. 48]**).** *Let $f \in \mathbb{F}_q[x]$ be an irreducible polynomial of degree s and let t be a natural number. Then:*

$$f \,\Big|\, (x^{q^t} - x) \iff s \mid t\,.$$

Proof. "$\Longrightarrow$": Let α be a root of f in its splitting field. As $f \mid g := x^{q^t} - x$, $\alpha^{q^t} - \alpha = 0 \Longrightarrow \alpha^{q^t} = \alpha$, so $\alpha \in \mathbb{F}_{q^t}$. This means that $\mathbb{F}(\alpha) \subseteq \mathbb{F}_{q^t}$ as a field. We know that $\mathbb{F}_q(\alpha) \cong \mathbb{F}_q[x]/(f)$, which means that $\mathbb{F}_q(\alpha)$ is an extension of degree s over $\mathbb{F}_q$. As $\left[\mathbb{F}_{q^t} : \mathbb{F}_q\right] = t \Longrightarrow s \mid t$.

"$\Longleftarrow$": Assume $s \mid t$. Then, $\mathbb{F}_{q^s}$ is a subfield of $\mathbb{F}_{q^t}$. As above, let α be an arbitrary root of f in its splitting field $\Longrightarrow \left[\mathbb{F}(\alpha) : \mathbb{F}_q\right] = s \Longrightarrow \mathbb{F}_q(\alpha) \cong \mathbb{F}_{q^s}$. As $\mathbb{F}_q(\alpha)$ is a subfield of $\mathbb{F}_{q^t}$, we have that $\alpha \in \mathbb{F}_{q^t}$, which implies $\alpha^{q^t} = \alpha$, hence α is also a root of g. By THM. 1.2.10 we know that all roots of f are distinct (this is actually their minimal polynomial). Thus, we have that $f \mid g$. □

An immediate consequence of this Lemma is the following

Corollary 2.1.2 ([23, 43]). *Let q and t as above. Then the polynomial $g(x) = x^{q^t} - x$ is given by the multiplication of all monic irreducible polynomials over $\mathbb{F}_q$ whose degrees are divisors of t.*

Proof. From the previous Lemma we know that g has as factors all irreducible polynomials $f_1, \ldots, f_s$ whose degrees divide t and does not admit any other. Obviously, the f_k's are also monic. It only remains to be shown that g splits in linear factors over $\mathbb{F}_{q^t}$ to make sure that $f_k^2 \nmid g$ $\forall k$. For this it is enough to notice that $g'(x) = q^t x^{q^t-1} - 1 = -1$: no roots of the formal derivative (see Def. 1.4.9) implies no multiple roots of g. □

This result can be summarized by the following formula:

$$x^{q^t} - x = \prod_{\substack{\deg f | t, \\ f \text{ monic}}} f(x), \tag{2.1}$$

where f ranges over all irreducible polynomials over $\mathbb{F}_q$.

Having found the factorization of $x^{q^t} - x$ over $\mathbb{F}_q$, we can now concentrate on the degrees of each polynomial. From Lemma 2.1.1 we know that for every divisor s of t there exists a certain number of monic irreducible polynomials, which we will call $N_q^I(s)$ (the function-like notation was not chosen at random, as we are going to see later). The right hand side of equation (2.1) gives us the value of $N_q^I(s)$ $\forall s$ and thanks to the left hand side, we can get their weighted sum:

$$q^t = \sum_{s|t} s N_q^I(s) \tag{2.2}$$

Before going to the next step, we need to introduce a special function.

Definition 2.1.3 (Möbius function). Let $t \in \mathbb{N}$. Then the *Möbius function* $\mu(t)$ is defined as:

$$\mu(t) := \begin{cases} 1 & \text{if } t = 1; \\ (-1)^\ell & \text{if } t = p_1 \cdots p_\ell \text{ with } p_i \text{ primes and } p_i \neq p_j \text{ for } i \neq j; \\ 0 & \text{else.} \end{cases}$$

Remark 2.1.4. The Möbius function is an arithmetic function, that is it admits as arguments natural numbers, and is multiplicative for coprime arguments, i.e.

$$\mu(t_1 t_2) = \mu(t_1)\mu(t_2) \quad \text{if} \quad \gcd(t_1, t_2) = 1.$$

Theorem 2.1.5 (Möbius inversion formula [21]). *Let f and F be arithmetic functions such that*

$$F(t) = \sum_{d|t} f(d).$$

Then the function f can be obtained through:

$$f(t) = \sum_{d|t} \mu\left(\tfrac{t}{d}\right) F(d).$$

Now we have enough material to introduce one of the main results of this section:

Theorem 2.1.6 ([43]). *The number $N_q^I(t)$ of monic irreducible polynomials of degree t over $\mathbb{F}_q$ is*

$$N_q^I(t) = \frac{1}{t} \sum_{d|t} \mu\left(\tfrac{t}{d}\right) q^d.$$

Proof. This statement follows immediately from the application of Möbius inversion formula to equation (2.2), more precisely, by defining the two functions from THM. 2.1.5 as $F(t) = q^t$ and $f(t) = t \cdot N_q^I(t)$ respectively. □

Example 2.1.7. In the case $q = 2$:

i. $N_2^I(1) = 2\mu(1) = 2 \qquad (\longleftrightarrow\ x,\, x+1)$

ii. $N_2^I(4) = \frac{1}{4}\left(2^1\mu(4) + 2^2\mu(2) + 2^4\mu(1)\right) = \frac{0-4+16}{4} = 3$
$(\longleftrightarrow\ x^4+x+1,\, x^4+x^3+1,\, x^4+x^3+x^2+x+1)$

Proposition 2.1.8. *The total number and the number of monic but not necessarily irreducible polynomials of degree t over $\mathbb{F}_q$ are given respectively by*

$$N_q^T(t) = q^t\,(q-1) \quad \text{and} \quad N_q^M(t) = q^t\,. \tag{2.3}$$

We now shortly restrict our attention to the binary case; in this way, (2.3) reduces to

$$N_2^T(t) = N_2^M(t) = 2^t\,.$$

To reach our goal, we evaluate N_2^I and N_2^M in function of some t, reporting the results in TAB. 2 and inserting into a third column their quotients.

t	N_2^I	N_2^M	$\frac{N_2^M}{N_2^I}$	t	N_2^I	N_2^M	$\frac{N_2^M}{N_2^I}$
1	2	2	1	12	335	4096	$\sim$12,227
2	1	4	4	13	630	8192	$\sim$13,003
3	2	8	4	14	1161	16384	$\sim$14,112
4	3	16	$5,\overline{3}$	15	2182	32768	$\sim$15,017
5	6	32	$5,\overline{3}$	16	4080	65536	$\sim$16,063
6	9	64	$7,\overline{1}$	$\cdots$			
7	18	128	$7,\overline{3}$	20	52377	1'048'576	$\sim$20,02
8	30	256	$8,5\overline{3}$	$\cdots$			
9	56	512	$9,\overline{142857}$	30	35'790'267	$\sim 1,074 \cdot 10^9$	$\sim$30,001
10	99	1024	$10,\overline{34}$	$\cdots$			
11	186	2048	$\sim$11,011	50	$\sim 2,252 \cdot 10^{13}$	$\sim 1,126 \cdot 10^{15}$	$\sim$50,000001

Table 2 – Values of N_2^I, N_2^M and $\frac{N_2^M}{N_2^I}$ for some natural numbers.

We notice that already for most of the first natural numbers, the values of t and $\frac{N_2^M}{N_2^I}$ differ only in the fractional part. The inequality decreases rapidly, but not monotonically; therefore, the connection between N_2^M and N_2^I needs to be verified with some more attention.

From the formula for the computation of N_2^I we see that the term 2^t appears for any t, as $d_1 = 1$ is always a divisor. Another point which all t have in common is the behaviour of N_2^I with the second-lowest divisor of t, say d_2, which has to be a prime number; thus, $\mu(d_2) = -1$. Being d_2 prime, $2^{\frac{t}{d_2}} \leqslant 2^{\frac{t}{2}}$, returning a much smaller value than the former one. Consequently,

any remaining term will be even less influential. Hence, it is possible to give an approximate value of $\frac{N_2^I}{N_2^M}$:

$$\frac{N_2^I(t)}{N_2^M(t)} = \frac{\frac{1}{t}\Big(2^t - \overbrace{2^{\frac{t}{d_2}} + - \dots}^{\ll 2^t \text{ for } t \text{ increasing}}\Big)}{2^t} \cong \frac{1}{t},$$

which confirms the validity of the hypothesis made by reading the data from Tab. 2. We now try extend this result, conjecturing that $\frac{1}{t}$ is also the ratio of irreducible polynomials among all monic polynomials over *any* field $\mathbb{F}_q$:

Lemma 2.1.9 ([14]).

$$\lim_{t\to\infty}\left(\frac{N_q^I(t)}{N_q^M(t)} - \frac{1}{t}\right) = 0.$$

Proof. From Thm. 2.1.6 and using the same notation as in the reasoning performed to get (2.1), we have that

$$N_q^I(t) = \frac{1}{t}\Big(q^t - q^{\frac{t}{d_2}} + - \dots\Big) \Longrightarrow \left|N_q^I(t) - \frac{q^t}{t}\right| = \frac{1}{t}\left|\sum_{\substack{d|t,\\ d<t}} \mu\left(\tfrac{t}{d}\right) q^d\right| \leqslant$$

$$\leqslant \frac{1}{t}\sum_{\substack{d|t,\\ d<t}} \left|\mu\left(\tfrac{t}{d}\right) q^d\right| = \frac{1}{t}\sum_{d=1}^{\lfloor \frac{t}{2}\rfloor} q^d \leqslant \frac{1}{t}\frac{t}{2}q^{\frac{t}{2}} = \frac{q^{\frac{t}{2}}}{2}.$$

Hence:

$$\left|\frac{N_q^I(t)}{N_q^M(t)} - \frac{1}{t}\right| = \frac{1}{q^t}\left|N_q^I(t) - \frac{q^t}{t}\right| \leqslant \frac{q^{\frac{t}{2}}}{2q^t} = \frac{1}{2q^{\frac{t}{2}}} \xrightarrow{t\to\infty} 0.$$

□

2.2 Constant weight encoding

During the thesis we will encounter some topics where vectors of predetermined weight are needed. The main example is the Niederreiter cryptosystem, presented in Sect. 3.2. Clearly, if one would simply cut a plaintext in strings of length n, he cannot expect all of them to have an amount of non-zero coordinates not exceeding the desired limit, so an algorithm is needed to convert text fragments into codewords which can be accepted by the cryptosystem. Moreover, as we will see in Sect. 4.4, this has the drawback that its running time must be taken into account for evaluating the efficiency of the entire cryptosystem. This procedure is sometimes called *constant weight encoding*. For this, we introduce the notation $\mathcal{W}_{n,t} \subseteq \mathbb{F}_2^n$ to denote the set of binary words having length n and weight t. This set contains $\binom{n}{t}$ elements. If we wanted to cut the plaintext into blocks of same length, say ℓ, we would actually only need a subset $\mathcal{U}_{n,t}$ of $\mathcal{W}_{n,t}$ whose cardinality is 2^ℓ, i.e. the same of the domain of the map (or the input set of the algorithm) we are looking for. To transmit as much information as possible in one go, the message section length should be chosen as $\ell = \max\{\lambda \mid 2^\lambda \leqslant \#\mathcal{W}_{n,t}\} = \left\lfloor \log_2\binom{n}{t}\right\rfloor$. Before citing some examples, we need to recall a particular concept of ordering.

Definition 2.2.1 (Lexicographic ordering [9, 11]).

i. Let $\nu \in \mathbb{N}$. Two elements $\mathbf{a}$ and $\mathbf{b}$ of $\mathbb{N}^\nu$ are said to be *lexicographically ordered* as $\mathbf{a} < \mathbf{b}$ if $\exists \kappa_0 < \nu$ such that $a_\kappa = b_\kappa \ \forall \kappa < \kappa_0$ and $a_{\kappa_0} < b_{\kappa_0}$.

ii. Let A be some subset of $\mathbb{N}^\nu$ and $\mathbf{x} \in A$, then the *lexicographic index* of the string $\mathbf{x}$ will be indicated as the unique integer $I_A(\mathbf{x}) \in \mathbb{N} \cup \{0\}$, or shortly $I(\mathbf{x})$, such that if all elements of A are listed in lexicographic order, $I(\mathbf{x})$ represents the position of $\mathbf{x}$ inside that list.

Example 2.2.2. With $\mathbf{a}$ and $\mathbf{b}$ as in (i), it would be $I(\mathbf{a}) < I(\mathbf{b})$.

An algorithm working as described was proposed by Guillot and is reported, sadly with some mistakes, in [12] and [32, p. 99]. It is based on the lexicographic ordering in the space $\mathbb{F}_2^\ell$ of message fragments and should allow a mapping into the set of codewords $\mathcal{W}_{n,t}$. In Alg. 1 we present a corrected and slightly modified version of Guillot's unpublished result. To be more precise, it produces a bijection between $\mathbb{F}_2^\ell$ and $\mathcal{U}_{n,t}$ by transforming constant length strings $\mathbf{x} \in \mathbb{F}_2^\ell$ into message words $\mathbf{m} \in \mathcal{U}_{n,t}$ and vice versa. This is true thanks to the inversion of the components of the output string $\mathbf{m}$, so that the algorithm now respects the lexicographic ordering in its output set. As both $\mathcal{W}_{n,t}$ and $\mathcal{U}_{n,t}$ are ordered after the same logic, the latter contains the first 2^ℓ elements of the former, implying that an element $\mathbf{x} \in \mathbb{F}_2^\ell$ is associated with the only $\mathbf{m} \in \mathcal{U}_{n,t}$ for which it is $I_{\mathbb{F}_2^\ell}(\mathbf{x}) = I_{\mathcal{U}_{n,t}(\mathbf{m})}$. The idea lying behind the algorithm will appear clearer after having described its inverse proceedings.

Retrieving the plaintext after having decoded the received message only requires some easy calculations. The inverse of the algorithm computes the lexicographic index of a given element $\mathbf{m} \in \mathcal{U}_{n,t}$ (this is actually possible in the whole $\mathcal{W}_{n,t}$). The conversion into the sent string $\mathbf{x}$ is now immediate, as this number is not only the index of $\mathbf{x}$ but, because $\mathbb{F}_2^\ell$ is the set containing *all* binary strings of length 4, it also corresponds to its representation in the binary system. Concretely:

Proposition 2.2.3 ([11]). *Let $\mathbf{m} = (m_1 m_2 \dots m_n)$ be a string of length n and weight t. If $I(\mathbf{m})$ denotes the lexicographic index of $\mathbf{m}$ in $\mathcal{U}_{n,t}$ (resp. $\mathcal{W}_{n,t}$), then $I(\mathbf{m})$ can be found through:*

$$I(\mathbf{m}) = \sum_{i=1}^{n} m_i \binom{n-i}{\sum_{j=i}^{n} m_j} = \binom{n-i_1}{t} + \binom{n-i_2}{t-1} + \dots + \binom{n-i_t}{1},$$

where $m_{i_1}, \dots, m_{i_t}$ are the positions of the 1's in $\mathbf{m}$.

What Alg. 1 does is basically deciding whether the current binomial coefficient, represented by the letter c', is "small enough" to be inserted into the summation introduced in Prop. 2.2.3. We define $c := \binom{n}{t}$, i.e. as the cardinality of $\mathcal{W}_{n,t}$. The first step is to reduce this number to $\binom{n-1}{t}$ — or $\binom{j-1}{t}$, as here it is still $n = j$ — and call it c' (see line 7). If c' is strictly bigger than the index i of the given string $\mathbf{x} \in \mathbb{F}_2^\ell$, the **if**-part will reject it by assigning to the related component m_1 the value 0 (line 9); otherwise it will be accepted by giving it the value 1 (line 12). In the case it is refused, the second execution of the **while**-loop retries the same task with a smaller coefficient, i.e. $\binom{j-2}{t}$. If instead we have $i \geqslant c'$ and hence $m_1 = 1$, the algorithm requires new values for i, c and t. Let $\tilde{\mathbf{m}} := (m_2 m_3 \dots m_n)$ be the remaining

Algorithm 1 – Corrected and modified version of Guillot's algorithm for transforming constant length strings into encodable words for the Niederreiter cryptosystem

1: ***Input:*** $n, t \in \mathbb{N}$ with $t \leqslant n$
2: $i := I(\mathbf{x})$ for an $\mathbf{x} \in \mathbb{F}_2^\ell$
3: ***Output:*** $\mathbf{m} = (m_1 m_2 \dots m_n) \in \mathcal{U}_{n,t}$

4: $c \longleftarrow \binom{n}{t}$
5: $j \longleftarrow n$
6: **while** $j > 0$ **do**
7: $c' \longleftarrow c \cdot \frac{j-t}{j}$
8: **if** $i < c'$ **then**
9: **print** $m_{n-j+1} = 0$
10: $c \longleftarrow c'$
11: **else**
12: **print** $m_{n-j+1} = 1$
13: $i \longleftarrow i - c'$
14: $c \longleftarrow c \cdot \frac{t}{j}$
15: $t \longleftarrow t - 1$
16: **end if**
17: $j \longleftarrow j - 1$
18: **end while**

part of $\mathbf{m}$: then it is clear that $\tilde{\mathbf{m}} \in \mathcal{W}_{j-1,t-1}$, thus i in the next execution of the cycle will stay for the index $I(\tilde{\mathbf{m}})$ in that set (line 13). In order to find the next component, c must be reduced to $\binom{j-1}{t-1} = \#\mathcal{W}_{n-1,t-1}$ (line 14). Letting t and j take respectively the values $t-1$ and $j-1$, all parameters now assume the forms they had before running the algorithm.

Example 2.2.4. (a) Imagine Alice needs to convert a plaintext into words with length $n = 6$ and weight $t = 3$, thus she starts by determining the length $\ell = \left\lfloor \log_2 \binom{6}{3} \right\rfloor = 4$. This tells her that she will have to cut her plaintext into fragments of length 4, i.e. half bytes or *nibbles*, and then substitute each of them through a block of length 6 but with fixed weight, so that they can be decoded by a 3-error correcting Niederreiter scheme.

(b) The encoding procedure requires at this point the transformation of the plaintext $(1101) \in \mathbb{F}_2^4$ into a string $\mathbf{m} = (m_1 m_2 \dots m_6) \in \mathcal{U}_{6,3}$. In the binary system, $(1101)_2$ corresponds to $(13)_{10}$, where the indices 2 and 10 indicate the binary and the decimal representation of natural numbers; hence $I(1101) = 13$. Thus the initial values are $c = 20$, $j = 6$ and $i = 13$.

The resulting vector is the one highlighted in the grey column of Tab. 3: $\mathbf{m} = (101001) \in \mathcal{U}_{6,3}$. The message is now ready to be encoded and sent to Bob.

(c) Bob retrieved $\mathbf{m}$ through a decoding algorithm. To be able to read the plaintext $\mathbf{x}$ he

$j = 6$:	$c' = 20 \cdot \frac{6-3}{6} = 10$	$13 \geqslant 10$	$\Longrightarrow$	$m_1 = 1$	$i = 13 - 10 = 3$	$c = 20 \cdot \frac{3}{6} = 10$	$t = 2$
$j = 5$:	$c' = 6$	$3 < 6$	$\Longrightarrow$	$m_2 = 0$		$c = 6$	
$j = 4$:	$c' = 3$	$3 \geqslant 3$	$\Longrightarrow$	$m_3 = 1$	$i = 0$	$c = 3$	$t = 1$
$j = 3$:	$c' = 2$	$0 < 2$	$\Longrightarrow$	$m_4 = 0$		$c = 2$	
$j = 2$:	$c' = 1$	$0 < 1$	$\Longrightarrow$	$m_5 = 0$		$c = 1$	
$j = 1$:	$c' = 0$	$0 \geqslant 0$	$\Longrightarrow$	$m_6 = 1$	$(i = 0)$	$(c = 1)$	$(t = 0)$

Table 3 – Guillot's algorithm applied to the binary string (1101).

now has to compute

$$I_{\mathbb{F}_2^4}(\mathbf{x}) = I_{\mathcal{U}_{6,3}}(\mathbf{m}) = \sum_{j=1}^{6} m_j \binom{6-j}{\sum_{k=j}^{6} m_k} = \binom{5}{3} + \binom{3}{2} + \binom{0}{1} = 13,$$

from which he easily recovers the binary form of 13: $\mathbf{x} = (1101)$.

(d) In the following we present some more examples of the inverse case, mapping a word of $\mathcal{W}_{6,3}$ into the corresponding 4-digit fragment, reported on the right. Let the index 2 denote the binary representation of the related natural number:

i. $I(000111) = \binom{2}{3} + \binom{1}{2} + \binom{0}{1} = 0 = (0)_2 \quad \longleftrightarrow (0000)$ in $\mathbb{F}_2^4$;

ii. $I(011001) = \binom{4}{3} + \binom{3}{2} + \binom{0}{1} = 7 = (111)_2 \quad \longleftrightarrow (0111)$ in $\mathbb{F}_2^4$;

iii. $I(110010) = \binom{5}{3} + \binom{4}{2} + \binom{1}{1} = 17 = (10001)_2$.

As $17 \geqslant 2^4$, it is equal to a 5-digit number in binary representation, hence it cannot be used for data encoding if only the elements of $\mathcal{U}_{6,3}$ are considered. In fact, $\mathcal{U}_{6,3} = \{(000111), (001011), \ldots, (101100)\}$ and $(110010) \in \mathcal{W}_{6,3} \setminus \mathcal{U}_{6,3}$.

2.3 Finding the systematic form of a GRS parity-check matrix

In the following we present an adaptation of the procedure proposed in [15, pp. 7–8] to obtain the systematic form of a parity-check matrix of a GRS code in general form. Let H be such a matrix, as given in Lemma 1.3.11. Our goal is to find

$$\mathsf{H}_s = \left(\mathsf{C} \mid \mathsf{I}_r\right) = \left(\begin{array}{cccc|c} C_{1,1} & C_{1,2} & \cdots & C_{1,k} & \\ C_{2,1} & C_{2,2} & \cdots & C_{2,k} & \\ \vdots & \vdots & \ddots & \vdots & \mathsf{I}_r \\ C_{r,1} & C_{r,2} & \cdots & C_{r,k} & \end{array}\right),$$

which can be obtained by anyone in $\mathcal{O}(n^3)$ operations by finding the inverse matrix of the r rightmost columns of H, denoted by $\mathsf{H}_{\to r}$. First, we need the following *Lagrange interpolation polynomials*:

$$L_i(x) := \sum_{s=1}^{r} L_{i,s} x^{s-1} = \prod_{\substack{1 \leqslant s \leqslant r, \\ s \neq i}} \frac{x - a_{k+s}}{a_{k+i} - a_{k+s}} \quad \forall i = 1, \ldots, r. \tag{2.4}$$

The variable can assume the values $a_j = a_{k+s}$, where s is chosen among 1 and r; that is, the index j represents one of the columns $k+1$ to n of H. Take now an a_j, where j is, say, equal to $k+\sigma$ for some $\sigma \in \{1, \ldots, r\}$ and keep it fixed. If $j = k+i$, or equivalently $\sigma = i$, all factors of the product are equal to 1; in the other case, the product will contain a 0. Summarizing:

$$L_i(a_j) = \sum_{s=1}^{r} L_{i,s} a_{k+\sigma}^{s-1} = \delta_{k+i,j} = \delta_{i,\sigma}, \tag{2.5}$$

where δ is the Kronecker symbol.

Lemma 2.3.1. *The inverse matrix* $\mathsf{L} := (\ell_{i,j})_{i,j\in\{1,\dots,r\}^2}$ *of* $\mathsf{H}_{\to r}$ *is given by* $\ell_{i,j} = \frac{L_{i,j}}{z_{k+i}}$, *where the* z_j *are the same ones of* Lemma 1.3.11.

