Universidade de Aveiro
2022

Cláudia Maria
Ferreira Sebastião

Chaves mais pequenas para criptossistemas de McEliece usando codificadores convolucionais

Smaller keys for McEliece cryptosystems using convolutional encoders

Universidade de Aveiro
2022

Cláudia Maria Ferreira Sebastião

Chaves mais pequenas para criptossistemas de McEliece usando codificadores convolucionais

Smaller keys for McEliece cryptosystems using convolutional encoders

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Matemática, Programa Doutoral em Matemática, realizada sob a orientação científica do Doutor Paulo José Fernandes Almeida, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro, e do Doutor Diego Oscar Napp Avelli, Professor Titular do Departamento de Matemática da Universidade de Alicante, Espanha.

Dedico este trabalho aos meus filhos.

o júri

presidente	Prof. Doutor António José Arsénia Nogueira Professor Catedrático, Universidade de Aveiro
vogais	Prof. Doutor Joan Josep Climent Coloma Professor Catedrático, Universitat d'Alacant
	Prof. Doutora Verónica Requena Arevalo Professora contratada Doctora, Universitat d'Alacant
	Prof. Doutor António Machiavelo Professor Auxiliar, Universidade do Porto
	Prof. Doutora Rita Isabel Gonçalves Simões Professora Auxiliar, Universidade de Aveiro
	Prof. Doutor Paulo José Fernandes Almeida (Orientador) Professor Auxiliar, Universidade de Aveiro

agradecimentos

O meu maior agradecimento é dirigido aos meus orientadores Professor Doutor Paulo José Almeida e Professor Doutor Diego Oscar Napp Avelli que foram a génese deste trabalho. Não existem palavras suficientes para agradecer a partilha de conhecimento, a paciência, o encorajamento, o apoio, a dedicação, a inesgotável persistência, energia e entusiasmo contagiante. Metaforicamente falando, foram anos a "puxar" um barco que, durante a sua viagem, foi sendo alvo de algumas tempestades. A sua força foi superior e conseguiram que esse barco retomasse a rota correta e fosse capaz de atracar. Ficarei para sempre grata a estes dois seres humanos incríveis. Agradeço à Universidade de Aveiro, em particular aos elementos do Departamento de Matemática e do CIDMA pela forma como fui acolhida e apoiada. Grata também a todos os meus alunos, amigos e família que foram naturalmente a minha fonte de motivação e resiliência. Por último, aos meus filhos, a minha inspiração e à minha mãe, a minha pedra basilar a quem devo absolutamente tudo.

palavras-chave

Criptografia pós-quântica, Criptografia baseada em códigos, Criptossistema de McEliece, Códigos convolucionais, Criptografia de Chave Pública.

resumo

A chegada da era da computação quântica é uma ameaça real à confidencialidade e integridade das comunicações digitais. É, por isso, urgente desenvolver técnicas criptográficas alternativas que sejam resilientes à computação quântica. Este é o objetivo da criptografia pós-quântica. O Criptossistema de McEliece continua a ser uma das alternativas pós-quânticas mais promissora, contudo, a sua principal desvantagem é o tamanho da chave pública, uma vez que é muito maior do que o das outras alternativas. Nesta tese estudamos as propriedades algébricas deste tipo de criptossistemas e apresentamos uma nova variante que usa um codificador convolucional para mascarar o código de Generalized Reed-Solomon. Conduzimos uma criptoanálise dessa nova variante para mostrar que altos níveis de segurança podem ser alcançados usando uma chave significativamente menor do que as variantes existentes do esquema de McEliece. Ilustramos, assim, as vantagens do criptossistema proposto apresentando vários exemplos práticos.

keywords

Post-quantum cryptography, Code-based cryptography, McEliece Cryptosystem, Convolutional codes, Public Key Cryptography

abstract

The arrival of the quantum computing era is a real threat to the confidentiality and integrity of digital communications. So, it is urgent to develop alternative cryptographic techniques that are resilient to quantum computing. This is the goal of pos-quantum cryptography. The code-based cryptosystem called Classical McEliece Cryptosystem remains one of the most promising post-quantum alternatives. However, the main drawback of this system is that the public key is much larger than in the other alternatives. In this thesis we study the algebraic properties of this type of cryptosystems and present a new variant that uses a convolutional encoder to mask the so-called Generalized Reed-Solomon code. We conduct a cryptanalysis of this new variant to show that high levels of security can be achieved using significant smaller keys than in the existing variants of the McEliece scheme. We illustrate the advantages of the proposed cryptosystem by presenting several practical examples.