Proof. Recall that $k+r=n$. From Lemma 1.3.11 and (2.4), as well as by the fact that $z_j \neq 0$ $\forall j$, it follows that:

$$\mathsf{L}\cdot\mathsf{H}_{\to r} = \begin{pmatrix} \frac{L_{1,1}}{z_{k+1}} & \frac{L_{1,2}}{z_{k+1}} & \cdots & \frac{L_{1,r}}{z_{k+1}} \\ \frac{L_{2,1}}{z_{k+2}} & \frac{L_{2,2}}{z_{k+2}} & \cdots & \frac{L_{2,r}}{z_{k+2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{L_{r,1}}{z_n} & \frac{L_{r,2}}{z_n} & \cdots & \frac{L_{r,r}}{z_n} \end{pmatrix} \begin{pmatrix} z_{k+1} & z_{k+2} & \cdots & z_n \\ z_{k+1}a_{k+1} & z_{k+2}a_{k+2} & \cdots & z_n a_n \\ \vdots & \vdots & \ddots & \vdots \\ z_{k+1}a_{k+1}^{r-1} & z_{k+2}a_{k+2}^{r-1} & \cdots & z_n a_n^{r-1} \end{pmatrix} =$$

$$= \left(\sum_{s=1}^{r} \frac{z_j}{z_{k+i}} L_{i,s} a_j^{s-1}\right)_{\substack{i=1,\dots,r,\\ j=k+1,\dots,n}} = \left(\frac{z_{k+\sigma}}{z_{k+i}} \sum_{s=1}^{r} L_{i,s} a_{k+\sigma}^{s-1}\right)_{\substack{(i,\sigma)\in\\ \{1,\dots,r\}^2}} \overset{(2.5)}{=} \left(\frac{z_{k+\sigma}}{z_{k+i}}\delta_{i,\sigma}\right)_{(i,\sigma)} = \left(\delta_{i,\sigma}\right)_{(i,\sigma)} = \mathsf{I}_r.$$

□

The systematic matrix H_s is obtained by multiplying the whole matrix H by L:

Corollary 2.3.2. *The matrix* $\mathsf{C} = \left(C_{i,j}\right)_{\substack{i=1,\dots,r,\\ j=1,\dots,k}}$ *is composed of the entries*

$$C_{i,j} = \frac{z_j}{z_{k+i}} \prod_{\substack{1\leqslant s\leqslant r,\\ s\neq i}} \frac{a_j - a_{k+s}}{a_{k+i} - a_{k+s}}.$$

Proof. By reusing the computations performed in the proof of Lemma 2.3.1, we get

$$\mathsf{LH} = \left(C_{i,j}\right)_{\substack{i=1,\dots,r,\\ j=1,\dots,n}} = \left(\frac{z_j}{z_{k+i}} \sum_{s=1}^{r} L_{i,s} a_j^{s-1}\right)_{(i,j)} = \left(\frac{z_j}{z_{k+i}} \prod_{s=1}^{r} \frac{a_j - a_{k+s}}{a_{k+i} - a_{k+s}}\right)_{(i,j)},$$

of which we only have to consider the first k columns. □

Chapter 3

Description of some cryptosystems

Codes are known for being employed in detection and correction of errors happening during information exchange between two users. McEliece [25] in 1978 and Niederreiter [30] in 1986 proposed each a public-key cryptosystem which exploits the capability of correcting errors for cryptographic applications, the former using as a basis a generator, the latter a parity-check matrix of a linear code. A third system, discovered by Sidel'nikov in 1994 [40], can be considered as a generalization of the former two.

3.1 McEliece cryptosystem

Among all existing classes of linear codes, to construct his public-key cryptosystem Robert McEliece chose to take Goppa codes. The advantage of this family of linear codes was the existence of a fast decoding algorithm already at his time [33]; though, later many authors have required only general $[n, k, d]$ linear codes, with the condition that there exists an algorithm able to decode the received messages within a computationally reasonable time (see e.g. [31, 39, 46]).

We describe here the original version of the cryptosystem, employing Goppa codes, as proposed by the author, listing the three steps of the cryptosystem (generation of public and private key, encryption and decryption) separately.

In Alg. 2a to 2c let m and t be natural numbers. Let $n = 2^m$, choose $t < \frac{n}{m}$ and pick an irreducible polynomial, say G, of degree t having coefficients in the field $\mathbb{F}_n$. In this way we can define a Goppa code Γ of length n with Goppa polynomial G, having dimension $k \geqslant n - mt$ and, because we are working over a binary field, being able to correct at least t errors happening during transmission on a noisy channel (for this, see Thm. 1.4.11). In his paper, McEliece proposed to take $m = 10$ and $t = 50$, that is a binary $[1024, 524, 101]$ Goppa code.

Even though these values are considered as not sufficient for the computational capabilities of today's most powerful calculators (see [46] and equation (4.1)), McEliece, together with the similar Niederreiter cryptosystem presented in the next section, is up to now one of the few ciphers which has resisted to almost every cryptanalytic attempt [2, 3, 49], mainly thanks to its structure based on Goppa codes [28, p. 348], with the only exception represented by the result reached by Bernstein, Lange and Peters [6], who managed to break the system with the original parameters, given above. Consequently, in spite of the dimension of the keys, it is of great interest in the theory of quantum computers: in fact, it seems that already relatively small parameters are enough to prevent such attacks [32, p. 115].

Algorithm 2a – McEliece: *Keys generation* [19, 25, 34]

- **Choose parameters** k, m and t as above.
- **Pick** an irreducible polynomial $G \in \mathbb{F}_{2^m}[x]$ of degree t and generate a Goppa code Γ.
- **Produce:**
  - a $k \times n$ generator matrix G of Γ;
  - a $k \times k$ random "scrambling" matrix S having $\det(\mathsf{S}) \neq 0$;
  - an $n \times n$ random permutation matrix P.
- **Compute** the $k \times n$ matrix $\mathsf{G}' := \mathsf{SGP}$.
- **Public key:** (G', t). For encryption, Alice needs n and k too, but she can directly obtain them from G'.
- **Private key:** $(\mathsf{S}, \mathsf{G}, \mathsf{P}, \Gamma)$.

Algorithm 2b – McEliece: *Encryption*

Recall that all vectors are represented as row vectors.
To encode a message, Alice cuts it into blocks of length k. Let $\mathbf{x} \in \mathbb{F}_2^k$ be one of them. Now she takes a random error vector $\mathbf{e} \in \mathbb{F}_2^n$ with $\mathrm{wt}(\mathbf{e}) = t$ and computes the ciphertext $\mathbf{y}$:

$$\mathbf{y} = \mathbf{x}\mathsf{G}' + \mathbf{e}.$$

Algorithm 2c – McEliece: *Decryption*

When Bob gets a message, he computes:

$$\tilde{\mathbf{y}} := \mathbf{y}\mathsf{P}^{-1} = \mathbf{x}\mathsf{SG} + \mathbf{e}\mathsf{P}^{-1}.$$

Since $\mathbf{x} \in \mathbb{F}_2^k$, $\mathbf{x}\mathsf{SG}$ is a codeword of Γ. We also know that

$$d_{\mathrm{H}}(\tilde{\mathbf{y}}, \mathbf{x}\mathsf{SG}) = \mathrm{wt}(\mathbf{e}\mathsf{P}^{-1}) = t,$$

where d_{H} represents the Hamming distance. Notice that both G and SG define the same code, because the rows of the latter are given by linear combinations of the rows of the former, that is they generate the same space. Hence, Bob can apply the decoding algorithm to $\tilde{\mathbf{y}}$ in order to retrieve $\mathbf{x}\mathsf{S}$ and thus $\mathbf{x}$.

Remark 3.1.1.

i. Instead of the matrices S, G and P together with the code Γ, as an alternative private key Bob could simply store the triple $(\mathsf{S}^{-1}, \mathsf{P}^{-1}, \mathcal{A}_t(\Gamma))$, where $\mathcal{A}_t(\Gamma)$ represents an efficient algorithm for correcting t errors in the code Γ. In this case, the generator G and other information about the code could be destroyed after the computation of G', slightly improving the security and reducing the size of the private key;

ii. A higher value for t does not necessarily imply greater security: in his paper, McEliece suggested to choose $t = 50$, but it has been shown that the highest work factor, that is the expected amount of operations for an eavesdropper, when $m = 10$ is reached at $t = 37$ (see [1] and Sect. 4.3).

3.2 Niederreiter cryptosystem

Few years after McEliece, another public-key cryptosystem was discovered, this time at the hand of Niederreiter. Differently from McEliece, he does not resort to a specific class of codes, but he simply assumes the use of some linear block code respecting two constraints: the code should have a large error-correcting capability and its decoding phase has to be performed in a reasonable amount of time; for this, he suggested the employment of e.g. alternant, Goppa or RS codes. However, we will see that not every class of codes is suitable for being employed for cryptographic purposes.

It is interesting that the Austrian mathematician introduces his own result underlining its analogies with knapsack-type cryptosystems, starting from the one published by Chor and Rivest, without even citing the McEliece cryptosystem, in spite of their undeniable similarity, being both based on error correcting codes. Indeed, each system exploits one of the two fundamental objects underlying the structure of a linear code: the generator and the parity-check matrix, in both cases multiplied by a scrambling and a permutation matrix. While McEliece transforms the message into a codeword which must be freed from an intentionally added error vector, Niederreiter wants exactly this vector to be the mean of transmission of Alice's information. For this, choose natural numbers n and k to construct an $[n, k]$ linear code $\mathcal{C}$, verify it can correct (say) t errors and make sure it is provided with an efficient t-error correcting algorithm $\mathcal{A}_t(\mathcal{C})$. This code will have parity-check matrix H, whose size is $r \times n$, $r = n - k$ representing the redundancy.

Algorithm 3a – Niederreiter: *Keys generation* [30]

- **Choose parameters** n and k (also t if already known).
- **Pick** an $[n, k]$ code $\mathcal{C}$ able of correcting t errors.
- **Produce:**
  - an $r \times n$ parity-check matrix H of $\mathcal{C}$;
  - an $r \times r$ random "scrambling" matrix M having $\det(\mathsf{M}) \neq 0$;
  - an $n \times n$ random permutation matrix P, like in McEliece.
- **Compute** the $r \times n$ matrix $\mathsf{H}' := \mathsf{MHP}$.
- ◇ **Public key:** (H', t). Analogously to McEliece cryptosystem, Alice can recover r from H'.
- ◆ **Private key:** $(\mathsf{M}, \mathsf{H}, \mathsf{P}, \mathcal{C})$.

Algorithm 3b – Niederreiter: *Encryption*

Messages are encoded as strings of length n. Let $\mathbf{x} \in \mathbb{F}_2^n$ be a plaintext which Alice wants to send to Bob. For enabling him to decrypt the received message, Alice must make sure that her plaintext's weight is $\leqslant t$, since it can be seen as an error vector related to the all-zero vector. She now encrypts $\mathbf{x}$ as:

$$\mathbf{y}^\top = \mathsf{H}'\mathbf{x}^\top.$$

Clearly here column vector representations are required.

Remark 3.2.1. The same reasoning made in Rk. 3.1.1 can be applied to the Niederreiter cryptosystem, substituting the triple $(\mathsf{S}^{-1}, \mathsf{P}^{-1}, \mathcal{A}_t(\Gamma))$ with $(\mathsf{M}^{-1}, \mathsf{P}^{-1}, \mathcal{A}_t(\mathcal{C}))$.

Algorithm 3c – Niederreiter: *Decryption*

Bob receives a column vector $\mathbf{y}^\top \in \mathbb{F}_2^r$. He starts the decoding procedure by computing

$$\tilde{\mathbf{y}}^\top := \mathsf{M}^{-1}\mathbf{y}^\top = \mathsf{HP}\mathbf{x}^\top = \mathsf{H}(\mathbf{x}\mathsf{P}^\top)^\top.$$

Once more we deal with a vector which is only a permutation of another one. Thus, Bob can exploit the property that

$$\mathrm{wt}(\mathbf{x}\mathsf{P}^\top) = \mathrm{wt}(\mathbf{x}) \leqslant t$$

together with his secret decoding algorithm to retrieve $\mathbf{x}\mathsf{P}^\top$ and, by inverting $\mathsf{P}^\top$, the original plaintext $\mathbf{x}$.

In Sect. 3.3 we will introduce the Sidel'nikov cryptosystem, which is closely related to McEliece and Niederreiter. We don't tell anything in advance about this cipher in order to avoid creating unnecessary duplications, but for greater clarity in Alg. 4c we insert here a variant of Niederreiter which makes extensive use of the characteristics of McEliece cryptosystem.

Assume a matrix G' is constructed like in Alg. 2a and let $\mathcal{C}$ be the $[n, k, d]$ code generated by G'. A message $\mathbf{x} \in \mathbb{F}_2^n$ of weight $\leqslant t$ is encrypted as $\mathbf{y}^\top = \mathsf{H}\mathbf{x}^\top$, where H is a parity-check matrix for $\mathcal{C}$.

Algorithm 3d – Niederreiter: *Alternative decryption* [40]

Bob receives the vector $\mathbf{y}^\top \in \mathbb{F}_2^r$, which is the syndrome of $\mathbf{x}$. He now looks for some $\mathbf{b} \in \mathbb{F}_2^n$ having the same syndrome, i.e. an element belonging to the coset $\mathbf{x} + \mathcal{C}$. The vector $\mathbf{b}$ can be seen as a ciphertext encrypted with the McEliece cryptosystem:

$$\mathbf{b} = \mathbf{a}\mathsf{G}' + \mathbf{x},$$

where $\mathbf{a} \in \mathbb{F}_2^k$ is the corresponding plaintext and G' a generator matrix for the code in use. Recall that $\mathrm{wt}(\mathbf{x}) \leqslant t$, so $\mathbf{x}$ is a valid error vector. Applying the decoding algorithm to $\mathbf{b}\mathsf{P}^{-1}$, we find $\mathbf{x}\mathsf{P}^{-1}$ and thus $\mathbf{x}$.

Alg. 3d follows the same reasoning contained in [22] to prove the equivalence of the security level of McEliece and Niederreiter.

3.3 Sidel'nikov cryptosystem

What we are going to present in this section does not actually involve anything new from a structural point of view: the Sidel'nikov cryptosystem can in fact be simply seen as a generalization of McEliece and Niederreiter ciphers. The author, in his paper [40] dated 1994, actually proposed a Niederreiter-type cryptosystem constructed over a McEliece-like basis. More precisely, the generalization lies in the construction of the public key, which is built starting from a certain amount u of invertible matrices, which for coherence we will call S_1 to S_u, instead of just one; the resulting blocks are then chained and inserted into a single matrix. To get the Niederreiter version of Sidel'nikov cryptosystem, the user has to find a parity-check matrix for the code resulting from the reasoning presented in Alg. 4a. The author chose to

insert this extra step because Niederreiter guarantees a higher rate and an increased security (for this, see also Ch. 4). Sidel'nikov also recommends the use of RM codes. Regarding the correction capability, we assume as usual that the employed code can correct t errors; we also define T as the number of errors which the output code of the following algorithm is able to correct.

Algorithm 4a – Sidel'nikov: *Keys generation* [40]

- **Choose parameters** ρ and m as in Def. 1.5.11; $u \in \mathbb{N}$.
- **Generate** the $\mathsf{RM}(\rho, m)$ code $\mathcal{R}$.
- **Produce:**
  - a $k \times n$ generator matrix G for the code $\mathcal{R}$;
  - u random pairwise distinct $k \times k$ matrices $\mathsf{S}_1, \dots, \mathsf{S}_u$ having $\det(\mathsf{S}_i) = 0 \ \forall i = 1, \dots, u$;
  - a $un \times un$ random permutation matrix P.
- **Compute:**
  - the $k \times n$ matrices $\mathsf{E}_i := \mathsf{S}_i \mathsf{G} \ \forall i$, sometimes called *blocks*;
  - the $k \times un$ matrix $\mathsf{E} := \left(\mathsf{E}_1 \,\middle|\, \cdots \,\middle|\, \mathsf{E}_u\right) \cdot \mathsf{P}$.
- ◇ **Public key:** (E, T). As in the previous cases, its size can be easily retrieved.
- ◆ **Private key:** $(\mathsf{S}_i, \mathsf{P})$.

Until now, the algorithm looks like a generalization of McEliece. The Niederreiter-type section of the cipher can be found in the encryption part, which behaves just like in Alg. 3b, so we just resume it here briefly. Let D be a parity-check matrix of the code generated by E. A plaintext is an element $\mathbf{x} \in \mathbb{F}_2^{un}$ with $\mathrm{wt}(\mathbf{x}) \leqslant T$. The corresponding ciphertext is obtained through $\mathbf{y}^\top = \mathsf{D}\mathbf{x}^\top \in \mathbb{F}_2^{\bar{r}}$, where $\bar{r} := un - k$. The reason because of which Sidel'nikov chose not to insert the computation of D directly in the part a of the scheme consists in the sizes of the parity-check matrix itself: as the author suggests the use of low-rate RM codes, i.e. with $\rho \ll m$, E is way smaller than D, so the transmission speed of the public key is enhanced. The decryption part will be presented after a short discussion about the weight of the plaintext.

In his paper, Sidel'nikov suggests to encode plaintexts with the highest possible weight and mentions as an admissible value $T = ut + u - 1$. If we divide this equation by u, we get that $\frac{T}{u} < t + 1$, implying that, if the plaintext is broken in strings of length n, there must be at least one with weight $\leqslant t$. To be able to encode words not exceeding the upper bound T, we can once more appeal to Guillot's algorithm. If we forget for now what we just reported about the possibility of correcting more errors through particular algorithms, we get that the length ℓ of the plaintext block which can be encoded at a time is at most

$$\bar{\ell} = \left\lfloor \log_2 \binom{un}{T} \right\rfloor = \left\lfloor \log_2 \binom{u2^m}{ut+u-1} \right\rfloor .$$

Differently from McEliece and Niederreiter, note that in Alg. 4a neither t nor T have been declared as an input of the algorithm: this is due to the fact that the error correcting capability of a RM code is a consequence of the parameters chosen to build it, in this instance ρ and m, and not a requirement, as it is the case for example of Goppa codes. From Thm. 1.5.12 it follows that a RM code should be able to correct $2^{m-\rho-1} - 1$ errors,[4] but Sidel'nikov claims that experimentally the codes $\mathsf{RM}(3, 10)$ and $\mathsf{RM}(3, 11)$ can "almost always" correct 200 and 420 errors respectively, many more than the 63 and 127 promised by the well-known formula

[4] This value is not exact only for the case $\rho = m$, which is more than enough for applications in practice.

Algorithm 4c – Sidel'nikovDecryption [40]

Bob receives from Alice the ciphertext $\mathbf{y}^\top = \mathsf{D}\mathbf{x}^\top \in \mathbb{F}_2^{\bar{r}}$. First of all, he has to find some $\mathbf{b} \in \mathbb{F}_2^{un}$ as in Alg. 3d. In order to apply it, he has to cut the vector $\tilde{\mathbf{b}} := \mathbf{b}\mathsf{P}^{-1}$ into pieces of length n, say $\tilde{\mathbf{b}} = (\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_u)$. Using the same notation for $\mathbf{x}$, we define $\tilde{\mathbf{x}} := \mathbf{x}\mathsf{P}^{-1} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_u)$, where each $\tilde{\mathbf{x}}_i \in \mathbb{F}_2^n$. Assume T was chosen to be $ut + u - 1$; then

$$\exists i \colon \mathrm{wt}(\tilde{\mathbf{x}}_i) < t + 1.$$

Let j be this index. Following Alg. 3d, this implies that $\tilde{\mathbf{b}}_j = \mathbf{a}\mathsf{E} + \tilde{\mathbf{x}}_j$ for some $\mathbf{a} \in \mathbb{F}_2^k$. Application of a decoding algorithm for McEliece allows to recover both $\mathbf{a}$ and $\tilde{\mathbf{x}}_j$. Thus, all other $\tilde{\mathbf{x}}_i$ are simply retrieved through

$$\tilde{\mathbf{x}}_i = \mathbf{a}\mathsf{E} + \tilde{\mathbf{b}}_i\,.$$

$t = \left\lfloor \frac{d-1}{2} \right\rfloor$. Also Minder and Shokrollahi [28, p. 350] support this statement, pointing out that there exist algorithms able to decode with high probability "many more errors" than the value reported above.

Another evident distinction between this and the former two cryptosystems lies in the private key. Note that Sidel'nikov does not include the generator matrix in it: this is due to the fact that, analogously to the weight of the plaintext, also the dimension of RM codes depends on ρ and m. In Thm. 1.5.12 we have seen that this is equal to $\sum_{i=0}^{\rho} \binom{m}{i}$ and by Alg. 4a, k is also the number of rows of E. Hence, given a public matrix for a Sidel'nikov cryptosystem, recovering the generator G is straightforward, since for each dimension there is almost always just one RM code. In Tab. 4 we list the values of k for m up to 14, which should be enough for a practical use.

ρ \ m	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	m / ρ
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1		2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
2			4	7	11	16	22	29	37	46	56	67	79	92	106	2
3				8	15	26	42	64	93	130	$\boxed{176}$	$\boxed{232}$	299	378	470	3
4					16	31	57	99	163	256	386	562	794	1093	1471	4
5						32	63	120	219	382	638	1024	1586	2380	3473	5
6							64	127	247	466	848	1486	2510	4096	6476	6
7								128	255	502	968	1816	3302	5812	9908	7
8									256	511	1013	1981	3797	7099	12911	8
9										512	1023	2036	4017	7814	14913	9
10											1024	2047	4083	8100	15914	10
11												2048	4095	8178	16278	11
12													4096	8191	16369	12
13														8192	16383	13
14															16384	14

Table 4 – Dimensions of RM codes in function of ρ and m.

The boxed numbers correspond to $\mathsf{RM}(3, 10)$ and $\mathsf{RM}(3, 11)$, that is the codes chosen by Sidel'nikov as an example. In the table we have also highlighted some cells in grey: these

are the cases where a certain dimension can be obtained with different choices of ρ and m, that is with more than one code. The ones with light background are the cases for which small parameter values are involved and consequently the dimension is very low. To this group belong few elements and the codes generated by $\rho \leqslant 2$ or $m \leqslant 5$ are not useful in practice. The darker gray has the goal of emphasizing a pattern repeating with regularity, with couples of cells containing the same number. As before, the amount of such cells is very low; furthermore, one of the two cells building those couples is always obtained by choosing $\rho = m$ and hence is not interesting for a concrete application. Thus, one can uniquely identify the generator matrix G just by counting the number of rows of E.

Moreover, in spite of Sidel'nikov's conclusion that his work represents an improvement of the two previously published code-based public-key cryptosystems, Minder and Shokrollahi [28] showed that this variant actually does not increase security if compared to the original work by McEliece, which corresponds to Sidel'nikov by assuming $u = 1$ and using RM codes instead of Goppa codes. The two mathematicians also found a way to cryptanalyse the system by exploiting minimum weight codewords.

Chapter 4

Costs and security

In the following we group all concerning the security and the efficiency of the cryptosystems encountered in CH. 3. At first, we will discuss their resistance against brute force attacks, considering in particular the number of possible different codes given some parameters (see also the reasoning performed in SECT. 2.1). After that we present the message-resend attack, to which we will return in the third section, dealing with the attractiveness of those cryptosystems from the point of view of the time (i.e. the amount of bit operations) needed for executing the different phases of which they are composed. The last part is dedicated to the well-known Sidel'nikov-Šestakov attack.

4.1 The security assumptions

The main part of cryptosystem consists in its security, that is the difficulty for an eavesdropper to retrieve the plaintext, either directly or after having discovered the secret key. The security of a McEliece-typecryptosystem can be reduced to two assumptions [3, 12, 46]:

- The *decoding attacks*: an eavesdropper intercepts a ciphertext and tries to recover the corresponding plaintext. This step should be intractable for everyone but the code designer, who is the only one in possession of the trapdoor (the private key). Eve must repeat the process for every new intercepted ciphertext. This corresponds to solving the syndrome decoding problem:

  Given an $r \times n$ parity-check matrix H over $\mathbb{F}_2$, a binary vector $\mathbf{y} \in \mathbb{F}_2^r$ and a natural number t, can one find a codeword $\mathbf{x} \in \mathbb{F}_2^n$ with $\mathrm{wt}(\mathbf{x}) \leqslant t$ such that $\mathsf{H}\mathbf{x}^\top = \mathbf{y}^\top$?

  This problem is known to be NP-complete [4, 19]. An example of a decoding attack, the *message-resend attack*, is reported in SECT. 4.3.

- The *structural attacks*, aiming at retrieving the private key (or an equivalent one) from the public information. In other words, in the case of McEliece, the fact Eve must establish is:

  Does a given $r \times n$ binary matrix H define a Goppa code?

  An example of a successful attack of this type against cryptosystems based on GRS codes was discovered by Sidel'nikov and Šestakov (see SECT. 4.5).

4.2 No hope for brute force

In order to reduce the risk of a successful attack, it is necessary to choose large values for the code parameters, so that it is not feasible for Eve to perform an exhaustive search among all possible choices for the keys.

In Sect. 2.1 we have seen that the number of irreducible polynomials of a given degree t increases with t and the probability of finding one by picking a random monic polynomial lies around $\frac{1}{t}$; equivalently, the amount of irreducible Goppa codes able of correcting t errors grows proportionally with this value.[5] By taking advantage of the general decoding problem, a good security level for the McEliece cryptosystem should be achieved just by choosing big enough values for the parameters. Using the same notation as in Thm. 2.1.6, for $m = 10$ and $t = 50$ as proposed by McEliece, we see that to build a Goppa code we can pick one of the $N^I_{2^{10}}(50) > 10^{148}$ different Goppa polynomials. Even the most powerful computers would struggle: just to give an idea, if the supercomputer *Monte Rosa*, owned by the Swiss National Supercomputing Centre (CSCS) and able to perform up to 141 teraflops ($= 141 \cdot 10^{12}$ mathematical operations per second), were faced with this situation, it would take over 10^{126} years to go trough all possible such polynomials.

In addition, the enormous amount of possibilities for the "scrambling" matrices P and S would make any brute force attack definitely hopeless. A binary permutation matrix is composed of all possible vectors with weight 1 of the given length. For the Sidel'nikov cryptosystem, this returns a total of $(un)!$ possible choices for P; in the case of McEliece and Niederreiter, just set $u = 1$. We now consider the matrix S in McEliece, but the same reasoning is clearly valid for M in Niederreiter respectively the S_i in Sidel'nikov too. Looking for the amount of possible combinations for S corresponds to calculating the cardinality of $\mathsf{GL}_k(\mathbb{F}_2)$, the group of all invertible $k \times k$ matrices over $\mathbb{F}_2$. An element of this group comprises k linearly independent vectors, hence in the ith row we have to subtract from the total of 2^k possible vectors the 2^{i-1} combinations in rows 1 to $i-1$ with which it would build a family of linearly dependent vectors. So, for large k we get:

$$\#\mathsf{GL}_k(\mathbb{F}_2) = \prod_{i=0}^{k-1} \left(2^k - 2^i\right) = (2^k)^k (1-2^{-k})(1-2^{1-k})\cdots(1-2^{-1}) = 2^{k^2} \underbrace{\prod_{i=1}^{k} \left(1 - 2^i\right)}_{\cong 0{,}29} > 2^{k^2-2}.$$

By putting in the suggested values of $m = 10$ and $k = 524$ (the lowest possible dimension with such parameters), it turns out that it is as well impossible to retrieve the other two elements building the private key just by guessing, since there exist $1024! > 10^{2639}$ different matrices P and nearly $2^{524^2} > 10^{82000}$ choices for S. In fact, the set of effectively serviceable matrices contains way less elements than $\#\mathsf{GL}_k(\mathbb{F}_2)$, as S is supposed to be chosen "dense" [25], but this will not surely make the eavesdropper's task any easier. For the same reasons, Eve will not be able to recover the message $\mathbf{x}$ directly from the received vector $\mathbf{y}$, as an exhaustive search in a standard array of such size is infeasible.

[5] For their number without taking into account the equivalent codes, the reader is referred to [13, Thm. 6.1]: if the Goppa code is based on some polynomial $G \in \mathbb{F}_{q^m}$ of degree ρ, where m, ρ are distinct primes, it is possible to give an upper bound of roughly $\frac{q^{m(r-2)}}{mr} + \frac{q^{r-2}}{r}$.

4.3 Message-resend attack

What we are going to present now is an example of a case where the Niederreiter cryptosystem proves to be stronger than McEliece. The *message-resend attack* (MRA) was discovered by Berson [7] and belongs to the class of decoding attacks, since its goal is to decode a single ciphertext and not the structure of the private code. It exploits the fact that the McEliece cryptosystem is non-deterministic, since the ciphertext partly depends from the error vector, which can be different at each encoding. The efficiency of an attack is measured by the work factor $\mathsf{WF}(\cdot)$, that is a function depending on the code parameters which returns the expected number of operations needed to accomplish the task.

Assume we are working with a McEliece scheme constructed over some $[n, k, d]$ code. Berson noticed that if some message is encrypted twice using two distinct error vectors $\mathbf{e}_1$ and $\mathbf{e}_2$, then anyone can immediately recognize that the two ciphertexts $\mathbf{y}_1$ and $\mathbf{y}_2$ correspond to the same plaintext, since the Hamming distance between the two $\mathbf{y}_i$ (that is, between the two error vectors) is at most equal to the code distance d; on the contrary, if two distinct messages are sent, the distance between them is on average about $\frac{n}{2}$. To implement the MRA, Eve starts from the fact that there exist $\binom{n}{t}$ different error vectors $\mathbf{e}$. Because G' and t are public, Eve knows the values of n and k, as well as that $\mathrm{wt}(\mathbf{e}) \leqslant t$. If she is able to guess where k of the 0's of $\mathbf{e}$ are, she builds a $k \times k$ submatrix G'' with the columns of G' corresponding to the k coordinates and defines $\mathbf{y}'' := \mathbf{x}\mathsf{G}''$. Then, $\mathbf{x}$ is retrieved by inverting G'', assuming it is nonsingular.

There are $\binom{n-t}{k}$ possibilities for picking k 0's from the $n-t$ available, hence the work factor is given by multiplying the computational complexity of matrix inversion by the expected number of attempts for actually finding a set of components of $\mathbf{e}$ all equal to 0. For an explicit computation of the work factor we set the cost for inverting a $k \times k$ matrix to $\mathcal{O}(k^{2,3727})$ bit operations, which corresponds to the result recently obtained in [50]; the other parameters $n = 2^m$ and $k = n - mt$ are determined by the value $m = 10$ as proposed by McEliece. Then, we have an expected running time of

$$\mathsf{WF}(m,t) = \mathsf{WF}(10,t) = k^{2,3727}\frac{\binom{n}{k}}{\binom{n-t}{k}} = (1024-10t)^{2,3727}\frac{\binom{1024}{1024-10t}}{\binom{1024-t}{1024-10t}} \tag{4.1}$$

operations, which reaches a maximum of about $\mathsf{WF}(10,37) = 2^{78,268}$. This proves Rk. 3.1.1(ii) and furthermore a lower t increases the ratio $\frac{k}{n} =: R$, called *transmission rate* (see Sect. 4.4). Though, in [46] Sendrier suggests that this number is still too low to guarantee security in practical use, but it should be enough to choose better code parameters to avoid the MRA. In spite of this, Berson itself shows that his attack can be much more powerful if, instead of the exhaustive search, a different method is applied, allowing a success after just few attempts (twelve if using the above parameter values). The reasoning returning equality (4.1) actually follows the procedure called *information-set decoding*, so that its complexity corresponds to the work factor function for the same parameters. Good approximations of the needed amount of bit operations for the information-set decoding can be found in [5, p. 3] and [46]. Note that $1 - R = \frac{r}{n}$; if $P(n)$ is some polynomial of degree at most 3, these approximations are respectively

$$\mathsf{WF}_1(n,R,t) \cong P(n)2^{t\log_2\frac{R}{1-R}} = P(n)\left(\tfrac{1}{1-R}\right)^t = P(n)\left(\tfrac{n}{r}\right)^t$$

and

$$\mathsf{WF}_2(R,n) \cong \frac{1}{0{,}29}\left(\frac{1}{(1-R)^{1-R}}\right)^{\frac{n}{\ln n}} \cong 3{,}45\left(\left(\frac{n}{r}\right)^{\frac{r}{n}}\right)^{\frac{n}{\log_2 n}} = 3{,}45\left(\frac{n}{r}\right)^{\frac{r}{n}\frac{n}{m}} = 3{,}45\left(\frac{n}{r}\right)^{t}.$$

Experimentally, with our computations using (4.1) we get values for $\mathsf{WF}(m,t)$ very close to $\mathsf{WF}_1(n,R,t)$ if $P(n)$ is a constant in a small interval around 1,3.

Again in [7], Berson also pointed out that a message can be attacked even if it is sent just once, potentially exposing to risk any message encrypted with the McEliece cryptosystem. For this second method, called *related-message attack*, the eavesdropper is supposed to know a relation between two plaintexts $\mathbf{x}_1$ and $\mathbf{x}_2$ such as their sum. If she is not able to obtain a second message, she can try to intercept just one (say $\mathbf{y}_1$, corresponding to the plaintext $\mathbf{x}_1$) and encrypt some plaintext $\mathbf{x}_2$ in her turn. Assume Eve has two distinct ciphertexts $\mathbf{y}_j = \mathbf{x}_j\mathsf{G}' + \mathbf{e}_j$ for $j = 1, 2$. Then she can compute

$$\mathbf{e}_1 + \mathbf{e}_2 = \mathbf{y}_1 + \mathbf{y}_2 + \big((\mathbf{x}_1 + \mathbf{x}_2)\mathsf{G}'\big)$$

and go on as in the previous case.

4.4 Key sizes and operation costs

A good cryptosystem, to be employable in practice, has to guarantee a high security level against eavesdropping, but at the same time the sender and the recipient should be able to quickly perform the sundry steps allowing message exchange, combined with low needs of data storage. It is also important to carry out an analysis of the amount of bit operations required to run the parts a, b and c of the presented algorithms, in order to verify the applicability of a cryptosystem in real life. For the whole section, assume we are working with codes of length $n = 2^m$ over the binaries.

We start with the cost for generating and storing the public and private keys in the three cryptosystems.

Keys generation

For all the three cryptosystems, the first task which has to be dealt with is the generation of the public and of the private keys. The permutation matrices P and the scrambling matrices S, M and S_i can be very easily produced by a computer, since it is enough to add a weight-1 vector resp. a more general random binary vector at a time to a matrix, making sure that the new inserted row or column is linearly independent from all others. Controlling the independence can be done in polynomial time. However, this and some more processes requiring randomized generation algorithms such as producing a Goppa polynomial or an error vector $\mathbf{e}$ for the McEliece cryptosystem, could present some application difficulties (see [20] and [39, p. 147]).

Regarding the generation of a Goppa code, recall that the matrix G is obtained by multiplying two matrices X and Y (see the proof of Thm. 1.4.6 for their definition) whose entries are the n elements of the support L and the coefficients of the Goppa polynomial G respectively. In order to have a code of length $n = 2^m$, the code must be generated starting with an irreducible polynomial; as a consequence, we need an algorithm telling us if a randomly generated polynomial is actually irreducible. One such procedure has been proposed by Rabin [35, p.

275] and has complexity $\mathcal{O}(dM(d)\log d\log 2)$, where $M(d) = d\log d\log\log d$ is the expected running time for the multiplication of two polynomials of degree d.

The entries of X and Y are all elements of the field $\mathbb{F}_{2^m}$ for some m; hence the matrix multiplication XY has a cost close to $m \cdot n \cdot \deg G$ bit operations, since a multiplication in $\mathbb{F}_{2^m}$ can be seen as m binary operations. In the Niederreiter cryptosystem, the user needs a parity-check matrix H, which can be computed in polynomial time if a generator matrix for the code is available [27, p. 17].

In the case of the Sidel'nikov cryptosystem, we see that the generator matrix of a RM code is made of the vectors corresponding to all monomials of degree up to some integer ρ. If the elements of $\mathbb{F}_2^{2^m}$ obtained by vector multiplication (i.e. through the operation described in Def. 1.5.2) of the degree-1 monomials $x_1, \ldots, x_m$ have not been precomputed, then we see that to calculate $\psi(x_i) \star \psi(x_j)$, $n = 2^m$ bit multiplications are executed. More generally, to compute $\psi(x_{i_1}x_{i_2}\cdots x_{i_\rho})$, $(\rho-1)n$ operations are needed. Keeping into account the number of monomials, we get a total of $n \cdot \sum_{i=2}^{\rho}(i-1)\binom{m}{i}$, which can be strongly reduced by storing the intermediate steps.

Key sizes

The storing costs of the scrambling (S, M and S_i) and the permutation matrices (P) as parts of the private keys follow immediately from their sizes. On the contrary, the generator and the parity-check matrices can be modified in order to let them occupy less space if they are brought to systematic form, that is when $\mathsf{G} = \left(\mathsf{I}_k \mid \mathsf{A}\right)$ for some $k \times r$ matrix A, respectively $\mathsf{H} = \left(\mathsf{B} \mid \mathsf{I}_r\right)$ for some $r \times k$ matrix B; in this case, the user just has to save the submatrices A and B, but this extra step entails further $\mathcal{O}(n^3)$ bit operations if the code designer works following Sect. 2.3, respectively only $\mathcal{O}(n^{2,3727})$ if the procedure described in [50] is preferred.

To get the public keys from them, the user is faced with some matrix multiplications. Each cryptosystem contains a permutation matrix as a factor, which has no effect on the amount of bit operations needed and therefore can be discounted. Assume that the matrices G and H are in systematic form: in the multiplications SG (McEliece) resp. $\mathsf{S}_i\mathsf{G}$ (Sidel'nikov) about k^2r bit operations have to be accomplished, while in the case of MH (Niederreiter) their number is close to kr^2. The public key storage needs in bits are of course equal to the sizes of the resulting matrices G', H' and E, that is nk, nr and unk respectively. We can now suppose that G', H' and D are systematic too, where D is a parity-check matrix generating the same code as E: in this case, the space occupied by the public keys reduces to the sizes of the submatrices remaining by elimination of the identity matrices I_k, I_r and $\mathsf{I}_{\bar{r}}$, which for the first two of them is equal to kr and for Sidel'nikov it is $k\bar{r}$, with $\bar{r} = un - k$; simultaneously, multiplication by smaller matrices allows faster encryption running times.

About this point, an important fact differentiates between McEliece and the two other cryptosystems: whenever working with plaintexts over $\mathbb{F}_2$, Niederreiter and Sidel'nikov can be brought back to the knapsack problem, as the encrypting procedure consists in the addition of some columns of the public parity-check matrix; hence, the hypothesis that H' and D are systematic does not endanger the security of the system. On the contrary, in McEliece a generator matrix G' in that form cannot be published, as the plaintext $\mathbf{x}$ would appear in the first k positions of the ciphertext $\mathbf{y}$ with only some coordinates switched by the error vector $\mathbf{e}$. Wherever the use of systematic or of non-systematic matrices leads to different results, both cases will be treated.

To see how large these numbers can be, let us now take the McEliece cryptosystem with

the values $k = 524$ and $r = 500$ as proposed by the author himself. We see that the public key measures about 33 kB (or 67 kB if the non-systematic form is considered), which could look little thing if we think to the amount of space present on the hard disk of a common personal computer. However, its magnitude becomes more apparent if compared with the size of the keys in RSA cryptosystem. For the modulus n, that is for the number resulting from the multiplication of two primes p and q, *RSA Laboratories* currently recommend to assume a length up to 2048 bits, though observing that a key size of 768 bits already lies "beyond the reach of all known key breaking algorithms".[6] To these it has to be added the length of the public and private exponents e and d (i.e. two numbers smaller than φ such that $ed = 1 \mod \varphi$, where $\varphi = (p-1)(q-1)$). If e is chosen very small, we can assume that the storage needs for the two exponents are the same as the length of n. To make an example, supposing that n, e and d together occupy 4000 bits or 0,5 kB, the total storage capacity required at present for both public and private keys for the RSA system is more than 100 times smaller than what was requested by McEliece in 1978.

Guillot's algorithm costs

In the transformation of binary strings into encodable words and backwards, presented in Sect. 2.2, Alice and Bob have to face the calculation of a certain number of binomial coefficients. To be more precise, the sender has to compute exactly one and the receiver exactly t. We would like to know the expected running time of this process without precomputations. For this, we define the function $\mathrm{Bin}(n,t) := \binom{n}{t}$. The exploitation of the formula

$$\binom{n}{t} = \binom{n-1}{t-1} + \binom{n-1}{t} \tag{4.2}$$

permits a reduction of the complexity due to computing factorials. In fact, thanks to this equality it is possible to define any binomial coefficient as a recurrence relation, keeping on dividing it in two parts until $t = 0$ and setting as a seed value $\mathrm{Bin}(n,0) = 1 \ \forall n$. In this way, calculating a binomial coefficient reduces to adding a corresponding number of 1's, each of which is computed only with the help of the seed value, i.e. with complexity $\mathcal{O}(1)$. For t approaching 0 and n, the result will be reached more quickly: for small t, the coefficients need less iterations of (4.2) to arrive to the form $\binom{j}{0}$, $j \leqslant n$; in the other case, many instantiations of the algorithm will cease to run because they get to a coefficient $\binom{j}{i}$ where $i > j$. To make the algorithm a bit faster, instead of forcing the computer to run the algorithm until the coefficient $\binom{0}{0}$ once a symbol of the form $\binom{j}{j}$ appears, it can be also given the instruction that $\mathrm{Bin}(j,j) = 1 \ \forall j$. This reasoning implies that the needed amount of bit operations is higher when t is around $\frac{n}{2}$. From this and by knowing that not more than nt distinct couples (n,t) have to be computed in the intermediate steps, we can expect a running time of $\mathcal{O}(n^2)$ for every coefficient, which thus represents a lower bound (but also a good approximation) on the running time of the algorithm, estimated to be $\mathcal{O}(n^2 \log_2 n)$ [32, p. 98].

Encryption and decryption costs

Even in the encryption part the Niederreiter and the Sidel'nikov cryptosystems seem to have some advantages if compared with McEliece. We start by analysing the latter. To encode a plaintext represented by a string $\mathbf{x} \in \mathbb{F}_2^k$ we saw that Alice must multiply $\mathbf{x}$ by a matrix

[6] `http://www.rsa.com/rsalabs/node.asp?id=2218`, viewed on 12th July 2012, 17:45.

G' and then add an error vector $\mathbf{e}$. To give an estimate of the computational complexity, we can consider the fact that the vector $\mathbf{x}$ on average contains an equal number of 0's and 1's; this means that on average only half of the rows of G' are significant for the calculation of the ciphertext $\mathbf{y}$, letting the cost of the matrix-vector multiplications reduce to $\frac{k}{2}$ bit operations per column. Moreover, assuming $\mathsf{G}' = \left(\mathsf{I}_k \mid \mathsf{A}'\right)$ is in systematic form (in practice not safe for use), it is enough to multiply $\mathbf{x}$ by the r columns of the submatrix A', getting a total amount of binary operations of $\frac{kr}{2}$. Taking G' in a more general form, no column can be neglected, making their number approach $\frac{nk}{2}$. The summation of $\mathbf{x}\mathsf{G}'$ and the vector $\mathbf{e}$ corresponds to switching t coordinates of $\mathbf{x}\mathsf{G}'$, thus it has an irrelevant impact on the running time of Alg. 2b (though, recall the remark made in the keys generation part about possible problems with random number generators).

Let now $\mathsf{H}' = \left(\mathsf{B}' \mid \mathsf{I}_r\right)$ be the systematic parity-check matrix of some Niederreiter cryptosystem and let $\mathbf{x}^\top \in \mathbb{F}_2^n$ be a plaintext. Similarly to the previous situation, only the first k columns of H' are relevant for the computational complexity of Niederreiter encryption algorithm. In addition, one can realistically imagine that the t non-zero coordinates of $\mathbf{x}^\top$ are equally distributed, so that the columns of the submatrix B' that are actually added through multiplication by $\mathbf{x}^\top$ are on average $\frac{kt}{n}$. Multiplying this result by the number of rows of H' gives a total cost for encryption of $\mathcal{O}\left(\frac{krt}{n}\right)$ binary operations. For a general matrix H it suffices to substitute k through n. As $\frac{t}{n} < \frac{1}{2}$ for any t-error correcting linear code, Niederreiter has a lower complexity than McEliece in the encryption part.

Assuming Goppa codes were used, for decoding McEliece and Niederreiter one has to find the error locator polynomial, i.e. a polynomial telling where the errors in the received vector are, and solve it. For these operations, Sendrier estimates the complexity in λrn bit operations, where $\lambda \cong 3$ for the former[7] and at most $\mathcal{O}(\mu r^2)$ with $\mu \leqslant m$ for the latter.[8]

Now, similarly to Niederreiter, assume a parity-check matrix D of a Sidel'nikov cryptosystem is also in systematic form and $\mathbf{x}^\top \in \mathbb{F}_2^{un}$ is a vector of weight less or equal to T. Then the weight of the "upper part" of $\mathbf{x}^\top$, that is the one not being multiplied with the submatrix $\mathsf{I}_{\bar{r}}$, is about $\frac{kT}{un}$. Thus, the expected amount of operations for the multiplication $\mathsf{D}\mathbf{x}^\top$ is $\frac{kT\bar{r}}{un} \cong kut\left(1 - \frac{k}{un}\right)$, respectively $\mathcal{O}(\bar{r}T)$ if D is not systematic. A decoding algorithm for RM codes is given has been published by Sidel'nikov himself and Pershakov [40, Ref. 3]; from its complexity we can say that this cryptosystem is decodable in $\mathcal{O}(un^2 \log^{\rho-1} n)$ time.

Transmission rates

After having generated and stored the keys, the public key is ready to be used for message encoding. Another relevant point is to know the length of fragments in which a plaintext has to be cut in order to be accepted by the cryptosystem and consequently how many information exchanges are needed to complete the message forwarding. If Bob wants messages to be encrypted with McEliece cryptosystem, Alice can send k bits of her plaintext at a time. With Niederreiter, by using the algorithm seen in Sect. 2.2, she can encode up to $\ell = \left\lfloor \log_2\binom{n}{t} \right\rfloor = \left\lfloor \log_2\binom{2^m}{t} \right\rfloor$ bits together. To give an idea, we apply the usual values 10 and 50 for the parameters m and t. The formula returns that it is possible to send 284 bits in one time, while McEliece allows about twice as many. But, if instead of absolute terms we investigate the speed at which a ciphertext is sent to the receiver, we see that to dispatch the

[7] Possible values for λ: $3 - \frac{2k}{n}$ [39, p. 148] or between 2 and 5 [46].
[8] $\mu = 3$ [39, p. 148] or $\mu = m$ [47].

524 bits, Alice needs a 1024-bits block, corresponding to a transmission rate $R_M \cong 51{,}2\%$, whereas Niederreiter for the transfer of 284 bits only requires a 500-bits ciphertext, yielding a better rate R_N around 56,8%. If we take once more the RSA cryptosystem, we see that the plaintext and the ciphertext have on average similar lengths, hence its rate is 1.

To increase R_M a bit, one could think of a mixture of the two cryptosystems by encoding a part of the plaintext with McEliece as usual, while another part takes the place of the error vector $\mathbf{e}$ after having being transformed to allow constant weight encoding. In fact, note that when a ciphertext is being decrypted, Bob also finds $\mathbf{e}$ as a by-product. Moreover, $\mathbf{e}$ and a plaintext in Niederreiter have the same size, so Guillot's algorithm can be applied without modifications. Though, this idea has already been proposed by Riek [36] and later, in more elaborated ways, by many other authors, for example by Sendrier and Biswas [8], who developed a variant called *Hybrid McEliece*.

As usual, in the case of Sidel'nikov we have to take into account the different number of matrices, which here also influences the amount of intentional errors allowed. We recall the notation $\bar{\ell} = \left\lfloor \log_2 \binom{un}{T} \right\rfloor$, where $n = 2^m$ and $T = u(t+1) - 1$. It is clear that for $u = 1$, $\bar{\ell} = \ell$. When u assumes higher values, we notice that the ratio $\frac{T}{un}$ increases and consequently, because in practice $T < \frac{un}{2}$, also $\bar{\ell}$ does. This is equivalent to say that compared to a given Niederreiter cryptosystem and imagining to have a Sidel'nikov system with the same parameters, the latter would allow to commit some more intentional errors, i.e. to encrypt a slightly larger block of the original message; clearly, as Niederreiter does not work with a chain of matrices as the other system does, saying that the matrix E in Sidel'nikov is composed of u blocks translates into encrypting u different messages independently in the other cryptosystem. Nevertheless, a better error correction capability does not imply that the rate of a Sidel'nikov cryptosystem R_S is bigger than R_N, since by adding more blocks to the matrix E, k remains unchanged, but the number $\bar{r}$ of rows of D is greatly increased.