Contents

Chapter 1

Introduction

Suppose that two people, say Alice and Bob, want to exchange a secret key or message. To this end, Bob builds a public-key cryptosystem (PKC), *i.e.*, he generates two keys: a private key that he keeps secret and a public key that he publishes and everyone can see. Then, Alice in order to send a message to Bob, uses his public key to encrypt her message and sends it to Bob. The system is developed in such a way that only someone who knows the secret key (that is Bob) can decrypt the encrypted message. Hence, the cryptosystem is considered to be secure if Eve, an eavesdropper, or Olga, an opponent, cannot reconstruct the message knowing only the public key and the encrypted message.

There are several PKC that are considered secure, being RSA and ECC (Elliptic Curve Cryptography) the most popular. In the RSA cryptosystem, Bob generates two distinct prime numbers p and q and computes an integer e that is coprime with $\phi(pq) = (p-1)(q-1)$. Bob publishes $n = pq$ and e and keeps secret p and q. If Alice wants to send a secret message m to Bob she uses the public key and sends

$$c = m^e \mod n.$$

Bob computes an integer d such that $dc \equiv 1 \mod \phi(n)$ to decrypt c:

$$c^d \equiv (m^e)^d \equiv m^{1-k\phi} \equiv m \mod n,$$

see more details in [54]. Note that Bob is the only one that can compute d, as for that one needs to know $\phi(n)$, or equivalently the two primes p and q. To break the system, *i.e.*, to obtain m, Eve could compute the decomposition of n but this is considered unfeasible if the primes p and q are selected carefully.

The ECC is based on the El Gamal cryptosystem, but instead of having the group $(\mathbb{Z}_p, \cdot)$, we consider the group $(E, +)$, where E is an elliptic curve.

Bob chooses a prime number p, an elliptic curve E over $\mathbb{Z}_p$, and a point P with a large order k. Bob also chooses $0 < b < k$ randomly and calculates the point
$$B \equiv bP \mod p.$$

The public key of Bob is (p, E, P, k, B). His secret key is the exponent b. Normally only B is published, because all the other parameters are "known". The set of plaintexts is $\{0, 1, \ldots, p-1\}$.

For example, many sites use the standard NIST P-256 with
$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$
and
$$\begin{aligned} y^2 &= x^3 - 3x + \quad 41058363725152142129326129780047268409114441015993 \\ &\qquad 725554835256314039467401291. \end{aligned}$$

If Alice wants to send a message to Bob, she uses Bob's public key (p, E, P, k, B), chooses randomly a number $a \in \{1, \ldots, k-1\}$ and computes the point
$$A \equiv aP \mod p.$$
In order to cipher a message m, Alice finds the positive solution y_m of the equation
$$y^2 = m^3 + um + v$$
and considers the point $M = (m, y_m)$ of E. She then computes the point
$$C \equiv aB + M \mod p,$$
and sends to Bob the pair (A, C).

In order to decipher the message m, Bob computes $x = k - b$ and the point $xA + C \mod p$. Notice that
$$\begin{aligned} xA + C &= (k-b)aP + abP + M \\ &= a(kP) + M = M. \end{aligned}$$
The first coordinate of M is m.