To give an example, in Tab. 5 we compare the rate of the Sidel'nikov cryptosystem with the two others. For this, we chose the parameter $m = 10$ as before, i.e. we use a code of length $1024u$. Because Sidel'nikov works over RM codes, we cannot take the same number t for the intentional errors. If we set $\rho = 3$, we get a $[1024, 176, 128]$ code (see Thm. 1.5.12), which on paper has a not too different error correcting capability from the Goppa code considered by McEliece, actually even higher. In spite of this, from the table we see that because of the bigger size of the syndrome, the rate dramatically drops. If, instead, a $[1024, 524, 101]$ code were used, its rate would be equal to R_N for $u = 1$ and fall below 40% already for $u = 2$. Though, as already mentioned in Sect. 3.3, the code $\mathsf{RM}(3, 10)$ should be able to correct 200 errors, giving a very good rate approaching 1 for small u.

Conclusion

To conclude this section we insert a table containing the costs of most steps needed for implementing the McEliece and the Niederreiter cryptosystems. The values are taken from this section, except for the last row, whose entries have been computed thanks to the results obtained in Sect. 4.3. Regarding the chosen values for t, Tab. 6 (reported on p. 65) includes the ones proposed by McEliece in [25] (first column), the ones yielding the highest work factor for each m (even columns) and the lowest making $\mathsf{WF}(t)$ exceed 2^{95} (third and fifth columns) which, according to [46], can be considered as sufficiently large to prevent the MRA for the whole decade, even if much larger parameter values are already commonly employed [5, p. 3]. These values can be taken as security indices for the Niederreiter cryptosystem too.

Code (int. errors)	$[1024, 524, 101]$ ($t = 50$)		$\mathsf{RM}(3, 10)$ ($t = 63$)		$\mathsf{RM}(3, 10)$ ($t = 200$)	
u	$\bar{\ell}$ [bits]	R_S [%]	$\bar{\ell}$ [bits]	R_S [%]	$\bar{\ell}$ [bits]	R_S [%]
1	284	56,8	337	39,741	724	85,377
2	575	37,73	682	36,432	1455	77,724
3	868	34,066	1027	35,463	2186	75,483
4	1160	32,475	1372	35	2918	74,439
5	1452	31,593	1717	34,729	3649	73,807
6	1744	31,032	2062	34,551	4381	73,408
7	2037	30,659	2408	34,439	5112	73,112
8	2329	30,373	2753	34,344	5844	72,904
9	2621	30,154	3098	34,27	6575	72,732
...						
15	4375	29,489	5170	34,049	10964	72,208
...						
30	8761	29,014	10350	33,886	21938	71,824
...						
50	14609	28,828	17258	33,823	36569	71,67

Table 5 – Rate of the Sidel'nikov cryptosystem when using the parameters of McEliece and the code $\mathsf{RM}(3, 10)$ in theory and in practice.

From the table we see that for both cryptosystems, the use of higher m combined with low t increases the transmission rate significantly and both the encryption and decryption parts can be performed faster in proportion to the plaintext length. To be more precise, for raising m McEliece's rates overtake the ones of Niederreiter and the ciphertexts can be decrypted more efficiently; on the contrary, the latter shows better performances in the encryption part. The main disadvantages of both cryptosystems are the keys sizes, which turn out to be already very big with McEliece's original parameters and occupy considerably more space with the chosen values for m and t.

Note that in Tab. 6 the key sizes and the encryption cost of the McEliece cryptosystem have been computed supposing the matrices to be non-systematic. Though, even if the systematic forms were used, the public keys would be as large as in Niederreiter, while the private keys would be bigger, with a greater difference for small t. This is due to the fact that in practice codes with low redundancy ($r < \frac{n}{2}$) are used, with the consequence that the $r \times r$ matrix M is smaller than the $k \times k$ matrix S when using the same parameters for both ciphers. Regarding the relative encryption costs, we see that Niederreiter is decidedly more advantageous than the non-systematic McEliece. Anyway, also in this case the systematic form of G' does not make it more attractive than Niederreiter, since its running time per message bit is still on average five times higher.

Going back to the rate of the McEliece cryptosystem, it is interesting to note that the three values in the even columns are very close to each other. In fact, when t gets the value making the MRA the hardest for a given m, R_M lies around 63–64% even for higher and lower values of this parameter. Concretely, the rate is equal to 65,6% for $m = 8$ and slowly falls to 62,9% for $m = 16$.

4.5 Sidel'nikov-Šestakov attack

As we have already seen in Sect. 3.2, Niederreiter, in contrast to McEliece, did not recommend a particular class of codes to generate his cryptosystem, but simply required the code to have a large enough error correcting capability and the existence of some algorithm allowing a fast decryption procedure, listing some examples of codes meeting these conditions. He did not forget to mention Goppa codes, but among them he also suggested the employment of RS codes.

While both McEliece and Niederreiter cryptosystems are still believed to be secure whenever they are based over Goppa codes (see Sect. 3.1), in 1992 Sidel'nikov and Šestakov [41] discovered a structural attack to the Niederreiter cryptosystem, that is an algorithm able of cryptanalysing efficiently public key cryptosystems constructed over generalized RS codes, in which they show how to find in $\mathcal{O}(r^4 + rn)$ bit operations two matrices which can be treated as private keys starting from the public matrix H'. Actually, the complexity should be even lower, since the authors themselves affirm that for its computation no particular "fast algorithms" were used. Indeed, few time later Gabidulin and Kjelsen proposed a slightly modified version of the attack ([15, p. 39] and [16]) with running time proportional to $\mathcal{O}(n^3)$.

Throughout this section we refer to the documents [15], [16] and [32]; though, the main source material is represented by the English version of the paper written by Sidel'nikov and Šestakov [41], even if to obviate to some small incorrectnesses we also refer to the original publication in Russian [42] without further specification. Any citation of the former will always imply the latter too.

Recall. We briefly repeat the notation adopted in Sect. 1.3. A GRS code has a parity-check matrix of the form

$$\mathsf{H} = \mathsf{H}(\mathbf{a};\mathbf{z}) = \begin{pmatrix} z_1 a_1^0 & z_2 a_2^0 & \cdots & z_n a_n^0 \\ z_1 a_1^1 & z_2 a_2^1 & \cdots & z_n a_n^1 \\ z_1 a_1^2 & z_2 a_2^2 & \cdots & z_n a_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ z_1 a_1^{r-1} & z_2 a_2^{r-1} & \cdots & z_n a_n^{r-1} \end{pmatrix}, \tag{4.3}$$

where the a_j and the z_j are elements of $\mathbb{F}_q$, $a_i \neq a_j$ for $i \neq j$, $z_j \neq 0$ $\forall j$. $\mathsf{H}(\mathbf{a};\mathbf{z})$ can be split into two matrices $\mathsf{X}(\mathbf{a})$ and $\mathsf{Y}(\mathbf{z})$, where the former is composed of the powers of the a_j and the latter is diagonal and contains the z_j. Also remember that the matrix $\mathsf{H}(\mathbf{a};\mathbf{z})$ defines a code of dimension k which is in general not equal to $\mathsf{GRS}_r(\mathbf{a};\mathbf{z})$ (this is actually its dual code) and hence the two notations should not be confused.

Goal of the algorithm

For this section we allow the vector $\mathbf{a}$ to contain also an element which we will call ∞, with the property that

$$\infty^i = \begin{cases} 0 & \text{for } i = 0, \ldots, r-2; \\ 1 & \text{for } i = r-1. \end{cases} \tag{4.4}$$

As a consequence, the corresponding column in X has a 1 in the last row and 0's elsewhere; thus:

$$\mathsf{H}(a_1,\ldots,a_{j-1},\infty,a_{j+1},\ldots,a_n;\mathbf{z}) = \begin{pmatrix} z_1 & \cdots & z_{j-1} & 0 & z_{j+1} & \cdots & z_n \\ z_1a_1 & \cdots & z_{j-1}a_{j-1} & 0 & a_{j+1}a_{j+1} & \cdots & z_na_n \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots & 0 & \vdots & \ddots & \vdots \\ z_1a_1^{r-1} & \cdots & z_{j-1}a_{j-1}^{r-1} & z_j & z_{j+1}a_{j+1}^{r-1} & \cdots & z_na_n^{r-1} \end{pmatrix}.$$

For convenience, we introduce the notation $\mathbb{F}_{q,\infty} := \mathbb{F}_q \cup \{\infty\}$. The presence of this symbol justifies the more general form of H including the terms with exponents 0. For the computations in this set we use the customary rules applied when calculating with ∞.

From [22], [31, p. 220] and [40, p. 193] we know that a successful attack against a McEliece cryptosystem also breaks Niederreiter, so it is enough to cryptanalyse either of them. In Alg. 3 we saw that a public key for the Niederreiter cryptosystem is given by multiplying a parity-check matrix by two other matrices M and P. The latter is a permutation matrix which just changes the order of the columns of MH, so we can ignore it without loss of generality. If M is such an $r \times r$ scrambling matrix, then the effect of the ith row of M in the product MH can be represented as a polynomial $\varphi_i \in (\mathbb{F}_q)_{<r}[x]$ over $\mathbb{F}_q$ of degree less than r. If the entries of M are denoted as $m_{i,j}$, then the coefficients of the ith polynomial $\varphi_i(x) = \varphi_{i,0}x^0 + \varphi_{i,1}x^1 + \ldots + \varphi_{i,r-1}x^{r-1}$ are $m_{i,1},\ldots,m_{i,r}$, that is $m_{i,j} = \varphi_{i,j-1}$ $\forall i,j \in \{1,\ldots,r\}$. Thus:

$$\mathsf{H}' = \mathsf{MH} = \begin{pmatrix} z_1\varphi_1(a_1) & z_2\varphi_1(a_2) & \cdots & z_n\varphi_1(a_n) \\ z_1\varphi_2(a_1) & z_2\varphi_2(a_2) & \cdots & z_n\varphi_2(a_n) \\ \vdots & \vdots & \ddots & \vdots \\ z_1\varphi_r(a_1) & z_2\varphi_r(a_2) & \cdots & z_n\varphi_r(a_n) \end{pmatrix}. \tag{4.5}$$

It is clear that the r polynomials are linearly independent, as the rows of M are. Concerning the treatment of ∞, by (4.4) we have that $\varphi_i(\infty) = \varphi_{i,r-1}$.

Now assume the above matrices M and H — or equivalently M and the vectors $\mathbf{a}$ and $\mathbf{z}$ — are the private keys of a Niederreiter cryptosystem and Eve, an eavesdropper, wants to discover them. All that she has is the public key H'. The goal of the algorithm is to find some matrices $\tilde{\mathsf{M}}$ and $\tilde{\mathsf{H}} := \mathsf{H}(\tilde{\mathbf{a}};\tilde{\mathbf{z}})$ such that $\mathsf{H}' = \tilde{\mathsf{M}}\tilde{\mathsf{H}}$, where $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{z}}$ have the same properties as $\mathbf{a}$ and $\mathbf{z}$, including $\tilde{\mathbf{a}} \in \mathbb{F}_{q,\infty}^n$. In this case, we say that the triple $(\tilde{\mathsf{M}};\tilde{\mathbf{a}};\tilde{\mathbf{z}}) = (\tilde{\mathsf{M}};\tilde{a}_1,\ldots,\tilde{a}_n;\tilde{z}_1,\ldots,\tilde{z}_n)$ is a solution of the equation $\mathsf{H}' = \tilde{\mathsf{M}}\tilde{\mathsf{H}}$.

Form of a solution

Let us start with the assumption that $(\mathsf{N};\mathbf{b};\mathbf{y})$ is some solution of this equation, i.e. $\mathsf{MH}(\mathbf{a};\mathbf{z}) = \mathsf{NH}(\mathbf{b};\mathbf{y})$.[9] We construct an $r \times r$ matrix N_1 over $\mathbb{F}_q$ with entries $n_{i,j}$, $(i,j) \in \{1,\ldots,r\}^2$ and choose $p \in \mathbb{F}_q^*$ and $s \in \mathbb{F}_q$ such that

$$(pX+s)^{i-1} = \sum_{\ell=1}^{r} n_{i,\ell}X^{\ell-1} \overset{\circledast}{=} \sum_{\ell=1}^{i} n_{i,\ell}X^{\ell-1} \quad \forall i = 1,\ldots,r. \tag{4.6}$$

[9] Gabidulin and Kjelsen [15, 16] underline the possibility of M and N resp. $\mathbf{z}$ and $\mathbf{y}$ to be different, while Sidel'nikov and Šestakov ambiguously maintain the same letters in both cases.

The equality at $\circledast$ holds since $n_{i,\ell} = 0$ when $\ell > i$; that is:

$$\begin{aligned} 1 &= \sum_{\ell=1}^{1} n_{1,\ell} X^{\ell-1} = n_{1,1} \\ pX + s &= n_{2,1} + n_{2,2} X \\ (pX+s)^2 &= n_{3,1} + n_{3,2} X + n_{3,3} X^2 \\ &\vdots \\ (pX+s)^{r-1} &= n_{r,1} + n_{r,2} X + n_{r,3} X^2 + \ldots + n_{r,r} X^{r-1}. \end{aligned}$$

Hence, N_1 is a lower triangular matrix and it is $\det(\mathsf{N}_1) = \prod_{j=1}^{r} n_{j,j} = \prod_{j=1}^{r-1} p^j \neq 0$. Now let $\mathbf{d} = d_1, \ldots, d_n$ be a vector with elements in $\mathbb{F}_q$. We compute the matrix $\mathsf{N}_1 \cdot \mathsf{H}(\mathbf{b}; \mathbf{d} \star \mathbf{y})$, where $\mathbf{d} \star \mathbf{y}$ is a short cut for $(d_1 y_1, \ldots, d_n y_n)$ as in Def. 1.5.2:

$$\mathsf{N}_1 \mathsf{H}(\mathbf{b}; \mathbf{d} \star \mathbf{y}) = \begin{pmatrix} d_1 y_1 n_{1,1} b_1^0 & d_2 y_2 n_{1,1} b_2^0 & \cdots & d_n y_n n_{1,1} b_n^0 \\ d_1 y_1 (n_{2,1} b_1^0 + n_{2,2} b_1^1) & d_2 y_2 (n_{2,1} b_2^0 + n_{2,2} b_2^1) & \cdots & d_n y_n (n_{2,1} b_n^0 + n_{2,2} b_n^1) \\ \vdots & \vdots & \ddots & \vdots \\ d_1 y_1 \sum_{\ell=1}^{r} n_{r,\ell} b_1^{\ell-1} & d_2 y_2 \sum_{\ell=1}^{r} n_{r,\ell} b_2^{\ell-1} & \cdots & d_n y_n \sum_{\ell=1}^{r} n_{r,\ell} b_n^{\ell-1} \end{pmatrix}.$$

We now assume $b_j \in \mathbb{F}_q$ and put an entry of this matrix equal to one of the matrix

$$\mathsf{H}(pb_1 + s, \ldots, pb_n + s; \mathbf{y}) = \left(y_j (pb_j + s)^{i-1} \right)_{(i,j) \in \{1,\ldots,r\} \times \{1,\ldots,n\}};$$

from (4.6) follows immediately $d_j = 1 \ \forall j$. If $b_j = \infty$, the components 1 to $r-1$ of column j are equal to 0 and in row r we have:

$$d_j y_j \sum_{\ell=1}^{r} n_{r,\ell} \cdot \underbrace{\infty^{\ell-1}}_{=0 \text{ for } \ell<r} = y_j (p\infty + s)^{r-1} \overset{\substack{(4.4),\\ y_j \neq 0 \ \forall j}}{\Longrightarrow} d_j n_{r,r} \infty^{r-1} = \infty^{r-1} \Longrightarrow$$
$$\Longrightarrow d_j p^{r-1} = 1 \Longrightarrow d_j = p^{1-r}.$$

Hence, $\mathsf{N}_1 \mathsf{H}(\mathbf{b}; \mathbf{d} \star \mathbf{y}) = \mathsf{H}(pb_1 + s, \ldots, pb_n + s; \mathbf{y})$; by bringing N_1 and the d_j to the right hand side and multiplying by N, we get

$$\mathsf{M}\mathsf{H}(\mathbf{a}; \mathbf{z}) = \mathsf{N}\mathsf{H}(\mathbf{b}; \mathbf{y}) = \mathsf{N}\mathsf{N}_1^{-1} \mathsf{H}(pb_1 + s, \ldots, pb_n + s; d_1^{-1} y_1, \ldots, d_n^{-1} y_n), \tag{4.7}$$

i.e. the triple composed by $\mathsf{N}\mathsf{N}_1^{-1}$ and the two vectors in brackets is a solution to our problem.

Another solution can be found by defining the matrix $\mathsf{N}_2 := (\delta_{i,r+1-i})_{i \in \{1,\ldots,r\}}$, where $\delta_{i,j}$ is the Kronecker symbol, that is the matrix with 1 on the secondary diagonal and 0 elsewhere. If a matrix is postmultiplied to N_2, the order of the rows of that matrix is inverted. Assume $b_j \in \mathbb{F}_q^* \ \forall j$; as above, we solve an equation with respect to $\tilde{d}_j$:

$$\begin{aligned} &\mathsf{N}_2 \mathsf{H}(\mathbf{b}; \tilde{\mathbf{d}} \star \mathbf{y}) = \mathsf{H}(b_1^{-1}, \ldots, b_n^{-1}; \mathbf{y}) \Longrightarrow \\ &\Longrightarrow \tilde{d}_j y_j b_j^{r-i} = y_j b_j^{1-i} \quad \forall (i,j) \in \{1,\ldots,r\} \times \{1,\ldots,n\} \Longrightarrow \tilde{d}_j = b_j^{1-r}. \end{aligned} \tag{4.8}$$

The cases where $b_j \in \{0, \infty\}$ require a more careful treatment. We rewrite the result obtained in (4.8) deleting the y_j and without multiplying the exponents, since the form

$$\tilde{d}_j b_j^{r-i} = \left(\tfrac{1}{b_j}\right)^{i-1}$$

enables the use of the particular computation rules involving 0 and ∞. Let in the following i and j denote as usual the row respectively the column number:

- $b_j = \infty$:
  - $i = 1$: $\tilde{d}_j \infty^{r-1} = \left(\frac{1}{\infty}\right)^0 \Longrightarrow \tilde{d}_j = 0^0 = 1$;
  - $i \in \{2, \ldots, r\}$: $\tilde{d}_j \infty^{r-i} = \left(\frac{1}{\infty}\right)^{i-1} \Longrightarrow 0\tilde{d}_j = 0$.
- $b_j = 0$:
  - $i = r$: $\tilde{d}_j 0^0 = \left(\frac{1}{0}\right)^{r-1} \Longrightarrow \tilde{d}_j = \infty^{r-1} = 1$;
  - $i \in \{1, \ldots, r-1\}$: $\tilde{d}_j 0^{r-i} = \left(\frac{1}{0}\right)^{i-1} \Longrightarrow 0\tilde{d}_j = 0$.

Then, when $b_j \in \{0, \infty\}$ we put $\tilde{d}_j = 1$. With this, in (4.8) we have equality for any b_j, which means that also the triple

$$(\mathsf{N}\mathsf{N}_2^{-1}; b_1^{-1}, \ldots, b_n^{-1}; \tilde{d}_1^{-1} y_1, \ldots, \tilde{d}_n^{-1} y_n) \tag{4.9}$$

possesses the requested characteristics, provided that the triple $(\mathsf{N}; \mathbf{b}; \mathbf{y})$ does.

We can now put together the two results obtained in (4.7) and (4.9) to get a more general form of triples returning the public key MH.

Lemma 4.5.1 ([32, 41]). *Let*

$$\begin{array}{rccc} \psi\colon & \mathbb{F}_{q,\infty} & \longrightarrow & \mathbb{F}_{q,\infty} \\ & \infty & \longmapsto & \frac{p}{p'} \\ & \mathbb{F}_q \ni X & \longmapsto & \frac{pX+s}{p'X+s'}, \end{array}$$

define an automorphism obtained by combining $pX + s$ and X^{-1}, where $p, p' \in \mathbb{F}_q^*$ resp. $s, s' \in \mathbb{F}_q$ as in (4.6) and $ps' \neq p's$ to avoid the numerator being a multiple of the denominator. Then, having a solution $(\mathsf{N}; \mathbf{b}; \mathbf{y})$ of $\mathsf{H}' = \tilde{\mathsf{M}}\tilde{\mathsf{H}}$, another solution is given by

$$(\mathsf{N}\mathsf{N}_\psi^{-1}; \psi(b_1), \ldots, \psi(b_n); \tilde{y}_1, \ldots, \tilde{y}_n) \tag{4.10}$$

for some matrix N_ψ and some vector $\tilde{\mathbf{y}} \in (\mathbb{F}_q^*)^n$.

Remark 4.5.2. Every automorphism can be obtained with a combination of functions of the form $pX + s$ and X^{-1}.

The function ψ can be completely defined by choosing which elements of the domain are mapped respectively to 1, 0 and ∞. Without loss of generality, let b_1, b_2 and b_3 be the preimages of these numbers. Analogously, we can wlog set them to be fixed points and get

$$(b_1, b_2, b_3) = (1, 0, \infty). \tag{4.11}$$

Then, by assigning to b_j, $j = 4, \ldots, n$, the remaining values in $\mathbb{F}_q \setminus \{0, 1\}$ (recall that the b_j are pairwise distinct) and by finding a suitable matrix N_ψ, a solution would be $(\mathsf{N}_\psi; 1, 0, \infty, b_4, \ldots, b_n; \tilde{\mathbf{y}})$ for some $\tilde{\mathbf{y}}$.

Finding the vector* b — *Part I

Consequently with this result, assume b_1, b_2 and b_3 have these three values respectively. We can now try to compute the remaining unknowns. To be applied, the attack needs the redundancy r to be less than $\frac{n}{2}$, which is quite common in practice. By equations (4.3) and (4.5) we know that the public parity-check matrix H' can be expressed as

$$(h'_{i,j})_{\substack{i=1,\dots,r,\\ j=1,\dots,n}} := \mathsf{H}' = \mathsf{N}\cdot\mathsf{H}(\mathbf{b};\mathbf{y}) = \mathsf{N}\cdot\mathsf{X}(\mathbf{b})\cdot\mathsf{Y}(\mathbf{y}) = \big(y_j\cdot f_i(b_j)\big)_{i,j}, \tag{4.12}$$

where $\mathsf{N} := \mathsf{N}_\psi$, X and Y are the matrices we aim to find and the $f_i(b_j) := \sum_{\ell=1}^{r} n_{i,\ell} b_j^{\ell-1}$ are an alternative description of the entries of NX.

We define a vector $\mathbf{c}_1 \in \mathbb{F}_q^r\backslash\{\mathbf{0}\}$ and

$$\mathbb{F}_q^n \ni \boldsymbol{\beta}_1 := (\beta_{1,1},\dots,\beta_{1,n}) := \mathbf{c}_1\mathsf{H}'. \tag{4.13}$$

The $r-1$ components $1, r+1, r+2, \dots, 2r-2$ of $\boldsymbol{\beta}_1$ must be equal to 0. This is equivalent to look for $\mathbf{c}_1$ as a non-trivial solution to the system

$$\begin{cases} c_{1,1}h'_{1,1} + c_{1,2}h'_{2,1} + \ldots + c_{1,r}h'_{r,1} = 0 \\ c_{1,1}h'_{1,r+1} + c_{1,2}h_{2,r+1} + \ldots + c_{1,r}h'_{r,r+1} = 0 \\ c_{1,1}h'_{1,r+2} + c_{1,2}h'_{2,r+2} + \ldots + c_{1,r}h'_{r,r+2} = 0 \\ \qquad\qquad\vdots \\ c_{1,1}h'_{1,2r-2} + c_{1,2}h_{2,2r-2} + \ldots + c_{1,r}h'_{r,2r-2} = 0 \end{cases} \tag{4.14}$$

having $r-1$ equations and r unknowns. Using (4.12) we can rewrite $\boldsymbol{\beta}_1$ as

$$\boldsymbol{\beta}_1 = \left(\sum_{i=1}^{r} c_{1,i}h'_{i,1},\dots,\sum_{i=1}^{r} c_{1,i}h'_{i,n}\right) =: \big(y_1F_1(b_1),\dots,y_nF_1(b_n)\big) =: \mathbf{F}_1\cdot\mathsf{Y}(\mathbf{y}), \tag{4.15}$$

where $\mathbf{F}_1 = \mathbf{c}_1\cdot\mathsf{N}\cdot\mathsf{X}(\mathbf{b}) \in \mathbb{F}_q^n$ is the vector composed of the $F_1(b_j) := \mathbf{c}_1\cdot\big(f_1(b_j),\dots,f_r(b_j)\big)^\top$. Note that by (4.13) all the $\beta_{1,j}$ can be easily computed even if neither $\mathbf{F}_1$ nor $\mathsf{Y}(\mathbf{y})$ are known. Since $\mathbf{c}_1$ has components in $\mathbb{F}_q$, the polynomial $F_1(x)$ has at most the same degree as the polynomials $f_i(x)$, i.e. $r-1$. Moreover, by (4.13) and (4.15):

$$F_1(b_j) = \frac{\beta_{1,j}}{y_j}$$

and hence, because $y_j \neq 0\ \forall j$

$$F_1(b_j) = 0 \iff \beta_{1,j} = 0 \overset{(4.14)}{\iff} j \in \{1, r+1, \dots, 2r-2\} =: \mathcal{I}_1,$$

from which we can deduce that the roots of $F_1(x)$ are the b_j's where j is an element of $\mathcal{I}_1$ and consequently that $\deg F_1$ is exactly $r-1$.

Analogously, we now define two vectors $\mathbf{c}_2 \in \mathbb{F}_q^r\backslash\{\mathbf{0}\}$ and $\boldsymbol{\beta}_2 := \mathbf{c}_2\mathsf{H}'$ for which it holds

$$\beta_{2,j} = 0 \iff j \in \{2, r+1, r+2, \dots, 2r-2\} =: \mathcal{I}_2;$$

as before, we can introduce a polynomial $F_2(x)$ of degree $r-1$ such that $\beta_{2,j} = y_j F_2(b_j)$ whose roots are the b_j with $j \in \mathcal{I}_2$.

If a_1 and a_2 are some constants (unknown to the attacker) in $\mathbb{F}_q$, we denote

$$F_\ell(x) = a_\ell \cdot \prod_{j \in \mathcal{I}_\ell} (x - b_j) = a_\ell x^{r-1} + \dots, \quad \ell \in \{1, 2\}.$$

In this way, the computation of a special image of the two polynomials is straightforward. Let $b_3 = \infty$ as in equation (4.11):

$$\beta_{\ell,3} = y_3 F_\ell(b_3) = y_3 F_\ell(\infty) = a_\ell y_3.$$

Note that the sets $\mathcal{I}_1$ and $\mathcal{I}_2$ differ in only one element; so, for $j \notin (\mathcal{I}_2 \cup \{1\})$ it is

$$B_j := \frac{\beta_{1,j}}{\beta_{2,j}} = \frac{y_j F_1(b_j)}{y_j F_2(b_j)} = \frac{a_1(b_j - b_1)}{a_2(b_j - b_2)} \overset{(4.11)}{=} \frac{a_1(b_j - 1)}{a_2 b_j};$$

in this way, the unknown y_j disappears. In particular:

$$B_3 = \frac{F_1(\infty)}{F_2(\infty)} = \frac{a_1}{a_2}$$

and because B_3 is the quotient of the two known elements $\beta_{1,3}$ and $\beta_{2,3}$, it can be computed and used to replace the unknown ratio $\frac{a_1}{a_2}$. By combining the previous two results, we get

$$B_j = B_3 \frac{b_j - 1}{b_j} \Longrightarrow b_j = \frac{B_3}{B_3 - B_j} \quad \forall j \notin (\mathcal{I}_2 \cup \{1, 3\}), \tag{4.16}$$

that is the eavesdropper is able to find all the b_j for $j \in \{4, \dots, r, 2r-1, \dots, n\}$.

Finding the vector* $\mathbf{b}$ — *Part II

In order to find the values of b_j for the remaining indices, we now repeat the same proceeding with $\mathcal{I}_3 := \{1, 3, 4, \dots, r\}$ and $\mathcal{I}_4 := \{2, \dots, r\}$: for $\ell \in \{3, 4\}$ let the $\mathbf{c}_\ell$ be vectors in $\mathbb{F}_q^r \setminus \{\mathbf{0}\}$ such that

$$\boldsymbol{\beta}_\ell := \mathbf{c}_\ell \mathsf{H}' \quad \text{with} \quad \beta_{\ell,j} = 0 \iff j \in \mathcal{I}_\ell \quad \text{(known)}$$

and let

$$F_\ell(b_j) = \sum_{\nu=0}^{r-1} F_{\ell,\nu} x^\nu := \frac{\beta_{\ell,j}}{y_j} \quad \text{(unknown)}.$$

The sets $\mathcal{I}_3$ and $\mathcal{I}_4$ are as usual composed of $r-1$ elements, but since they both contain the index 3, we know that $b_3 = \infty$ is a root of F_3 and F_4. Applying (4.4), this means that for $\ell \in \{3, 4\}$

$$F_{\ell,r-1} = F_\ell(\infty) = 0,$$

implying that both polynomials have degree at most $r-2$. Since $\#(\mathcal{I}_3 \setminus \{3\}) = \#(\mathcal{I}_4 \setminus \{3\}) = r-2$, the two polynomials can be represented as

$$F_\ell(x) = a_\ell \cdot \prod_{j \in \mathcal{I}_\ell \setminus \{3\}} (x - b_j) = a_\ell x^{r-2} + \dots, \quad \ell \in \{3, 4\}$$

and similarly as above, for $j \notin (\mathcal{I}_4 \cup \{1\})$ we define

$$B'_j := \frac{\beta_{3,j}}{\beta_{4,j}} = \frac{y_j F_3(b_j)}{y_j F_4(b_j)} = \frac{a_3(b_j - b_1)}{a_4(b_j - b_2)} \overset{(4.11)}{=} \frac{a_3(b_j - 1)}{a_4 b_j} \overset{(4.16)}{=} \frac{a_3 B_j}{a_4 B_3},$$

where B_3 has been computed in the first part of the attack. Note that not all of the B_j's can be calculated as $\frac{\beta_{1,j}}{\beta_{2,j}}$; though, the value of just one of them is enough to even substitute the unknown $\frac{a_3}{a_4}$ with another expression. Let j be some element of the intersection

$$\{4, \ldots, r, 2r-1, \ldots, n\} \cap \{r+1, \ldots, n\} = \{2r-1, \ldots, n\},$$

e.g. $j = n$:

$$B'_n = \frac{\beta_{3,n}}{\beta_{4,n}} = \frac{a_3 B_n}{a_4 B_3} \Longrightarrow \frac{a_3}{a_4} = \frac{B_3 \beta_{3,n}}{B_n \beta_{4,n}} \Longrightarrow B'_j = \frac{B_3 \beta_{3,n}}{B_n \beta_{4,n}} \frac{b_j - 1}{b_j}. \tag{4.17}$$

This returns a description for b_j also when the indices are not included in the other half of the proceeding. Summarizing the results obtained in equations (4.11), (4.16) and (4.17), we now have a complete description for b_j, that is we have found a valid matrix $\mathsf{X}(\mathbf{b})$:

$$b_j = \begin{cases} 1 & \text{if } j = 1; \\ 0 & \text{if } j = 2; \\ \infty & \text{if } j = 3; \\ \frac{B_3}{B_3 - B_j} & \text{if } j \in \{4, \ldots, r, 2r-1, \ldots, n\}; \\ \frac{B_3}{B_3 - \tilde{B}_j} & \text{if } j \in \{r+1, \ldots, 2r-2\}, \end{cases}$$

where $\tilde{B}_j := \dfrac{B_n \beta_{4,n} \beta_{3,j}}{\beta_{3,n} \beta_{4,j}}$.

If n is strictly smaller than q, it is possible to have a vector $\mathbf{b}$ not containing the element ∞ (say $\bar{\mathbf{b}}$) but which still produces a valid solution for the attack: consequently with Lemma 4.5.1, it suffices to choose some element $A \in \mathbb{F}_q$ such that $A \neq b_j \ \forall j \in \{1, \ldots, n\}$ and set

$$\bar{b}_j := \frac{1}{A - b_j}.$$

***Finding the vector* y**

After the vector $\mathbf{b}$, the attacker now needs to find the vector $\mathbf{y}$. Similarly to what we have done for finding the b_j's, solve a system of r equations to find a vector $\mathbf{C} \in \mathbb{F}_q^{r+1} \backslash \{\mathbf{0}\}$ satisfying

$$\begin{cases} C_1 h'_{1,1} + C_2 h'_{1,2} + \ldots + C_{r+1} h'_{1,r+1} = 0 \\ C_1 h'_{2,1} + C_2 h'_{2,2} + \ldots + C_{r+1} h'_{2,r+1} = 0 \\ \qquad\qquad \vdots \\ C_1 h'_{r,1} + C_2 h'_{r,2} + \ldots + C_{r+1} h'_{r,r+1} = 0. \end{cases} \tag{4.18}$$

If we use a notation already encountered in Sect. 2.3 and define $\mathsf{H}'_{\leftarrow r+1}$ as the matrix composed of the $r+1$ leftmost columns of H', the system can be rewritten as $\mathsf{H}'_{\leftarrow r+1} \cdot \mathbf{C}^\top = \mathbf{0}^\top$ and by equation (4.12) it holds

$$\mathbf{0}^\top = \mathsf{H}'_{\leftarrow r+1}\mathbf{C}^\top = \mathsf{N}\big(\mathsf{X}(\mathbf{b})\mathsf{Y}(\mathbf{y})\big)_{1,\ldots,r+1}\mathbf{C}^\top = \mathsf{N}\mathsf{X}(b_1,\ldots,b_{r+1})\mathsf{Y}(y_1,\ldots,y_{r+1})\mathbf{C}^\top$$

and hence

$$\mathbf{0}^\top = \mathsf{N}^{-1}\mathbf{0}^\top = \mathsf{X}(b_1,\ldots,b_{r+1}) \cdot \big((y_1,\ldots,y_{r+1})^\top \star \mathbf{C}^\top\big), \tag{4.19}$$

where $\mathsf{X}(b_1,\ldots,b_{r+1})$ is the submatrix of $\mathsf{X}(\mathbf{b})$ composed only of its leftmost $r+1$ columns, $\mathsf{Y}(y_1,\ldots,y_{r+1})$ is the diagonal matrix containing those elements and "$\star$" is the operation introduced in Def. 1.5.2. In this way, the unknown matrix N has been removed from the equality and among the remaining elements composing it, only the y_j's still have to be found out. If the equality in (4.19) is translated into a system, though, there are only r equations with $r+1$ unknowns. To work around this little difficulty, one of them (say y_J for some $J \in \{1,\ldots,r+1\}$) can be assigned a priori the value 1: in fact, if all y_j's are divided by y_J, then it suffices to substitute the matrix N of the product $\mathsf{N} \cdot \mathsf{H}(\mathbf{b}, \frac{1}{y_J}\mathbf{y})$ through $y_J\mathsf{N}$ to get the same result. In this way, we get the system

$$\begin{cases} C_1 b_1^0 y_1 + \ldots + C_{J-1} b_{J-1}^0 y_{J-1} + C_{J+1} b_{J+1}^0 y_{J+1} + \ldots + C_{r+1} b_{r+1}^0 y_r = -C_J b_J^0 \\ \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ C_1 b_1^{r-1} y_1 + \ldots + C_{J-1} b_{J-1}^{r-1} y_{J-1} + C_{J+1} b_{J+1}^{r-1} y_{J+1} + \ldots + C_{r+1} b_{r+1}^{r-1} y_{r+1} = -C_J b_J^{r-1} \end{cases}$$

having r equations and r unknowns. About the existence of the unique solution, put the system in matrix form:

$$\mathsf{X}(b_i)(C_{ii})(y_i)^\top = (-C_J b_J^j), \tag{4.20}$$

where $i \in \{1,\ldots,J-1,J+1,\ldots,r+1\}$ and $j \in \{0,\ldots,r-1\}$. Observe that $\mathsf{X}(b_i)$ is a square Vandermonde matrix, thus its determinant is different from 0. This is obviously true even if the element ∞ is contained in it. Moreover, for the public parity-check matrix it holds that $\operatorname{rank}(\mathsf{H}') = r$, thus, because in system (4.18) there are the $r+1$ leftmost columns of H', the C_ℓ's are such that they give a linear combination of these columns summing to $\mathbf{0}^\top$, hence it must be $C_\ell \neq 0\ \forall \ell \in \{1,\ldots,r+1\}$. As a consequence, $\det\big(\mathsf{X}(b_i)(C_{ii})\big) \neq 0$ and the vector on the right of the equals sign is also different from $\mathbf{0}^\top$.

Finding the matrix N

The last thing the eavesdropper wants to discover is the matrix $\mathsf{N} := (n_{i,j})_{(i,j)\in\{1,\ldots,r\}^2}$. We know that $\mathsf{H}' = \mathsf{N}\mathsf{H}(\mathbf{b};\mathbf{y})$. Sadly, H cannot be inverted, but its r leftmost columns are enough to describe the $n_{i,j}$ uniquely. Let $i \in \{1,\ldots,r\}$ be fixed; consider the system of equations

$$(n_{i,1},\ldots,n_{i,r}) \cdot \mathsf{H}(b_1,\ldots,b_r;y_1,\ldots,y_r) = (h'_{i,1},\ldots,h'_{i,r}),$$

where $(n_{i,1},\ldots,n_{i,r})$ represents the ith row of N, $\mathsf{H}(b_j;y_j) = \mathsf{X}(b_j)\mathsf{Y}(y_j)$ for $j = 1,\ldots,r$ and the $h'_{i,j}$ are a part of the ith row of H'. The system has r equations and r unknowns. The matrix $\mathsf{H}(b_j;y_j)$ $(j = 1,\ldots,r)$ is completely known. The arguments to prove its invertibility are similar to the case of the previous system (4.20): the submatrix $\mathsf{X}(b_j)$ inherits the Vandermonde structure from $\mathsf{X}(\mathbf{b})$. Remembering that $y_j \neq 0\ \forall j \in \{1,\ldots,n\}$, we get that also $\det\big(\mathsf{H}(b_j;y_j)\big)$ is different from 0. Repeat the process for every $i \in \{1,\ldots,r\}$.

***The last components of* $\mathbf{y}$**

With the recovering of N, the attack is actually not yet complete: the numbers $y_{r+2}, \ldots, y_n$ are missing. Though, Eve has enough information about the structure of the code to make the task very easy. Having discovered the whole matrix N, we can compute $\mathsf{N}^{-1}\mathsf{H}'$ and get

$$\mathsf{X}(\mathbf{b})\mathsf{Y}(\mathbf{y}) = \mathsf{N}^{-1}\mathsf{H}' \Longrightarrow (y_j b_j^{i-1})_{(i,j)} = \left(\sum_{\ell=1}^{r} \bar{n}_{i,\ell} h'_{\ell,j}\right)_{(i,j)},$$

where $\mathsf{N}^{-1} := (\bar{n}_{i,\ell})$, $i \in \{1, \ldots, r\}$ and $j \in \{1, \ldots, n\}$. The eavesdropper can now select any row of the two resulting matrices (i.e. choose a fixed $I \in \{1, \ldots, r\}$) and compute

$$y_j = b_j^{1-I} \sum_{\ell=1}^{r} \bar{n}_{I,\ell} h'_{\ell,j}$$

for all $j \in \{r+2, \ldots, n\}$.

	McEliece						Niederreiter					
m	10	10	11	11	12	12	10	10	11	11	12	12
t	50	37	23	68	15	126	50	37	23	68	15	126
Plaintext length [bits]	524	654	1795	1300	3916	2584	284	225	178	436	139	807
Prv. key size [kB][10]	232	268	1387	1068	6019	4255	195	178	589	716	2189	2871
Pub. key size [kB][10]	67,1	83,7	459,5	332,8	2005	1323	32,8	30,2	56,8	121,6	88,1	488,4
Encr. cost [bit ops][10]	268'288	334'848	1'838'080	1'331'200	8'019'968	5'292'032	12793	8743	5100	32287	2581	120'186
Rel. encr. cost [bit ops/msg. bit][10]	512	512	1024	1024	2048	2048	45	38,7	28,7	73,9	18,6	149
Ciphertext length [bits]	1024	1024	2048	2048	4096	4096	500	370	253	748	180	1512
Transm. rate [%]	51,2	63,9	87,6	63,5	95,6	63,1	56,8	60,8	70,4	58,3	77,2	53,3
Decr. cost[11] [bit ops]	1'012'000	652'680	646'162	2'650'912	802'080	10'765'440	750'000	410'700	192'027	1'678'512	97200	6'585'432
Rel. decr. cost [bit ops/msg. bit]	1931	998	360	2039	205	4166	16652	10604	6702	22710	5234	10263
MRA security ($\log_2(\mathsf{WF})$)	75,05	78,27	96,35	126,26	96,77	212,98	—	—	—	—	—	—

Table 6 – Comparison of the application costs of the McEliece and the Niederreiter cryptosystems when using the same $[2^m, 2^m - mt, 2t+1]$ codes, in function of some choices for the parameters m and t. For comparison, the RSA cryptosystem with modulus size of 1024 bits requires a relative encryption running time of 2403 and a relative decryption running time of 738'113 bit operations [24, p. 588], but the private resp. the public key would occupy only 1 resp. 2 kB.

[10] Non-systematic for McEliece and systematic for Niederreiter.

[11] By [39, p. 148]: $(3n-2k)r$ for McEliece and $3r^2$ for Niederreiter.

Chapter 5

A variant of the McEliece cryptosystem

This chapter is based on paper [2], in which the authors present a modified version of the McEliece resp. the Niederreiter cryptosystem obtained by subsituting the permutation matrix with a matrix of different type. After a presentation of the paper, we will discuss a possible way to increase the security of the cryptosystem. We will conclude the thesis with an attack to that cipher which was found in spring 2012 [17].

This new variant of McEliece cryptosystem was proposed very recently by Baldi, Bianchi, Chiaraluce, Rosenthal and Schipani [2]. The idea underlying the work is the introduction of an invertible transformation matrix Q taking the place of the permutation P. This matrix Q should offer the possibility of increasing the security of the system through the combination of a dense and of a sparse component. In the following, if not differently indicated, we assume to be working over a general finite field $\mathbb{F}_q$.

5.1 Description of the modified cryptosystems

Let n and z be natural numbers with $z \leqslant n$. Define $\mathsf{a}_1, \ldots, \mathsf{a}_w, \mathsf{b}_1, \ldots, \mathsf{b}_w$ to be some matrices of size $z \times n$, as well as $\mathsf{p}_1, \ldots, \mathsf{p}_v$ to be $n \times n$ generalized permutation matrices, i.e. matrices having only one non-zero entry in each row and in each column. Then the matrix Q has the following form:

$$\mathsf{Q} := \mathsf{R} + \mathsf{T}, \quad \text{where}$$

$$\mathsf{R} := \left(\mathsf{a}_1^\top \mid \cdots \mid \mathsf{a}_w^\top\right) \cdot \begin{pmatrix} \mathsf{b}_1 \\ \hline \vdots \\ \hline \mathsf{b}_w \end{pmatrix}, \qquad \mathsf{T} := \sum_{i=1}^{v} \mathsf{p}_i\,, \qquad \mathsf{Q}, \mathsf{T} \in \mathsf{GL}_n(\mathbb{F}_q), \tag{5.1}$$

such that R is dense and T sparse. Observe that it is not necessary to set $n = wz$. So, given a generator and a parity-check matrix of a code, the public keys of a McEliece and of a Niederreiter cryptosystem would be respectively $\mathsf{G}' = \mathsf{SGQ}^{-1}$ and $\mathsf{H}' = \mathsf{MHQ}^\top$. The presence of inversion and transposition will be explained in Alg. 5 and 6.

The reason for combining two different kinds of matrices to build Q relies in the fact that in presence of dense matrices the search of low-weight codewords is made more tricky, while a permutation matrix, as we have already seen in the other cryptosystems, makes the eavesdropper's task slightly more complicated, though without endangering the effectiveness of the decoding algorithm owned by the code designer.

Note that this matrix Q looks like a general scrambling matrix, exactly as S and M are. Thus, compared to the original cryptosystems, this modified version involving such a matrix instead of a simple permutation matrix has the advantage of protecting more powerfully the secret generator G (resp. the parity-check H). Though, thanks to the particular structure of Q, for the legal user in the decoding phase it will be very easy to cancel the action of this matrix and hence retrieve the original message. More precisely, the authors propose to verify that a randomly chosen error vector $\mathbf{e}$ respects some determined set of constraints, say $\mathcal{K}$, so that in the decoding process the effect of R disappears; consequently, the matrix Q is replaced by T, allowing the application of the secret algorithm for the code in use, just like in the cryptosystems proposed by McEliece and Niederreiter. In the description of the modified algorithms we leave the maximum number of intentional errors outside of the set $\mathcal{K}$. Note that the code designer has to take into account the effect that the matrix T has on the plaintext, so, in order to implement syndrome decoding, for $\mathbf{e}$ it must also be true that $\mathrm{wt}(\mathbf{e}) \leqslant \frac{t}{v}$.

Algorithm 5 – Variant of the McEliece cryptosystem using a matrix Q [2]

Keys generation:

- ▸ **Produce** a scrambling matrix S as in Alg. 2a, a systematic matrix G generating some linear code (so not necessarily a Goppa code) able of correcting t errors and a matrix Q as defined above.
- ⋄ **Public key:** $(\mathsf{G}', \frac{t}{v}, \mathcal{K})$, where G' is the $k \times n$ matrix SGQ^{-1} and v is the number of matrices composing T.

Encryption:
Let $\mathbf{x} \in \mathbb{F}_q^k$ be a plaintext and $\mathbf{e} \in \mathcal{W}_{n,tv^{-1}}$ a random error vector in $\mathbb{F}_q^n$ satisfying the constraints $\mathcal{K}$. Then, the ciphertext is given by

$$\mathbf{y} = \mathbf{x}\mathsf{G}' + \mathbf{e}.$$

Decryption:
Having inserted the inverse of Q in the public key permits Bob to exploit immediately the characteristic of $\mathbf{e}$:

$$\tilde{\mathbf{y}} := \mathbf{y}\mathsf{Q} = \mathbf{x}\mathsf{SG} + \mathbf{e}\mathsf{Q} = \mathbf{x}\mathsf{SG} + \mathbf{e}\mathsf{T},$$

since $\mathbf{e}$ has been so chosen that it is possible to retrieve $\mathbf{e}\mathsf{T}$ from $\mathbf{e}\mathsf{Q}$. Hence, we go back to the original McEliece cryptosystem, where we have that

$$d_{\mathrm{H}}(\tilde{\mathbf{y}}, \mathbf{x}\mathsf{SG}) = \mathrm{wt}(\mathbf{e}\mathsf{T}) \leqslant t,$$

from which Bob obtains the original message by applying the decoding algorithm.

Algorithm 6 – Variant of the Niederreiter cryptosystem using a matrix Q [2]

Keys generation:

- **Produce** a scrambling matrix M as in Alg. 3a, a systematic parity-check matrix H of some linear code able of correcting t errors and a matrix Q as defined above.
- **Public key:** $(\mathsf{H}', \frac{t}{v}, \mathcal{K})$, where H' is the $r \times n$ matrix $\mathsf{MHQ}^\top$.

Encryption:
Alice takes a plaintext and performs constant weight encoding on it, getting the string $\mathbf{e} \in \mathbb{F}_q^n$ of the desired weight. The ciphertext is

$$\mathbf{y}^\top = \mathsf{H}'\mathbf{e}^\top.$$

Decryption:
As before, we exploit the particular form of Q:

$$\tilde{\mathbf{y}}^\top := \mathsf{M}^{-1}\mathbf{y}^\top = \mathsf{HQ}^\top\mathbf{e}^\top = \mathsf{H}(\mathbf{e}\mathsf{Q})^\top = \mathsf{H}(\mathbf{e}\mathsf{T})^\top = \mathsf{HT}^\top\mathbf{e}^\top.$$

The private decoding algorithm returns $\mathsf{T}^\top\mathbf{e}^\top$ and hence $\mathbf{e}$, to which Bob has to apply the inverse of the used constant weight algorithm.