In both of these cryptosystems the computation cost to cipher and decipher is much larger than other PKC, namely code based cryptosystems, since one has to use fast modular exponentiation (in both) and operations in elliptic curves (in the second). But they have the advantage of having very small public keys, namely around 2048 or 4096 bits for RSA and 256 bits for ECC. However, it was pointed out recently by National Institute of Standards and Technology (NIST) [13] that it might be possible that a large-scale quantum computer will be available in 2030 capable of breaking the RSA and most of the widely-used public key cryptosystems, such as ElGamal or ECC. For this reason NIST urged researchers all around the world to come up with systems which might still be safe in an environment where quantum computers exist [34]. Hence, there is a huge urgency for research in the so-called post-quantum cryptography. This is the starting point and the main motivation of this thesis.

NIST asked for new proposals to be submitted by November the 30th, 2017, and 70 proposals were submitted [34]. One of the most promising families of public key post-quantum cryptosystems are the code-based cryptosystems. As opposed to RSA or ElGamal, whose security relays on the integer factorization or discrete logarithms problems, the security of code-based cryptosystems rely on the hardness of decoding a linear block code without any visible structure which is known to be an NP-hard problem [9]. Moreover, such decoding problem is known to be potentially secure in a post-quantum computer environment, as it has so far proved resistant to quantum computer-aided attacks. The most prominent code-based cryptosystem is the McEliece cryptosystem. Here, the secret key is a code for which an efficient decoding algorithm is known. The public key is a masking version of the secret code that appears to be a random code, thus hiding the structure of the code that allow the efficient decoding. Niederreiter [47] proposed in 1986 a cryptosystem, which is similar to the McEliece system, but instead of using the generator matrix of a code it uses the parity check matrix of the code. Only four cryptosystems are third round finalists to the Post Quantum Cryptosystem (PQC) Standardization Process (with five more as alternate candidates), and one of them is the classic McEliece cryptosystem.

The security of McEliece cryptosystem is based on two assumptions. First, it is unfeasible to decode the seemingly random public code, and second it is intractable for an attacker to recover the underlying structure of the secret code from the public code. As the best known algorithms for decoding random linear codes are exponential in the length of the code, the first assumption can be attained by simply increasing the size of the code. The second assumption heavily relies on the type of underlying code that is used. The original McEliece and Niederreiter cryptosystems use Goppa codes as the private key, and hide its structure by a dense transformation matrix and a permutation matrix. Surprisingly, these cryptosystems have resisted cryptanalysis for more than forty years, as there is not currently any efficient (polynomial-time) attack to the systems. Also, another advantage of the McEliece and Niederreiter cryptosystems is that they allow for faster encryption and decryption procedures when compared to other cryptosystems, like RSA or ElGamal. However, they have not been widely implemented due to two main disadvantages: low encryption rate and large key size, both due to the Goppa codes they are based on (see Table 1.1 for a comparison between the binary Goppa code-based Niederreiter cryptosystem and RSA).

	Binary Goppa code-based Niederreiter	RSA
Key size	1537536	6144
Encryption complexity	72	5406
Decryption complexity	15302	6643013

Table 1.1: Comparison between Goppa code-based cryptography and RSA in bit operations. (extracted from [6])

For this reason, there has been many attempts in the last forty years to replace the Goppa codes by other families of codes in order to reduce the key size [5, 6, 7, 10, 11, 17, 18, 22]. The Generalized Reed-Solomon (GRS) code is the most preferred alternative, as a significant improvement in the reduction of the key size would be obtained if the class of GRS could be securely included as a secret key in the McEliece cryptosystem. The use of GRS codes could yield significant advantages, since they achieve maximum error correction capability, *i.e.*, they are maximum distance separable (MDS) codes. This translates into much smaller keys for the same security level with respect to Goppa codes. The first proposal to use GRS codes directly in the McEliece system was proposed by Niederreiter [47] but it was broken by the attack of Sidelnikov and Shestakov [56]. The Berger-Loidreau cryptosystem [7] is a variant of the Niederreiter scheme which resists the Sidelnikov-Shestakov attack. However, extending the method by Sidelnikov and Shestakov an efficient attack on Berger-Loidreau cryptosystem was presented in [61]. In fact, as GRS have very strong algebraic properties it is difficult to mask and therefore, to the best of our knowledge, all code-based cryptographic practical schemes using GRS codes or low-codimensional subcodes of GRS codes have been partially or fully broken [16]. Most of these attacks use a square code distinguisher to retrieve the GRS structure of the secret key from the public code.