5.2 Respecting public constraints

We now see an example of constraints, as described in [2]. Let $w = 1$, that is $\mathsf{R} = \mathsf{a}^\top\mathsf{b}$ for some $z \times n$ matrices a and b. When choosing the error vector $\mathbf{e}$, assume we want Alice to verify that

$$\mathsf{a}\mathbf{e}^\top = \mathbf{0}^\top, \tag{5.2}$$

where $\mathbf{0}$ is the all-zero vector. In this case, for the set of constraints we have $\#\mathcal{K} = \mathrm{rank}(\mathsf{a})$, since we consider every row of a as a different condition. Then, during the decryption phase Bob computes

$$\mathbf{e}\mathsf{R} = \mathbf{e}\mathsf{a}^\top\mathsf{b} = (\mathsf{a}\mathbf{e}^\top)^\top\mathsf{b} \overset{(5.2)}{=} \mathbf{0},$$

which immediately returns $\mathbf{e}\mathsf{T}$ from $\mathbf{e}\mathsf{Q}$.

Sadly, the utility of having found a form for Q with strong scrambling properties but easy to remove could be compromised when using the Niederreiter version of this cryptosystem together with R as here. In fact, for any vector $\mathbf{c}$ satisfying (5.2) it is

$$\mathsf{H}'\mathbf{c}^\top = \mathsf{MHT}^\top\mathbf{c}^\top.$$

So, with the further assumption that $\mathsf{T} = \mathsf{p}_1$ is a simple permutation matrix, the encodable messages determined by a together with the code generated by H' actually define a *public* subcode of H' equal to

$$\ker\left(\frac{\mathsf{H}'}{\mathsf{a}}\right) = \ker\left(\frac{\mathsf{MHT}^\top}{\mathsf{a}}\right)$$

and whose security is brought back to the one of the original Niederreiter cryptosystem.

To prevent this attack, it would be useful to find a way allowing Alice to encode her messages without Bob being forced to publish the set $\mathcal{K}$, in our case the matrix a, but only

the highest weight allowed for $\mathbf{e}$, as in the original cryptosystems. The authors of [2] suggested this possibility too, but this needs a step more in the decryption phase, as the code designer is not able to directly remove the action of R. Assume Bob constructed his cipher with ALG. 6 and $z = 1$, that is taking a as a vector for which it is $\mathsf{a}\mathbf{e}^\top =: \gamma \in \mathbb{F}_q$. Then, when receiving a ciphertext, Bob computes

$$\tilde{\mathbf{y}}^\top = \mathsf{H}(\mathbf{e}\mathsf{Q})^\top = \mathsf{H}(\mathbf{e}\mathsf{R})^\top + \mathsf{H}(\mathbf{e}\mathsf{T})^\top = \gamma\mathsf{H}\mathsf{b}^\top + \mathsf{H}\mathsf{T}^\top\mathbf{e}^\top, \tag{5.3}$$

since with the help of the particular form of R we have

$$(\mathbf{e}\mathsf{R})^\top = (\mathbf{e}\mathsf{a}^\top\mathsf{b})^\top = \mathsf{b}^\top\mathsf{a}\mathbf{e}^\top = \gamma\mathsf{b}^\top.$$

Before applying his secret decoding algorithm, Bob only has to eliminate the vector $\gamma\mathsf{H}\mathsf{b}^\top$ from (5.3). For this, he considers

$$\tilde{\mathbf{y}}' := (\gamma - \gamma_B)\mathsf{H}\mathsf{b}^\top + \mathsf{H}\mathsf{T}^\top\mathbf{e}^\top$$

and tries to retrieve the value of γ just by hit and try, expecting a success after about $\frac{q}{2}$ attempts.

5.3 Respecting private constraints

The best way to avoid this attack would be to keep the matrix a secret. Thus, instead of delegating the task of finding some suitable error vector to the sender, the code designer could make available as a part of the public key not the set of constraints $\mathcal{K}$, but directly the set $\mathcal{S}$ of vectors respecting them and not overtaking a given weight, so that the whole matrix R and hence Q need not be published. To reduce the risk of retrieving $\mathcal{K}$ starting from $\mathcal{S}$, one could publish only a subset (e.g. a subspace) of $\mathcal{S}$, with the drawback of needing an increased size for this set. This method employing an error vector randomly chosen among the public elements of $\mathcal{S}$ is clearly applicable only to the McEliece version of the cryptosystem, since the ciphertext can be transmitted with any of them. For the Niederreiter version, a solution could be a particular algorithm uniquely transforming any plaintext into an encodable vector.

From [2] we know that if a code over the binaries has to be defined, it is not safe to choose Q as sum of a rank-1 matrix R and of a simple permutation matrix $\mathsf{T} = \mathsf{p}_1$ even if any information about R is kept secret. The choice of an R with higher rank could be a solution, as well as the construction of T starting from many permutation matrices; however, increasing the number of linearly independent rows in R would have as a consequence a smaller amount of error vectors available for encoding, while a denser T requires error vectors of lower weight, or equivalently codes able of correcting a higher number of errors.

The next goal is hence to find, for a private code of length n, a matrix R of not too low rank together with a set $\mathcal{S}$ containing as many error vectors as possible, though not overtaking the given weight. A low-rank matrix T is still preferable. Consider a valid matrix R of rank ρ. Select ρ independent rows of $\mathsf{R}^\top$ to construct a $\rho \times n$ matrix H_R, which will be treated as a parity-check matrix of some code. Build a corresponding generator G_R; then, $\mathcal{S} \subseteq \langle \mathsf{G}_\mathsf{R} \rangle = \ker(\mathsf{R}^\top)$.

In the following we will work over the binaries, trying to find the set $\mathcal{S}$ as a subspace of $\mathbb{F}_2^n$. We have searched for the elements of this set with the software `Magma`, choosing H_R as the one-line matrix $\left(\mathbf{x} \mid 1\right)$ and $\mathsf{G}_\mathsf{R} = \left(\mathsf{I}_{n-1} \mid \mathbf{x}^\top\right)$, where $\mathbf{x} := (10\ldots0) \in \mathbb{F}_2^{n-1}$. Even if it is $\rho = 1$,

which is not safe in practice [2], we use this value for ρ since it allows to find big subspaces of $\mathbb{F}_2^n$, approaching an upper bound for the cardinality of $\mathcal{S}$ in function of n. The number of words of these codes is clearly 2^{n-1}; the weight distributions for small n are given in TAB. 8 at p. 80.

The new variant, like many others, has been proposed as a response to the Sidel'nikov-Šestakov attack, aiming at reinserting RS codes into the list of safe codes for cryptographic purposes. In fact, RS codes reach the Singleton bound, making them attractive from this point of view. In practice, RS codes with low redundancy are used, since among other things this is a synonym for a higher rate. So, we can assume r to be smaller than half the code length. Together with this fact, the Singleton bound yields

$$d + k = n + 1 \Longrightarrow 2t + 1 = n - k + 1 \Longrightarrow t = \frac{r}{2} < \frac{n}{4}.$$

Hence, among the values for t we found, only the ones smaller than $\frac{n}{4}$ are relevant for the cryptosystem. These are the ones lying above the broken line in TAB. 8 and their total numbers for each n are reported in the second last row. We denote these numbers as $\Sigma(n)$. Sadly, a better analysis comprising also larger numbers was not possible because of implementation difficulties due to the slowness of the employed software, which took for example about two hours to return the row space of the 20×21 generator matrix and seven hours for the space with 21 dimensions.

Consider now the $1 \times n$ all-one parity-check matrix $\tilde{\mathsf{H}}_{\mathsf{R}} := \begin{pmatrix} \mathbf{1} \end{pmatrix}$ and let $\tilde{\Sigma}$ be the equivalent of Σ applied to $\tilde{\mathsf{H}}_{\mathsf{R}}$. It must be pointed out that for some values of n, this matrix produces codes of the same size (2^{n-1}) but with more vectors of small weight than H_{R} does. Though, all its vectors have even weight, so, imagining TAB. 8 to be based on $\tilde{\mathsf{H}}_{\mathsf{R}}$, the "jump" of the broken line from an even to an odd weight has no particular influence on the function $\tilde{\Sigma}$. As for the case of $\Sigma(n)$, we are not able to calculate the images of $\tilde{\Sigma}(n)$ for too large n. The running times for the latter are even longer: up to 16 hours for $\tilde{\Sigma}(22)$. The consequence is that neither $\tilde{\mathsf{H}}_{\mathsf{R}}$ can be considered for the computations we are about to execute. We have anyway inserted the images of $\tilde{\Sigma}$ in the bottom line of TAB. 8. A good second choice could be the subspace with $\begin{pmatrix} \mathbf{x} + \mathbf{1} \mid 1 \end{pmatrix} = \begin{pmatrix} 01\ldots1 \mid 1 \end{pmatrix}$ as a parity-check matrix, which for some consecutive values of n has the same values of $\Sigma(n)$ as H_{R} but a different weight distribution.

In order to get at least an approximation of $\Sigma(n)$ for much larger values of the variable, we can resort to exponential regression. The data considered for computing the function are the ones highlighted with a dark-gray background in the $\Sigma(n)$-row of TAB. 8, which are equal to the sum of the above cells having a lighter background. Let S be the set grouping those five numbers, which are the images of Σ for $n = 1 \mod 4$. With the help of `OpenOffice.org` we get that the function

$$\sigma(n) \cong 0{,}2738e^{0{,}5259n},$$

represents a good solution, at least for the data in our possession, having a coefficient of determination R^2 overtaking 0,9998 (see FIG. 1).

Exploiting the definition of the set S, we can say that for the code for which $\tilde{\mathsf{H}}_{\mathsf{R}}$ is a parity-check matrix, the set S would contain only the images of $\tilde{\Sigma}(n)$ with $n = 1 \mod 8$ (see also TAB. 8).

It is interesting to compare the function σ with the ones obtained by using only the lowest two, three or four of the values of the set S: we notice that the extrapolated approximations of $\Sigma(n)$ for the higher supporting points n are a bit underestimated; for example, for $n = 17$

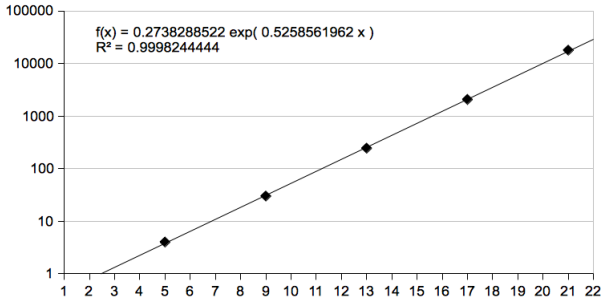


Figure 1 – The function $\Sigma(n)$ for $n \leqslant 22$ (small squares) and the approximating function $\sigma(n)$ (line).

these tree results are equal to 12656, 14684 and 16104 respectively while it is $\Sigma(17) = 17824$. In other words, we are most probably giving lower bounds for the actual values of $\Sigma(n)$ when $n > 17$.

We can now introduce the set $\mathcal{S}$ mentioned at the beginning of this section. This is the set of vectors which can be used as error vectors given some n. The code designer chooses the desired size of $\mathcal{S}$, say s, which she wants to be large and from that she computes the least code length n for which it holds $\Sigma(n) \geqslant s$. Assuming that for reasons of time or of storing capacities she has not all values of $\Sigma(n)$ available, she inverts the extrapolating function. In the case of the data in our possession, it is

$$\sigma^{-1}(s) \cong 1{,}902 \ln(s) + 2{,}463.$$

A similar approach can be carried out with only the number of vectors having the highest weight for given n. In this case, we get

$$\widetilde{\sigma}(n) \cong 0{,}2082 e^{0{,}5222n} \quad \text{and} \quad \widetilde{\sigma}^{-1}(s) \cong 1{,}915 \ln(s) + 3{,}006.$$

We have seen that the function $\sigma(n)$ is believed to return lower values than $\Sigma(n)$, so it is probable that the closest integer to $\sigma^{-1}(s)$ for given s is a sufficient value to produce a set $\mathcal{S}$ of the wanted size. Some examples for values of σ^{-1} are given in Tab. 7. The images of $\widetilde{\sigma}^{-1}$ are very similar: for example, values of s equal to 2^{128} and 2^{512} return 172,914 and 682,64 respectively. To design a code with such a size for $\mathcal{S}$, a field of size $2^m \geqslant \sigma^{-1}(s)$ for some $m \in \mathbb{N}$ is needed.

s	2^8	17200	2^{50}	2^{128}	2^{256}	2^{512}
$\sigma^{-1}(s)$	13,008	21,009	68,4	171,2	339,9	677,3
	$\Sigma(13) = 244 < 256$	$\Sigma(21) = 17284 > 17200$				
	$\Sigma(14) = 312 > 256$					

Table 7 – Some images of the function σ^{-1}. By comparing the preimage of 13,008 with $\Sigma(13)$ and the one of 21,009 with $\Sigma(21)$, we see that in the former case the requested cardinality of $\mathcal{S}$ is not reached, while in the latter $n = 21$ is sufficient.

5.4 An attack to the new variants

Very recently, a group formed by the French mathematicians Gauthier, Otmani and Tillich discovered an attack to the cryptosystems proposed at the beginning of this chapter [17]. We resume it after a brief recall of relevant arguments.

Recall. The following concepts have been described at the end of Sect. 1.3. A GRS code of length n and dimension k is defined as the set

$$\mathsf{GRS}_k(\mathbf{a};\mathbf{v}) = \left\{ \big(v_1 f(a_1), \ldots, v_n f(a_n)\big) \;\middle|\; f \in (\mathbb{F}_{q^m})_{<k}[x] \right\},$$

where $\mathbf{a} \in \mathbb{F}_q^n$ is composed of n distinct coordinates and $\mathbf{v}$ is some vector over $\mathbb{F}_q^*$. A generator matrix is given by $\mathsf{G}(\mathbf{a};\mathbf{v}) = \big(v_j a_j^{i-1}\big)_{\substack{i=1,\ldots,k,\\ j=1,\ldots,n}}$; a parity-check matrix for GRS codes has been given in equation (4.3).There, $\mathbf{z}$ is some vector belonging to the same set of $\mathbf{v}$ and closely related to it. In fact, since the parity-check matrix of a code is the generator of its dual, it holds that $\mathsf{GRS}_k(\mathbf{a};\mathbf{v})^\perp = \mathsf{GRS}_r(\mathbf{a};\mathbf{z})$. Using the same notation, a parity-check matrix for RS codes can be obtained by setting $\mathbf{v} = \mathbf{z} = \mathbf{1}$, i.e. $\mathsf{Y} = \mathsf{I}_n$. In Def. 1.5.2 we defined the vector multiplication (star product) of two vectors $\mathbf{a}$ and $\mathbf{b}$ having same length as

$$\mathbf{a} \star \mathbf{b} = (a_1 b_1, \ldots, a_\nu b_\nu).$$

Definition 5.4.1 (Star product code [17]). Let $\mathcal{C}$ and $\mathcal{D}$ be two codes of equal length. Then:

i. The code
$$\mathcal{C} \star \mathcal{D} := \langle \{ \mathbf{c} \star \mathbf{d} \mid \mathbf{c} \in \mathcal{C},\ \mathbf{d} \in \mathcal{D} \} \rangle$$
is called *star product code* of $\mathcal{C}$ and $\mathcal{D}$;

ii. The code
$$\mathcal{C}^{\star 2} := \mathcal{C} \star \mathcal{C}$$
is called *square code of* $\mathcal{C}$.

The attack described in [17] is presented for the McEliece version of the cryptosystem, but it is clearly valid for the Niederreiter cipher too. Let G be the generator matrix of some q-ary $[n,k,d]$ GRS code $\langle \mathsf{G} \rangle$ with $k < \frac{n}{2} - 1$, assume R has rank 1, choose $\mathsf{T} := \mathsf{p}$ to be a permutation matrix (i.e. with the notation of equation (5.1) it is $v = 1$) and define a second GRS code as the one generated by the matrix $\tilde{\mathsf{G}} := \mathsf{G}\mathsf{p}^{-1}$; as usual, $\langle \mathsf{G}' \rangle$ is the public code. Even if the cryptanalyst tries to construct the code $\langle \tilde{\mathsf{G}} \rangle$ through the identification of a subcode $\mathcal{G} \subseteq \langle \tilde{\mathsf{G}} \rangle \cap \langle \mathsf{G}' \rangle$ of dimension $k-1$, this has to be considered as a decoding attack because the procedure aims at recovering a single plaintext at a time.

Description of the subcode $\mathcal{G}$

It is clear that $\mathrm{rank}(\mathsf{R}) = \mathrm{rank}(\mathsf{R}\mathsf{p}^{-1})$. Let $\mathsf{R}\mathsf{p}^{-1} = \mathbf{Y}^\top \mathbf{X}$ for some vectors $\mathbf{X}$ and $\mathbf{Y}$ in $\mathbb{F}_q^n$. We also define a matrix F through

$$\mathsf{F} := \mathsf{Q}\mathsf{p}^{-1} = (\mathsf{R} + \mathsf{p})\mathsf{p}^{-1} = \mathsf{R}\mathsf{p}^{-1} + \mathsf{I}_n = \mathbf{Y}^\top \mathbf{X} + \mathsf{I}_n. \tag{5.4}$$

F is invertible since by definition both Q and p are. Furthermore, it can be easily verified that

$$\mathsf{F}^{-1} = \mathsf{I}_n + \boldsymbol{\lambda}^\top \mathbf{X}, \tag{5.5}$$

where $\boldsymbol{\lambda}^\top := \frac{-1}{\mathbf{X}\cdot\mathbf{Y}+1}\mathbf{Y}^\top$ and the operation in the denominator of the fraction is the usual dot product. By denoting the code spanned by $\boldsymbol{\lambda}$ as $\langle\boldsymbol{\lambda}\rangle := \{s\boldsymbol{\lambda} \mid s \in \mathbb{F}_q\} \subseteq \mathbb{F}_q^n$, it clearly holds that $\langle\boldsymbol{\lambda}\rangle = \langle\mathbf{Y}\rangle$. The consistency of this expression is verified by the following

Proposition 5.4.2. *For* $\mathbf{X}$ *and* $\mathbf{Y}$ *as above, it holds* $\mathbf{X}\cdot\mathbf{Y} = \mathbf{X}\mathbf{Y}^\top \neq -1$.

Proof. Assume $\mathbf{X}\cdot\mathbf{Y} = -1$. Then it is

$$\big(\mathbf{Y}^\top\mathbf{X}\big)\mathbf{Y}^\top = \mathbf{Y}^\top\big(\mathbf{X}\mathbf{Y}^\top\big) = -\mathbf{Y}^\top,$$

that is $\mathbf{Y}^\top$ is an eigenvector of $\mathbf{Y}^\top\mathbf{X}$ with eigenvalue -1. In general, some element $\mu \in \mathbb{F}_q$ is an eigenvalue of $\mathbf{Y}^\top\mathbf{X}$ if and only if

$$\det\big(\mathbf{Y}^\top\mathbf{X} - \mu\mathsf{I}_n\big) = 0;$$

then we have that $\det\big(\mathbf{Y}^\top\mathbf{X} - (-1)\mathsf{I}_n\big) = 0$, thus the matrix $\mathbf{Y}^\top\mathbf{X} + \mathsf{I}_n$ is singular. $\lightning$ $\square$

We can now describe the elements of the code $\mathcal{G}$:

Lemma 5.4.3 ([17]).

i. If $\boldsymbol{\lambda} \notin \langle\tilde{\mathsf{G}}\rangle^\perp$*, then* $\forall\mathbf{c} \in \langle\mathsf{G}'\rangle$ *there is a* $\mathbf{d} \in \langle\tilde{\mathsf{G}}\rangle$ *such that*

$$\mathbf{c} = \mathbf{d} + (\boldsymbol{\lambda}\cdot\mathbf{d})\mathbf{X}; \tag{5.6}$$

ii. Similarly, for any $\mathbf{c}^\perp \in \langle\mathsf{G}'\rangle^\perp$ *there exists a* $\mathbf{d}^\perp \in \langle\tilde{\mathsf{G}}\rangle^\perp$ *such that*

$$\mathbf{c}^\perp = \mathbf{d}^\perp + (\mathbf{X}\cdot\mathbf{d}^\perp)\mathbf{Y}. \tag{5.7}$$

Proof. (i) Alg. 5 defines $\langle\mathsf{G}'\rangle = \langle\mathsf{G}\rangle\mathsf{Q}^{-1}$. Then we have:

$$\langle\tilde{\mathsf{G}}\rangle = \langle\mathsf{G}\rangle\mathsf{p}^{-1} = \langle\mathsf{G}'\rangle\mathsf{Q}\mathsf{p}^{-1} \overset{(5.4)}{=} \langle\mathsf{G}'\rangle\mathsf{F} \quad\Longrightarrow\quad \langle\mathsf{G}'\rangle = \langle\tilde{\mathsf{G}}\rangle\mathsf{F}^{-1}. \tag{5.8}$$

So, if one picks any $\mathbf{c} \in \langle\mathsf{G}'\rangle$, by (5.5) there exists a $\mathbf{d} \in \langle\tilde{\mathsf{G}}\rangle$ satisfying (5.6).

(ii) Let $\mathbf{c}$ and $\mathbf{d}$ be as before. We conjecture that $\langle\mathsf{G}'\rangle^\perp = \langle\tilde{\mathsf{G}}\rangle^\perp\mathsf{F}^\top$. For convenience, we use here the transposed notation to indicate the dot product.

$$0 = \mathbf{c}\cdot\mathbf{c}^\perp = \mathbf{c}(\mathbf{c}^\perp)^\top \overset{(5.8)}{=} \big(\mathbf{d}\mathsf{F}^{-1}\big)(\mathbf{c}^\perp)^\top \overset{!}{=} \big(\mathbf{d}\mathsf{F}^{-1}\big)\big(\mathbf{d}^\perp\mathsf{F}^\top\big)^\top = \mathbf{d}\mathsf{F}^{-1}\mathsf{F}(\mathbf{d}^\perp)^\top = 0.$$

The assumption follows from $\mathbf{c}^\perp = \mathbf{d}^\perp\mathsf{F}^\top$. $\square$

Remark 5.4.4. In Lemma 5.4.3(i), the condition that $\boldsymbol{\lambda}$ is not an element of the dual code of $\langle\tilde{\mathsf{G}}\rangle$ is needed for not having $\boldsymbol{\lambda}\cdot\mathbf{d} = 0$ $\forall\mathbf{d}$. A consequence would be that in (5.6) we have $\mathbf{c} = \mathbf{d}$, i.e. $\langle\mathsf{G}'\rangle = \langle\tilde{\mathsf{G}}\rangle = \langle\mathsf{G}\rangle$, reducing to the case in which the Sidel'nikov-Šestakov attack [41] can be applied.

For now, only part (i) of LEMMA 5.4.3 is necessary; we will go back to part (ii) in a while. Using the results just obtained, the code that the authors aim to find is

$$\mathcal{G} := \left\{ \mathbf{d} \in \langle \tilde{\mathsf{G}} \rangle \;\middle|\; \boldsymbol{\lambda} \cdot \mathbf{d} = 0 \right\} = \langle \tilde{\mathsf{G}} \rangle \cap \langle \boldsymbol{\lambda} \rangle^{\perp}. \tag{5.9}$$

Lemma 5.4.5 ([17]). *The code $\mathcal{G}$ has dimension $k-1$ and it is $\mathcal{G} = \langle \tilde{\mathsf{G}} \rangle \cap \langle \mathsf{G}' \rangle$.*

Proof. Per definition, $\dim(\langle \tilde{\mathsf{G}} \rangle) = k$. So, from $\boldsymbol{\lambda} \notin \langle \tilde{\mathsf{G}} \rangle^{\perp}$, $\dim(\langle \boldsymbol{\lambda} \rangle) = 1$ and $\mathcal{G} \subseteq \langle \tilde{\mathsf{G}} \rangle$ we have $\dim(\mathcal{G}) = k-1$. Concerning the structure of $\mathcal{G}$: Let $\mathbf{d} \in \mathcal{G} \subseteq \langle \tilde{\mathsf{G}} \rangle$ and $\mathbf{c} \in \langle \mathsf{G}' \rangle$. By (5.6), $\langle \mathsf{G}' \rangle \ni \mathbf{c} = \mathbf{d} \implies \mathcal{G} \subseteq \langle \mathsf{G}' \rangle$. From (5.8) we know that $\langle \tilde{\mathsf{G}} \rangle \neq \langle \mathsf{G}' \rangle$, implying that $\dim(\mathcal{G}) \leqslant \dim(\langle \tilde{\mathsf{G}} \rangle \cap \langle \mathsf{G}' \rangle) < k$. □

Proposition 5.4.6 ([17, 49]).

i. Given two codes $\mathcal{C}$ and $\mathcal{D}$ of same length, it is $\dim(\mathcal{C} \star \mathcal{D}) \leqslant \dim \mathcal{C} \cdot \dim \mathcal{D}$;

ii. If $\mathcal{C} = \mathsf{GRS}_k(\mathbf{a}; \mathbf{v})$ for some $\mathbf{a}$ and $\mathbf{v}$, $\mathcal{C}^{\star 2} \subseteq \mathsf{GRS}_{2k-1}(\mathbf{a}; \mathbf{v}^{\star 2})$;

iii. $\langle \mathsf{G}' \rangle^{\star 2} \subseteq \langle \tilde{\mathsf{G}} \rangle^{\star 2} + \langle \tilde{\mathsf{G}} \star \mathbf{X} \rangle + \langle \mathbf{X} \rangle^{\star 2}$;

iv. $\dim\big(\langle \mathsf{G}' \rangle^{\star 2}\big) \leqslant 3k-1$.