One recent interesting idea to remove the algebraic structure of GRS codes was to replace the permutation used in the original McEliece cryptosystem with more general transformations, [6, 19, 32, 33, 43, 59, 60]. To thwart the attack of Sidelnikov and Shestakov and its variants, the BBCRS scheme [6] proposed the use of the sum of matrices $T + R$ for masking instead of a permutation matrix, where T is a matrix with column weight m and R is a matrix of rank z. Although modifications of this idea and different parameters are currently under investigation, this system was broken in [19] using attacks based on the (twisted) Schur square code distinguisher that permits to distinguish GRS codes from random ones. See also [17, 42] for similar distinguishers in the context of AG codes. However, the attack works only for $m < 2$, and the *weight two masking* proposal, *i.e.*, when $m = 2$, can successfully hide the structure of the GRS code, even under the Schur product [11, 33]. This important fact will be used in our proposal and will allow the use of GRS codes in the proposed variant of the McEliece cryptosystem. Note that in the BBCRS scheme, when $m \geq 2$, the structure of the GRS codes can be successfully hidden but as less errors could be added in this case, the length and dimension of the code have to be increased in order to protect it against ISD attacks. Unfortunately, this implies that the key size becomes very similar to the size of the original version using Goppa codes. This, in turn, makes these variants less appealing as the desired key reduction is not achieved.

Another interesting variant, which we call the LJ scheme, was proposed in [39] where the secret code is a convolutional code. Convolutional codes are a powerful and widely used class of codes where the encoded bits depend on the current k input bits and a few past input bits, *i.e.*, the encoder has memory. Hence, the main difference between block and convolutional codes, is that at the encoder, in a convolutional code, we may have different states and after encoding the input, we possibly move into another state. Convolutional codes can be represented via a state machine, which corresponds to a view of the encoder as a set of states with well-defined transitions between them. The state machine view is fundamental for decoding, e.g., using the Viterbi algorithm.

One of the most appealing features of the LJ scheme is the fact that its secret generator matrix contains large parts that are generated completely at random. Another interesting property is that convolutional codes allow to deal with sequences of data in a sequential fashion which makes the computations of data very efficient. But the authors already pointed out that this approach suffers from two main problems. The first one being the lack of efficient decoding algorithms, namely, that if one wants a maximum likelihood decoding, such as the Viterbi decoding algorithm, the memory used must be very limited. The second problem stems from the fact that convolutional codes usually start from the all zero state, and therefore the first code symbols that are generated will have low weight parity-checks, which would imply security threats. In fact, the schemes in [39] had low weight codewords that revealed the underlying code structure, and was broken in [35].

In this dissertation we continue this line of research and explore a new variant that allows the use of GRS codes using a convolutional mask. In our scheme, the plaintext is not a block vector but a stream of smaller vectors sent in a sequential fashion. The public key is given by the polynomial convolutional encoder $G'(D) = S(D)\, G\, P(D^{-1}, D)$ where G is the generator matrix of a GRS code, $S(D)$ a polynomial matrix and $P(D^{-1}, D)$ an invertible Laurent polynomial matrix. Hence, the proposed class of convolutional codes uses GRS codes adding a convolutional layer to it in order to thwart the key recovery attack against GRS codes, and at the same time admits a simple iterative algebraic decoding algorithm. This idea is different from the above ideas, and therefore the cryptanalysis has to be adapted to this case.

The matrices $S(D)$ and $P(D^{-1}, D)$ are selected to protect against both ISD and structural attacks. A crucial fact to ensure security against ISD attacks to the first blocks and bootstrap from there, is that the truncated sliding matrices of the convolutional encoder are not invertible, and recovering the initial blocks is not possible. As for structural attacks, we build our matrix $P(D^{-1}, D)$ to provide weight-ρ masking at every instant with $\rho \geq 2$. As pointed out in [11, 32], see also [33], the weight-two masking appears to be enough to remove any identifiable algebraic structure from the public code, and therefore structural attacks, and in particular any distinguisher attack based on the Schur product, seem to fail as well. We note that our construction uses matrices with large parts, generated completely at random and can be easily constructed. Moreover, it seems not obvious that subcodes of the public code with small support can be computed in this proposal. Thus, this thesis presents a novel code-based cryptosystem that is very different from the existing variants of the McEliece cryptosystem. Among its several interesting properties we underline that the proposed scheme significantly reduces the key size of the public key, for a given security level, with respect to existing McEliece type cryptosystems.