Proof. (i) The star product of the basis vectors of $\mathcal{C}$ and $\mathcal{D}$ generates the new code.

(ii) By DEF. 1.3.8, the star product of two elements of a GRS_k code cannot return vectors of polynomials of degree exceeding $2k-2$.

(iii) Follows from (5.6) by vector multiplication of any two elements of the public code. Let $\mathbf{c}, \mathbf{c}' \in \langle \mathsf{G}' \rangle$ and let $\mathbf{d}, \mathbf{d}'$ be the corresponding elements in $\langle \tilde{\mathsf{G}} \rangle$ (see (5.6)); we get:

$$\mathbf{c} \star \mathbf{c}' = \underbrace{\mathbf{d} \star \mathbf{d}'}_{\in \langle \tilde{\mathsf{G}} \rangle^{\star 2}} + \underbrace{\big((\boldsymbol{\lambda} \cdot \mathbf{d}')\mathbf{d} + (\boldsymbol{\lambda} \cdot \mathbf{d})\mathbf{d}'\big) \star \mathbf{X}}_{\in \langle \tilde{\mathsf{G}} \star \mathbf{X} \rangle} + \underbrace{(\boldsymbol{\lambda} \cdot \mathbf{d})(\boldsymbol{\lambda} \cdot \mathbf{d}')\mathbf{X}^{\star 2}}_{\in \langle \mathbf{X} \rangle^{\star 2}}. \tag{5.10}$$

(iv) By (i)-(iii):

$$\dim\big(\langle \mathsf{G}' \rangle^{\star 2}\big) \leqslant \dim\big(\langle \tilde{\mathsf{G}} \rangle^{\star 2}\big) + \dim(\langle \tilde{\mathsf{G}} \star \mathbf{X} \rangle) + \dim\big(\langle \mathbf{X} \rangle^{\star 2}\big) \leqslant (2k-1) + k + 1 = 3k.$$

This number can be reduced to $3k-1$ by noticing that $\langle \mathsf{G}' \rangle^{\star 2}$ is actually a subcode of $\langle \tilde{\mathsf{G}} \rangle^{\star 2} + \mathcal{T} \star \mathbf{X}$ for a code $\mathcal{T}$ of dimension at most k. For this we rewrite the generator matrix of a GRS code as

$$\mathsf{G}(\mathbf{a}; \mathbf{v}) = \big(v_j a_j^{i-1}\big)_{\substack{i=1,\dots,k,\\ j=1,\dots,n}} = \big(\mathbf{v} \star \mathbf{a}^{\star(i-1)}\big)_{i=1,\dots,k}.$$

We define this new code as $\mathcal{T} := \{\mathbf{t}_{i,j} \mid i, j = 1, \dots, k\}$, where the $\mathbf{t}_{i,j}$ are defined as

$$\begin{aligned} &\big(\boldsymbol{\lambda} \cdot (\mathbf{v} \star \mathbf{a}^{\star(i-1)})\big)(\mathbf{v} \star \mathbf{a}^{\star(j-1)}) + \big(\boldsymbol{\lambda} \cdot (\mathbf{v} \star \mathbf{a}^{\star(j-1)})\big)(\mathbf{v} \star \mathbf{a}^{\star(i-1)}) + \\ &\quad + \big(\boldsymbol{\lambda} \cdot (\mathbf{v} \star \mathbf{a}^{\star(i-1)})\big)\big(\boldsymbol{\lambda} \cdot (\mathbf{v} \star \mathbf{a}^{\star(j-1)})\big) \star \mathbf{X}. \end{aligned} \tag{5.11}$$

It can be easily shown that $\mathcal{T}$ is generated by some vectors $\mathbf{u}_i := \mathbf{t}_{i,i_0}$, where i_0 is a fixed index such that the scalar product $\boldsymbol{\lambda} \cdot (\mathbf{v} \star \mathbf{a}^{\star(i_0-1)}) \neq 0$. The existence of such an i_0 is guaranteed by the fact that $\boldsymbol{\lambda} \notin \langle \tilde{\mathsf{G}} \rangle^{\perp}$. This means that at most k codewords of the form of $\mathbf{u}_i$ are needed to span $\mathcal{T}$. We see that the first two summands of (5.11) are a word of $\langle \tilde{\mathsf{G}} \rangle$, while the rightmost one is in $\langle \mathbf{X} \rangle$; this becomes more evident by comparing $\mathbf{d} := \mathbf{v} \star \mathbf{a}^{\star(i-1)}$ and $\mathbf{d}' := \mathbf{v} \star \mathbf{a}^{\star(j-1)}$ with (5.10). Hence, $\langle \mathsf{G}' \rangle \subseteq \langle \tilde{\mathsf{G}} \rangle^{\star 2} + \mathcal{T} \star \mathbf{X}$, implying that $\dim\big(\langle \tilde{\mathsf{G}} \rangle^{\star 2}\big) \leqslant (2k-1) + k = 3k-1$. □

Remark 5.4.7. In part (iv), notice the small improvement from $3k$ to $3k-1$ in the computations: it could look little thing, but, apart from special cases, the dimension of $\langle \mathsf{G}' \rangle^{\star 2}$ seems to be exactly $3k-1$ [17].

Finding a basis for $\mathcal{G}$

The next step is to find the structure of $\mathcal{G}$ by looking for a basis. We proceed as follows: let $\mathbf{g}'_1, \ldots, \mathbf{g}'_k$ be a basis of $\langle \mathsf{G}' \rangle$; without loss of generality, let these vectors be the rows of G'. Since $\mathcal{G}$ is a subset of $\langle \mathsf{G}' \rangle$, the basis vectors of $\mathcal{G}$ can be selected among the words of the public code. To recognize whether a vector belongs to that basis, we exploit some new codes generated starting from the known ones.

Proposition 5.4.8 ([17]). *If some random words $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \langle \mathsf{G}' \rangle$ are chosen, the resulting star product code $\mathcal{G}' := \langle \mathsf{G}' \rangle \star \langle \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \rangle$ has dimension at most $3k-3$ ($3k-1$ if there are at least four vectors $\mathbf{w}_i$). If instead the three $\mathbf{w}_i$ are taken from the subcode $\mathcal{G}$, $\dim(\mathcal{G}')$ is only $\leqslant 2k+2$.*

Proof. Assume the $\mathbf{w}_i$ can be freely chosen among all elements of $\langle \mathsf{G}' \rangle$; then, they have the form (say) $\mathbf{w}_i = \mathbf{w}'_i \mathsf{G}' = \sum_{j=1}^{k} w'_{i,j} \mathbf{g}'_j$. By Prop. 5.4.6(i), a basis for $\mathcal{G}'$ can be found among the $3k$ elements $\{\mathbf{w}_1 \star \mathbf{g}'_1, \ldots, \mathbf{w}_3 \star \mathbf{g}'_k\}$. We show that in this set there are at least three linearly dependent vectors. For $\ell, s \in \{1,2,3\}$, $\ell \neq s$, we get

$$\sum_{j=1}^{k} \mathbf{w}_\ell \star (w'_{s,j} \mathbf{g}'_j) = \mathbf{w}_\ell \star \sum_{j=1}^{k} w'_{s,j} \mathbf{g}'_j = \mathbf{w}_\ell \star \mathbf{w}_s = \mathbf{w}_s \star \sum_{j=1}^{k} w'_{\ell,j} \mathbf{g}'_j = \sum_{j=1}^{k} \mathbf{w}_s \star (w'_{\ell,j} \mathbf{g}'_j).$$

Suppose now that $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \mathcal{G} \subseteq \langle \tilde{\mathsf{G}} \rangle$. As elements of $\langle \mathsf{G}' \rangle$, by (5.6) for all $j \in \{1, \ldots, k\}$ there exist some $\mathbf{d}'_j \in \langle \tilde{\mathsf{G}} \rangle$ such that $\mathbf{g}'_j = \mathbf{d}'_j + (\boldsymbol{\lambda} \cdot \mathbf{d}'_j)\mathbf{X}$. Thus:

$$\mathbf{w}_i \star \mathbf{g}'_j = \underbrace{\mathbf{w}_i \star \mathbf{d}'_j}_{\in \langle \tilde{\mathsf{G}} \rangle^{\star 2}} + \underbrace{(\boldsymbol{\lambda} \cdot \mathbf{d}'_j)(\mathbf{w}_i \star \mathbf{X})}_{\in \langle \mathbf{w}_i \star \mathbf{X} \rangle} \Longrightarrow$$

$$\Longrightarrow \langle \mathbf{w}_1 \star \mathbf{g}'_1, \ldots, \mathbf{w}_3 \star \mathbf{g}'_k \rangle \subseteq \langle \tilde{\mathsf{G}} \rangle^{\star 2} + \sum_{i=1}^{3} \langle \mathbf{w}_i \star \mathbf{X} \rangle.$$

By Prop. 5.4.6(ii), the code on the right hand side has dimension at most $2k-1+3 = 2k+2$. □

This is also the reason why it is required for k to be less than $\frac{n}{2}-1$. In fact, thanks to this proposition, we have an efficient way to describe the code $\mathcal{G}$ just by knowing the public code $\langle \mathsf{G}' \rangle$. The algorithm contained in [17] requires that the eavesdropper first finds three linearly independent vectors $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \langle \mathsf{G}' \rangle$ at a stroke, then she adds a new element to them until she has collected $k-1$, i.e. $\dim(\mathcal{G})$ (see also Alg. 7 at p. 79). Thanks to Wieschebrink [49, Sect. 5], the eavesdropper is now able to recover the structure of the slightly larger code $\langle \tilde{\mathsf{G}} \rangle$ just by knowing the basis $\mathbf{w}_1, \ldots, \mathbf{w}_{k-1}$. This document presents a way to retrieve a GRS code just by knowing a $(k-\ell)$-dimensional subcode, working "even for larger ℓ". His algorithm is applicable to our case because $\ell = 1$ and the searched space $\langle \tilde{\mathsf{G}} \rangle = \langle \mathsf{G}\mathsf{p}^{-1} \rangle$ is a GRS code.

Decoding the public code

To be able to prove some equalities in the following, we first need a

Lemma 5.4.9. *Let Z be some vector space on which a symmetric bilinear form "$\cdot$" is given and let U and V be subspaces of Z. Then:*

$$U^\perp \cap V^\perp = (U+V)^\perp.$$

Proof. "$\subseteq$": Let $\mathbf{x} \in U^\perp \cap V^\perp$, $\mathbf{u} \in U$ and $\mathbf{v} \in V$; then, $\mathbf{u}+\mathbf{v} \in U+V$. We have that $\mathbf{x}\cdot(\mathbf{u}+\mathbf{v}) = \mathbf{x}\cdot\mathbf{u}+\mathbf{x}\cdot\mathbf{v} = 0$, hence $\mathbf{x} \in (U+V)^\perp$.

"$\supseteq$": Let $\mathbf{x} \in (U+V)^\perp$ and $\mathbf{u} \in U \subseteq U+V$. Then $\mathbf{x}\cdot\mathbf{u} = 0$, hence $\mathbf{x} \in U^\perp$. By repeating this with V, we get that $\mathbf{x} \in U^\perp \cap V^\perp$. □

In Lemma 5.4.5 we had already given a description of the code $\mathcal{G}$, noticing that

$$\mathcal{G} = \langle\tilde{\mathsf{G}}\rangle \cap \langle\mathsf{G}'\rangle = \{\mathbf{d} \in \langle\tilde{\mathsf{G}}\rangle \mid \boldsymbol{\lambda}\cdot\mathbf{d} = 0\}$$

has dimension $k-1$. Before going on with the last part of the attack, we first need the structure of the code $\mathcal{H} := \langle\tilde{\mathsf{G}}\rangle^\perp \cap \langle\mathsf{G}'\rangle^\perp$:

Lemma 5.4.10 ([17]). *It holds that $\mathcal{H} = \{\mathbf{d}^\perp \in \langle\tilde{\mathsf{G}}\rangle^\perp \mid \mathbf{X}\cdot\mathbf{d}^\perp = 0\}$.*

Proof. The proof is similar to the one of Lemma 5.4.5. By Lemma 5.4.9 it is

$$\begin{aligned}\dim(\mathcal{H}) &= \dim\big(\langle\tilde{\mathsf{G}}\rangle^\perp\big) + \dim\big(\langle\mathsf{G}'\rangle^\perp\big) - \dim\big(\langle\tilde{\mathsf{G}}\rangle^\perp + \langle\mathsf{G}'\rangle^\perp\big) = \\ &= r + r - \dim\big(\mathcal{G}^\perp\big) = 2r - (n-(k-1)) = 2r - r + 1 = r - 1,\end{aligned}$$

where r denotes the redundancy of the codes, which is always the same having all equal dimension.

About the structure of the code $\mathcal{H}$, this time we cannot rely o the assumption that $\mathbf{X} \notin \langle\tilde{\mathsf{G}}\rangle$ as we did in the proof of Lemma 5.4.5 with $\mathcal{G}$ and $\boldsymbol{\lambda}$. Since $\dim\big(\langle\tilde{\mathsf{G}}\rangle^\perp\big) = r$, the dimension of the set on the right hand side can be only r or $r-1$, depending on the fact that $\mathbf{X}$ is an element of $\langle\tilde{\mathsf{G}}\rangle$ or not. Let now $\mathbf{c}^\perp \in \langle\mathsf{G}'\rangle^\perp$; then, by (5.7) we have $\mathbf{c}^\perp = \mathbf{d}^\perp + (\mathbf{X}\cdot\mathbf{d}^\perp)\mathbf{Y} = \mathbf{d}^\perp \in \langle\tilde{\mathsf{G}}\rangle^\perp$ for *any* $\mathbf{d}^\perp \in \langle\tilde{\mathsf{G}}\rangle^\perp$, that is $\langle\tilde{\mathsf{G}}\rangle = \langle\mathsf{G}'\rangle$, contradicting the fact that $\dim(\mathcal{G}) = k-1$. □

We now want to find a pair $(\mathbf{X}, \boldsymbol{\lambda})$ allowing us to decode $\langle\mathsf{G}'\rangle$. At first, we need a

Definition 5.4.11 (Validity of a pair [17]). Let $\mathbf{A}, \boldsymbol{\Lambda} \in \mathbb{F}_q^n$. Then, the pair $(\mathbf{A}, \boldsymbol{\Lambda})$ is said to be a *valid* $(\mathbf{X}, \boldsymbol{\lambda})$ *pair for* $\big(\langle\mathsf{G}'\rangle, \langle\tilde{\mathsf{G}}\rangle\big)$ if it holds:

i. $\mathbf{A}\cdot\boldsymbol{\Lambda} \neq -1$;

ii. $\forall \mathbf{c} \in \langle\mathsf{G}'\rangle$ there is a $\mathbf{d} \in \langle\tilde{\mathsf{G}}\rangle$ such that $\mathbf{c} = \mathbf{d} + (\boldsymbol{\Lambda}\cdot\mathbf{d})\mathbf{A}$.

We now introduce new elements $\mathbf{X}_0 \in \mathcal{H}^\perp \setminus \langle\tilde{\mathsf{G}}\rangle$ and $\mathbf{Y}_0 \in \mathcal{G}^\perp \setminus \langle\tilde{\mathsf{G}}\rangle^\perp$ such that $\mathbf{X}_0\cdot\mathbf{Y}_0 = 0$. With these definitions it holds

Lemma 5.4.12 ([17]). *It is possible to find $\alpha, \beta \in \mathbb{F}_q^*$, $\mathbf{d}_0 \in \langle\tilde{\mathsf{G}}\rangle$ and $\tilde{\mathbf{d}}_0 \in \langle\tilde{\mathsf{G}}\rangle^\perp$ such that*

$$\mathbf{X}_0 = \mathbf{d}_0 + \alpha\mathbf{X} \quad \text{and} \quad \mathbf{Y}_0 = \tilde{\mathbf{d}}_0 + \beta\mathbf{Y}. \tag{5.12}$$

Proof. We only prove the second equality since the proof of the first is analogous. For this, one can argue about the fact that equation (5.9) can be rewritten as

$$\mathcal{G}^{\perp} = \left(\langle\tilde{\mathsf{G}}\rangle \cap \langle\boldsymbol{\lambda}\rangle^{\perp}\right)^{\perp} \overset{\substack{\text{Lemma}\\5.4.9}}{\overset{\downarrow}{=}} \langle\tilde{\mathsf{G}}\rangle^{\perp} + \langle\boldsymbol{\lambda}\rangle = \langle\tilde{\mathsf{G}}\rangle^{\perp} + \langle\mathbf{Y}\rangle. \qquad \square$$

Now define an element γ as

$$\gamma := \frac{-\mathbf{p}\cdot\mathbf{r}}{(\mathbf{X}_0\cdot\mathbf{r})(\mathbf{Y}_0\cdot\mathbf{p})}, \tag{5.13}$$

where $\mathbf{p} \in \langle\tilde{\mathsf{G}}\rangle \setminus \langle\mathsf{G}'\rangle$ and $\mathbf{r} \in \langle\mathsf{G}'\rangle^{\perp} \setminus \langle\tilde{\mathsf{G}}\rangle^{\perp}$ are arbitrary elements in the corresponding sets; moreover, for $\mathbf{r}$ it must be true that $\mathbf{X}_0\cdot\mathbf{r} \neq 0$. On the contrary, $\mathbf{Y}_0\cdot\mathbf{p}$ is always different from 0 because

$$\mathbf{Y}_0\cdot\mathbf{p} = (\tilde{\mathbf{d}}_0 + \beta\mathbf{Y})\cdot\mathbf{p} \overset{\substack{\tilde{\mathbf{d}}_0\in\langle\tilde{\mathsf{G}}\rangle^{\perp},\\ \mathbf{p}\in\langle\tilde{\mathsf{G}}\rangle}}{\overset{\downarrow}{=}} \beta(\mathbf{Y}\cdot\mathbf{p}) \overset{\langle\mathbf{Y}\rangle=\langle\boldsymbol{\lambda}\rangle}{\overset{\downarrow}{=}} \tilde{\beta}(\boldsymbol{\lambda}\cdot\mathbf{p}) \overset{\boldsymbol{\lambda}\notin\langle\tilde{\mathsf{G}}\rangle^{\perp}}{\overset{\downarrow}{\neq}} 0,$$

where $\tilde{\beta}$ is some element in $\mathbb{F}_q^*$. With these assumptions, we have that

Proposition 5.4.13 ([17]). $(\mathbf{X}_0, \gamma\mathbf{Y}_0)$ *is a valid pair for* $(\langle\mathsf{G}'\rangle, \langle\tilde{\mathsf{G}}\rangle)$.

Proof. (a) $\mathbf{X}_0\cdot(\gamma\mathbf{Y}_0) = 0$ by definition of $\mathbf{X}_0$ and $\mathbf{Y}_0$.

(b) By Def. 5.4.11 it has to be shown that $\mathbf{c}_0 := \mathbf{d} + (\gamma\mathbf{Y}_0\cdot\mathbf{d})\mathbf{X}_0 \in \langle\mathsf{G}'\rangle$ for any $\mathbf{d} \in \langle\tilde{\mathsf{G}}\rangle$. It will be proved in two parts. Let $\mathbf{d} \in \mathcal{G}$; then:

$$\begin{aligned}\gamma\mathbf{Y}_0\cdot\mathbf{d} &\overset{(5.12)}{=} \gamma(\tilde{\mathbf{d}}_0 + \beta\mathbf{Y})\cdot\mathbf{d} = \gamma\tilde{\mathbf{d}}_0\cdot\mathbf{d} + \gamma\beta\mathbf{Y}\cdot\mathbf{d} \overset{\substack{\mathbf{d}\in\langle\tilde{\mathsf{G}}\rangle,\\ \tilde{\mathbf{d}}_0\in\langle\tilde{\mathsf{G}}\rangle^{\perp}}}{\overset{\downarrow}{=}}\\ &= \gamma\beta\mathbf{Y}\cdot\mathbf{d} \overset{\langle\mathbf{Y}\rangle=\langle\boldsymbol{\lambda}\rangle}{\overset{\downarrow}{=}} \tilde{\gamma}\boldsymbol{\lambda}\cdot\mathbf{d} \overset{\mathbf{d}\in\mathcal{G}}{\overset{\downarrow}{=}} 0,\end{aligned}$$

where $\tilde{\gamma}$ is some element in $\mathbb{F}_q^*$. This implies that $\mathbf{c}_0 = \mathbf{d} \in \mathcal{G} \subseteq \langle\mathsf{G}'\rangle$.

We now have to show that this is also true for the remaining elements of $\langle\tilde{\mathsf{G}}\rangle$, that is for any $\mathbf{d} \in \langle\tilde{\mathsf{G}}\rangle \setminus \mathcal{G}$. The vector $\mathbf{p}$ introduced above belongs to this set. Thus, define $\mathbf{c}_1 := \mathbf{p} + (\gamma\mathbf{Y}_0\cdot\mathbf{p})\mathbf{X}_0$. This proof must be split into two parts too, showing that $\mathbf{c}_1$ is perpendicular to all elements of both $\mathcal{H} = \langle\tilde{\mathsf{G}}\rangle^{\perp} \cap \langle\mathsf{G}'\rangle^{\perp}$ and $\langle\mathsf{G}'\rangle^{\perp} \setminus \langle\tilde{\mathsf{G}}\rangle^{\perp}$. Let $\mathbf{q} \in \mathcal{H}$:

$$\begin{aligned}\mathbf{c}_1\cdot\mathbf{q} &= \big(\mathbf{p} + \overbrace{(\gamma\mathbf{Y}_0\cdot\mathbf{p})}^{=:P}\mathbf{X}_0\big)\cdot\mathbf{q} = \mathbf{p}\cdot\mathbf{q} + P\mathbf{X}_0\cdot\mathbf{q} \overset{\substack{\mathbf{p}\in\langle\tilde{\mathsf{G}}\rangle,\\ \mathbf{q}\in\langle\tilde{\mathsf{G}}\rangle^{\perp}}}{\overset{\downarrow}{=}}\\ &= P\mathbf{X}_0\cdot\mathbf{q} \overset{(5.12)}{=} P(\mathbf{d}_0 + \alpha\mathbf{X})\cdot\mathbf{q} \overset{\substack{\mathbf{d}_0\in\langle\tilde{\mathsf{G}}\rangle,\\ \mathbf{q}\in\langle\tilde{\mathsf{G}}\rangle^{\perp}}}{\overset{\downarrow}{=}} P\alpha\mathbf{X}\cdot\mathbf{q} \overset{\mathbf{q}\in\mathcal{H}}{\overset{\downarrow}{=}} 0.\end{aligned}$$

We have already defined an element of $\langle\mathsf{G}'\rangle^{\perp} \setminus \langle\tilde{\mathsf{G}}\rangle^{\perp}$ in (5.13), namely $\mathbf{r}$:

$$\mathbf{c}_1\cdot\mathbf{r} = \big(\mathbf{p} + (\gamma\mathbf{Y}_0\cdot\mathbf{p})\mathbf{X}_0\big)\cdot\mathbf{r} = \mathbf{p}\cdot\mathbf{r} + \gamma(\mathbf{X}_0\cdot\mathbf{r})(\mathbf{Y}_0\cdot\mathbf{p}) \overset{(5.13)}{=} 0.$$

It is now possible to describe the connection between the bases of $\langle\tilde{\mathsf{G}}\rangle$ and $\langle\mathsf{G}'\rangle$. Define

$$\begin{array}{rccl} \Phi\colon & \langle\tilde{\mathsf{G}}\rangle & \longrightarrow & \langle\mathsf{G}'\rangle \\ & \mathbf{u} & \longmapsto & \mathbf{u} + \underbrace{(\gamma\mathbf{Y}_0 \cdot \mathbf{u})}_{=:U}\mathbf{X}_0. \end{array}$$

We prove that its inverse is

$$\begin{array}{rccl} \Phi^{-1}\colon & \langle\mathsf{G}'\rangle & \longrightarrow & \langle\tilde{\mathsf{G}}\rangle \\ & \mathbf{t} & \longmapsto & \mathbf{t} + (\tilde{\boldsymbol{\lambda}} \cdot \mathbf{t})\mathbf{X}_0, \end{array}$$

where $\tilde{\boldsymbol{\lambda}} = \frac{-1}{\mathbf{X}_0\cdot(\gamma\mathbf{Y}_0)+1}\gamma\mathbf{Y}_0 = -\gamma\mathbf{Y}_0$ by definition of $\mathbf{X}_0$ and $\mathbf{Y}_0$, by computing

$$\begin{aligned} \Phi^{-1}\big(\Phi(\mathbf{u})\big) &= (\mathbf{u} + U\mathbf{X}_0) - \big(\gamma\mathbf{Y}_0 \cdot (\mathbf{u} + U\mathbf{X}_0)\big)\mathbf{X}_0 = \\ &= \mathbf{u} + U\mathbf{X}_0 - U\mathbf{X}_0 - \big((\gamma\mathbf{Y}_0) \cdot (U\mathbf{X}_0)\big)\mathbf{X}_0 = \mathbf{u} - \gamma U(\mathbf{Y}_0 \cdot \mathbf{X}_0)\mathbf{X}_0 = \mathbf{u}. \end{aligned}$$

$\Phi\big(\Phi^{-1}(\mathbf{t})\big)$ can be computed in an analogous way. This, together with the linearity of Φ, proves the bijection existing between the bases of the two codes, i.e. that for every $\mathbf{c} \in \langle\mathsf{G}'\rangle$ there is a $\mathbf{d} \in \langle\tilde{\mathsf{G}}\rangle$ such that $\mathbf{c} = \mathbf{d} + (\gamma\mathbf{Y}_0 \cdot \mathbf{d})\mathbf{X}_0$. □

After having retrieved a valid $(\mathbf{X}, \boldsymbol{\lambda})$ pair, say $(\mathbf{A}, \boldsymbol{\Lambda})$, it is possible to decode $\langle\mathsf{G}'\rangle$. Let $\mathbf{x}$ be some plaintext encrypted with Alg. 5. Assume the ciphertext $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c} = \mathbf{x}\mathsf{G}'$ and $\mathbf{e}$ is an error vector, has been intercepted. Then, by Lemma 5.4.3(i) the eavesdropper knows that there exists a $\mathbf{d} \in \langle\tilde{\mathsf{G}}\rangle$ such that

$$\mathbf{c} = \mathbf{d} + (\boldsymbol{\Lambda} \cdot \mathbf{d})\mathbf{A}. \tag{5.14}$$

Define

$$\begin{array}{rccl} y\colon & \mathbb{F}_q & \longrightarrow & \mathbb{F}_q^n \\ & a & \longmapsto & \mathbf{y} + a\mathbf{A} \end{array}$$

and compute all its images. Since $\boldsymbol{\Lambda} \cdot \mathbf{d} \in \mathbb{F}_q$ too, the cryptanalyst can find

$$\mathbf{y} + (-\boldsymbol{\Lambda} \cdot \mathbf{d})\mathbf{A} = \mathbf{d} + (\boldsymbol{\Lambda} \cdot \mathbf{d})\mathbf{A} + \mathbf{e} - (\boldsymbol{\Lambda} \cdot \mathbf{d})\mathbf{A} = \mathbf{d} + \mathbf{e} \in \mathbf{e} + \langle\tilde{\mathsf{G}}\rangle.$$

Eve does not yet know whether she has found the right a, so she assumes $y(a)$ is equal to, say, $\mathbf{D} + \mathbf{E}$ for some $\mathbf{D}$ and $\mathbf{E}$. Because the code $\langle\tilde{\mathsf{G}}\rangle$ has the same distance as the secret code $\langle\mathsf{G}\rangle$, she treats $\mathbf{E}$ as a valid error vector. Decoding returns $\mathbf{D}$ and thus $\mathbf{E}$ and a vector $\mathbf{C}$, supposed to be $\mathbf{c}$. She repeats this procedure until she can verify that $\mathbf{D}$, $\mathbf{E}$ and $\mathbf{C}$ are respectively $\mathbf{d}$, $\mathbf{e}$ and $\mathbf{c}$. If so, the plaintext $\mathbf{x}$ can be recovered by constructing a $k \times k$ invertible submatrix G'' of G' and computing $\mathbf{x} = \mathbf{c}''\mathsf{G}''^{-1}$, where $\mathbf{c}'' \in \mathbb{F}_q^k$ is the vector composed of the k coordinates of $\mathbf{c}$ corresponding to the k selected columns of G'.

Case where the code rate is $> \frac{1}{2}$

At the beginning of this section we have required the dimension of the employed code to be smaller than half its length. However, in SECT. 4.4 we have seen that in practice codes with rate over $\frac{1}{2}$ are used and so do the authors of [2] too. We know that if any $[n, k, d]$ code $\mathcal{C}$ is given, the dimension of its dual code $\mathcal{C}^{\perp}$ is equal to $r = n - k$ (the construction of the dual code of a GRS code has been repeated at the beginning of the present section). So, a possibility to have the attack succeeding also with codes for which it holds $\frac{k}{n} > \frac{1}{2}$ is to work with the dual codes of $\langle \mathsf{G}' \rangle$ and $\langle \tilde{\mathsf{G}} \rangle$ respectively instead of with the codes themselves as we have done until now. Since the attack we have just seen works for $k < \frac{n}{2} - 1$, the "dual version" breaks the cryptosystem whenever the dimension of the employed code satisfies the inequality $r < \frac{n}{2} - 1 \iff k > \frac{n}{2} + 1$. In this case, part (ii) of LEMMA 5.4.3 is needed.

Algorithm 7 – Attack to the new variants [17]

Input: a $k \times n$ public key G' with $k < \frac{n}{2} - 1$, the number $\frac{t}{v}$,
a basis $\mathbf{g}'_1, \ldots, \mathbf{g}'_k$ of $\langle \mathsf{G}' \rangle$, a ciphertext $\mathbf{y}$
Output: a plaintext $\mathbf{x}$

Finding a basis $\mathcal{B}$ for $\mathcal{G}$

$\mathcal{B} \longleftarrow \varnothing$
repeat Search for $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \langle \mathsf{G}' \rangle$
until $\dim(\langle \mathbf{w}_1 \star \mathbf{g}'_1, \ldots, \mathbf{w}_3 \star \mathbf{g}'_k \rangle) \leqslant 2k + 2$ **and** $\dim(\langle \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \rangle) = 3$
$\mathcal{B} \longleftarrow \{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$
$s \longleftarrow 4$
for $s \leqslant k - 1$ **do**
  repeat Search for $\mathbf{w}_s \in \langle \mathsf{G}' \rangle$
  until $\dim(\langle \mathbf{w}_j \star \mathbf{g}'_1, \ldots, \mathbf{w}_j \star \mathbf{g}'_k \rangle) \leqslant 2k + 2 \ \forall j \in \{1, 2, s\}$ **and** $\dim(\langle \mathcal{B} \cup \{\mathbf{w}_s\} \rangle) = s$
  $\mathcal{B} \longleftarrow \mathcal{B} \cup \{\mathbf{w}_s\}$
  $s \longleftarrow s + 1$
end for
return $\mathcal{B}$
The procedure described in [49] yields $\langle \tilde{\mathsf{G}} \rangle$

Decoding the public code

Find $\mathbf{X}_0$ and $\mathbf{Y}_0$ as in equation (5.12) and γ as in (5.13). Construct an invertible $k \times k$ submatrix G'' of G'. Let $\mathbb{F}_q = \{a_1, \ldots, a_q\}$.
$s \longleftarrow 1$
for $s > 0$ **do**
  $\mathbf{y}_s \longleftarrow \mathbf{y} + a_s \mathbf{X}_0$
  Decode $\mathbf{y}_s$ to retrieve $\mathbf{D}_s \in \langle \tilde{\mathsf{G}} \rangle$ and $\mathbf{E}_s$ with lowest possible weight.
  $\mathbf{C}_s \longleftarrow \mathbf{D}_s + (\gamma \mathbf{Y}_0 \cdot \mathbf{D}_s) \mathbf{X}_0$
  if $\mathbf{C}_s + \mathbf{E}_s = \mathbf{y}_s$ **then** ▷ Also check whether $\mathrm{wt}(\mathbf{E}_s) \leqslant \frac{t}{v}$
    return $\mathbf{C}_s \mathsf{G}''^{-1}$
  else $s \longleftarrow s + 1$
  end if
end for

wt \ n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	n \ wt
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1
2	1	2	4	7	11	16	22	29	37	46	56	67	79	92	106	121	137	154	172	191	2
3	1	2	4	8	15	26	42	64	93	130	176	232	299	378	470	576	697	834	988	1160	3
4		1	3	7	15	30	56	98	162	255	385	561	793	1092	1470	1940	2516	3213	4047	5035	4
5			1	4	11	26	56	112	210	372	627	1012	1573	2366	3458	4928	6868	9384	12597	16644	5
6				1	5	16	42	98	210	420	792	1419	2431	4004	6370	9828	14756	21624	31008	43605	6
7					1	6	22	64	162	372	792	1584	3003	5434	9438	15808	25636	40392	62016	93024	7
8						1	7	29	93	255	627	1419	3003	6006	11440	20878	36686	62322	102714	164730	8
9							1	8	37	130	385	1012	2431	5434	11440	22880	43758	80444	142766	245480	9
10								1	9	46	176	561	1573	4004	9438	20878	43758	87516	167960	310726	10
11									1	10	56	232	793	2366	6370	15808	36686	80444	167960	335920	11
12										1	11	67	299	1092	3458	9828	25636	62322	142766	310726	12
13											1	12	79	378	1470	4928	14756	40392	102714	245480	13
14												1	13	92	470	1940	6868	21624	62016	164730	14
15													1	14	106	576	2516	9384	31008	93024	15
16														1	15	121	697	3213	12597	43605	16
17															1	16	137	834	4047	16644	17
18																1	17	154	988	5035	18
19																	1	18	172	1160	19
20																		1	19	191	20
21																			1	20	21
22																				1	22
$\Sigma(n)$	1	1	4	5	6	7	30	38	47	57	244	312	392	485	2062	2654	3368	4220	17824	23051	$\Sigma(n)$
$\tilde{\Sigma}(n)$	1	1	1	1	1	1	37	46	56	67	79	92	106	121	2617	3214	4048	5036	6196	7547	$\tilde{\Sigma}(n)$

Table 8 – The number of words of weight wt included in the code defined by the parity-check matrix $\left(10\ldots0 \,\middle|\, 1\right)$ for $n = 3, \ldots, 22$. From the values of $\Sigma(n)$ we see that it is $S = \{4, 30, 244, 2062, 17824\}$. In the last line we have inserted the function $\tilde{\Sigma}(n)$ corresponding to the subspaces generated by the all-one parity-check matrix $\left(\mathbf{1}\right)$. In this case, $S = \{37, 2617\}$.

Conclusion

During this thesis we have analysed the McEliece, the Niederreiter and the Sidel'nikov cryptosystems, focussing in particular on the former two, which are known to share the same security level [22] and are candidates to play a role in the post-quantum era if suitable parameters are chosen (see also [6] and [32, p. 115]). As a first thing, in Sect. 2.2 we presented a corrected version of an algorithm, initially conceived by Guillot but not published [12, Ref. 11] for constant weight encoding, which can be performed in polynomial time. This is needed for the Niederreiter encryption scheme, since the ciphertext must have a weight at most equal to the error-correcting capability of the code in use.

From our analysis of the running time of the considered cryptosystems, we can say that, on the whole, Niederreiter seems to be more advantageous than McEliece. First of all, the former is resistant against the message-resend attack, since, unlike the latter, it is deterministic. Secondly, Niederreiter allows systematic keys (i.e. less storage capacity and less computations needed) without compromising the security. This point is particularly relevant for small values of t: for example, the parameter values $m = 12$ and $t = 45$ produce generator matrices of size 3556×4096, occupying 1,8 MB, while the public key in systematic form (or equivalently, the stored part in the Niederreiter version with the same parameters) would require only 240 kB. If instead $t = 15$, the storage needs of a non-systematic public matrix would be 23 times higher than its systematic version.

Regarding the transmission rates, McEliece is generally preferable, with exception of parameter values close to the ones proposed by McEliece himself. Better rates are guaranteed by the Sidel'nikov cryptosystem if the assumptions about the great error-correcting capabilities of RM codes are exploited [28, 40]. Otherwise, the very low rates and the extremely large key sizes make it unattractive. Moreover, it has already been broken [28]. The large keys are counterbalanced by the very fast encryption and decryption procedures if compared for example with RSA, which is still advantageous if considering the key sizes and the rate, but does not resist any attack performed by quantum computers [5, p. 1].

In summary, it can be stated that if m increases (11 or 12) and t is low, one has both a good rate as well as fast encryption and decryption procedures. The only missing element is the work factor. A code designer must find a compromise between a small t and a high resistance against attacks. If $m = 12$, good security levels (WF around 2^{100}) are attained even for $t < 20$. Clearly, all this is valid only provided that some classes of codes as avoided: the Sidel'nikov-Šestakov attack (Sect. 4.5) shows that if a McEliece-type cryptosystem is constructed with GRS codes, an eavesdropper is able to perform a polynomial-time structural attack against it.

In response to this, the authors of [2] proposed a new variant of the McEliece cryptosystem in order to make the use of GRS codes possible (see Sect. 5.1). The permutation matrix is substituted by a more complex matrix which the legal user is able to reduce to a simpler

one. Unfortunately, this new approach, which requires the error vector to respect some public constraints, could endanger the system. In order to avoid this possibility, Bob could keep these constraints secret, publishing only the set $\mathcal{S}$ (or a subset of it) of the vectors satisfying them (see Sect. 5.3). With the means available to us and working over the binaries, we tried to construct an extrapolating function returning the needed code length n in function of the desired size of $\mathcal{S}$. For this, we calculated the amount of vectors having weight below $\frac{n}{4}$. We noticed that the choice of making public for encoding all such vectors or just the ones with the highest weight among them has no particular influence on the needed value of n. Moreover, we see that if one wanted to have at least, say, $2^{512} > 10^{154}$ error vectors available, a code of length 1024 ($m = 10$) would be enough. Sadly, very recently also this variant was cryptanalysed [17] with an approach exploiting the existence of a code being a large subcode of both the public code and of a permutation of the private code, enabling the eavesdropper to retrieve the latter.

Bibliography

[1] Adams, C. M. and Meijer, H: *Security-related comments regarding McEliece's public-key cryptosystem*, in: Advances in cryptology – CRYPTO '87, proceedings, ed. by Carl Pomerance, Berlin etc. 1988, pp. 224–228;

[2] Baldi, M., Bianchi, M., Chiaraluce, F., Rosenthal, J. and Schipani, D.: *Enhanced public key security for the McEliece cryptosystem*, `arXiv:1108.2462v2`, viewed on 8th February 2012, 16:23;

[3] Berger, T. P., Cayrel, P.-L., Gaborit, P. and Otmani, A.: *Reducing key length of the McEliece cryptosystem*, in: Africacrypt 2009, ed. by Bart Preneel, Berlin etc. 2009, pp. 77–97;

[4] Berlekamp, E. R., McEliece, R. J. and van Tilborg, H. C. A.: *On the inherent intractability of certain coding problems*, in: IEEE Transactions on information theory 24 (1978), pp. 384–386;

[5] Bernstein, D. J.: *Grover vs. McEliece*, version of 3rd March 2010, permanent ID: `e2bbcdd82c3e967c7e3487dc945f3e87`, viewed on 16th October 2011, 22:43;

[6] Bernstein, D. J., Lange, T. and Peters, C.: *Attacking and defending the McEliece cryptosystem*, `http://eprint.iacr.org/2008/318.pdf`, viewed on 20th June 2012, 17:24;

[7] Berson, T. A.: *Failure of the McEliece public-key cryptosystem under message-resend and related-message attack*, in: Advances in cryptology – Proceedings of Crypto '97, ed. by B. Kaliski, Berlin etc., pp. 213–220;

[8] Biswas, B.: *Implementational aspects of code-based cryptography*, PhD thesis, `hal.inria.fr/pastel-00523007/PDF/thesis.pdf`, viewed on 26th March 2012, 18:56;

[9] Bosch, S.: *Algebra*, Berlin etc. 2009;

[10] Cooke, B.: *Reed-Muller Error Correcting Codes*, in: MIT undergraduate journal of mathematics 1 (1999), pp. 21–26;

[11] Cover, T. M.: *Enumerative source encoding*, in: IEEE Transactions on information theory 19 (1973), pp. 73–77;

[12] Fischer, J.-B. and Stern, J.: *An efficient pseudo-random generator provably as secure as syndrome decoding*, in: Eurocrypt 1996, ed. by U. Maurer, Berlin etc. 1996, pp. 245–255;

[13] Fitzpatrick, P. and Ryan, J. A.: *Counting irreducible Goppa codes*, in: Journal of the Australian Mathematical Society 2001, pp. 299–306;

[14] Fontein, F.: *Einführung in die Algebra*, lecture held at the University of Zurich in Autumn Semester 2011;

[15] Gabidulin, E. M.: *Public-key cryptosystems based on linear codes*, Delft (NL) 1995;

[16] Gabidulin, E. M. and Kjelsen, O.: *How to avoid the Sidel'nikov-Šestakov attack*, in: Error control, cryptology, and speech compression, ed. by Andrew Chmora and Stephen B. Wicker, Berlin etc. 1994, pp. 25–32;

[17] Gauthier, V., Otmani, A. and Tillich, J. P.: *A distinguisher-based attack on a variant of McEliece's cryptosystem based on Reed-Solomon codes*, `arXiv:1204.6459v1`, viewed on 14th May 2012, 21:43;

[18] Hill, R.: *A first course in coding theory*, Oxford 1986;

[19] Horster, P. and Sperling, R.: *Das Kryptosystem von McEliece: ein Public-key-Kryptosystem auf der Basis von Goppa-Codes*, Heidelberg 1990;

[20] Heninger, N., Durumeric, Z., Wustrow, E. and Halderman, J. A.: *Mining your Ps and Qs: detection of widespread weak keys in network devices*, `https://factorable.net`, viewed on 21st July 2012, 12:38, to appear in August 2012 in Proceedings of the 21st USENIX security symposium;

[21] Leutbecher, A.: *Zahlentheorie: Eine Einführung in die Algebra*, Berlin etc. 1996;

[22] Li, Y. X., Deng, R. H. and Wang, X. M.: *On the equivalence of McEliece's and Niederreiter's public-key cryptosystems*, in: IEEE Transactions on information theory 40 (1994), pp. 271–273;

[23] Lidl, R. and Niederreiter, H.: *Introduction to finite fields and their applications*, Cambridge 1986;

[24] Loidreau, P.: *Strengthening McEliece cryptosystem*, in: Asiacrypt 2000, ed. by Tatsuaki Okamoto, Berlin etc. 2000, pp. 585–598;

[25] McEliece, R. J.: *A public-key cryptosystem based on algebraic coding theory*, in: DNS Progress Report 42–44 (1978), pp. 114–116;

[26] McWilliams, F. J. and Sloane, N. J. A.: *The theory of error-correcting codes*, Amsterdam 1977;

[27] Minder, L.: *Cryptography based on error correcting codes*, PhD thesis, `http://algo.epfl.ch/_media/en/projects/lorenz_thesis.pdf`, viewed on 8th January 2012, 17:43;

[28] Minder, L. and Shokrollahi, A.: *Cryptanalysis of the Sidel'nikov Cryptosystem*, in: Eurocrypt 2007, ed. by Antoine Joux, Berlin etc. 2007, pp. 347–360;

[29] Moon, T. K.: *Error Correction Coding: Mathematical Methods and Algorithms*, Hoboken (USA) 2005;

[30] Niederreiter, H.: *Knapsack-type cryptosystems and algebraic coding theory*, in: Problems of control and information theory 15 (1986), pp. 159–166;

[31] Niederreiter, H. and Xing, C.: *Algebraic geometry in coding theory and cryptography*, Princeton 2009;

[32] Overbeck, R. and Sendrier, N.: *Code-based cryptography*, in: Post-Quantum Cryptography 2009, ed. by Daniel J. Bernstein etc., Berlin 2009, pp. 95–145;

[33] Patterson, N. J: *The algebraic decoding of Goppa codes*, in: IEEE Transactions on information theory 21 (1975), pp. 203–207;

[34] Peters, C.: *Information-set decoding for linear codes over* $\mathbb{F}_q$, in: Post-Quantum Cryptography 2010, ed. by Nicolas Sendrier, Berlin 2010, pp. 81–94;

[35] Rabin, M. O.: *Probabilistic algorithms in finite fields*, in: SIAM Journal on computing 9 (1980), pp. 273–280;

[36] Riek, J. R.: *Observations on the application of error correcting codes to public key encryption*, in: Proceedings IEEE – 1990 international Carnahan conference on security technology: crime countermeasures, Lexington (USA) 1990, pp. 15–18;

[37] Roman, S.: *Coding and Information Theory*, New York 1992;

[38] Rosenthal, J.: *Coding theory*, lecture held at the University of Zurich in Autumn Semester 2010;

[39] Sendrier, N.: *On the security of the McEliece public-key cryptosystem*, in: Information, coding and mathematics: Proceedings of workshop honoring prof. Bob McEliece on his 60th birthday, ed. by M. Blaum etc., Norwell (USA) 2002, pp. 141–163;

[40] Sidel'nikov, V. M.: *A public-key cryptosystem based on binary Reed-Muller codes*, in: Discrete Mathematics and Applications 4 (1994), pp. 191–207;

[41] Sidel'nikov, V. M. and Šestakov, S. O.: *O sisteme šifrovanja postroennoj na osnove obobščennych kodov Rida-Solomona*, in: Diskretnaya matematika 4 (1992), pp. 57–63 (in Russian);

[42] Sidel'nikov, V. M. and Šestakov, S. O.: *On an encoding system constructed on the basis of generalized Reed-Solomon codes*, in: Discrete Mathematics and Applications 2 (1992), pp. 439–444;

[43] Simmons, G. J.: *The number of irreducible polynomials of degree* n *over* $GF(p)$, in: The American Mathematical Monthly 77 (1970), pp. 743–745;

[44] Sun, H.-M.: *Enhancing the security of the McEliece public-key cryptosystem*, in: Journal of information science and engineering 16 (2000), pp. 799–812;

[45] van Lint, J. H. and Springer, T. A.: *Generalized Reed-Solomon codes from algebraic geometry*, in: IEEE Transactions on information theory 33 (1987), pp. 305–309;

[46] van Tilborg, H. C. A.: *Encyclopedia of cryptography and security*, New York 2005, article *McEliece public key cryptosystem*;

[47] van Tilborg, H. C. A.: *Encyclopedia of cryptography and security*, New York 2005, article *Niederreiter encryption scheme*;

[48] Verheul, E. R., Doumen, J. M. and van Tilborg, H. C. A.: *Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece public-key cryptosystem*, in: Information, coding and mathematics: Proceedings of workshop honoring prof. Bob McEliece on his 60th birthday, ed. by M. Blaum etc., Norwell (USA) 2002, pp. 99–119;

[49] Wieschebrink, C.: *Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes*, in: Post-Quantum Cryptography 2010, ed. by Nicolas Sendrier, Berlin 2010, pp. 61–72;

[50] Williams, V. V.: *Breaking the Coppersmith-Winograd barrier*, `http://www.cs.berkeley.edu/~virgi/matrixmult.pdf`, viewed on 30th April 2012, 00:48.