The structure of the thesis

This thesis is divided into 5 chapters. A briefly outline of the contents of the chapters is given below.

Chapter 2 - Linear codes

In Chapter 2 we present the necessary background of linear coding theory that is useful for the cryptographic issues treated in this dissertation. In particular, we focus on presenting the basic theory of two classes of linear block codes (Goppa codes and GRS codes) and the class of convolutional codes. We show how these codes are defined, their main properties and some of their efficient decoding algorithms. Most of the definitions and results can be also found in [40, 45].

Chapter 3 - Block code Cryptography

Chapter 3 is devoted to describe how the McEliece cryptosystems work. We present the original version together with Neiderreiter variant using the parity check matrix and the most interesting variants that use GRS codes. Their corresponding attacks are analyzed. In this chapter we also explain when these variants are considered secure despite the fact that they are not practical. These results will be useful to our proposal.

Chapter 4 - McEliece Cryptosystems with Convolutional Encoders

Chapter 4 is the main chapter of the thesis and contains the details of the proposed McEliece-type cryptosystem.

This novel variant is very different from the existing variants in many aspects. One of the most interesting features is that it allows to significantly reduce the public key with respect to other alternatives of the McEliece cryptosystem. As opposed to other variants of the McEliece cryptosystem, where block codes are used, we propose the use of a convolutional encoder of a convolutional code to be part of the public key. The secret key is constituted by the generator matrix of a Generalized Reed-Solomon code and two invertible matrices. In this scheme the plaintext is divided into a sequence of shorter messages and encrypted sequentially. We analyze ISD and structural attacks, and conclude the chapter by presenting several examples for different security levels to illustrate the significant advantages of the proposed PKC. Some of these results have been partially presented in [2, 3] and most of the results in this chapter have been submitted for publication, see [1].

Chapter 5: Conclusions and future work

In the last chapter we summarize the main results obtained in our work, and discuss some interesting avenues for future work.

Chapter 2

Linear Codes

Communication systems are everywhere and they have become increasingly important with the development of new technologies for data communications and data storage. Errors in digital communication systems may occur due to noisy communication channels and in order to guarantee reliable transmission or to recover degraded data, techniques from Coding Theory are used. Coding theory deals with the description of information in a way that it is possible to discover and correct mistakes as long as the number of errors is not too large. Hence, the aim of Coding Theory is to develop methods to detect and correct these errors. In the last decades it became an active subject of research in different areas of knowledge such as mathematics, computer science, electrical engineering, statistics, among others. Coding theory is one of the main mathematical tools in information and communication theory. Decoding a general code has a huge complexity and this can be used to develop cryptosystems. Researchers call cryptosystems based on the hardness of the decoding of codes ”code-based cryptographic system”. It is believed that such systems will still be safe even once a practically quantum computer is built.

Linear codes are an important class of error-correcting codes that is mainly divided into two types: block codes and convolutional codes. In this chapter we introduce some definitions and properties of these codes. In particular we describe two linear block codes that can be decoded efficiently, which means that there exists an efficient polynomial time decoding algorithm for them: Generalized Reed Solomon (GRS) codes and Goppa codes. Goppa codes was the family of codes used in the original proposal of the McEliece cryptosystem, and GRS codes are the class of codes that we shall use in our variant of the McEliece cryptosystem. The chapter concludes with the necessary introduction of convolutional codes needed to understand the construction of our proposal.

2.1 Basic concepts

We start by introducing the notion of a linear code. In order to have a code, we need an alphabet. Let $\mathbb{F}_q$ be the alphabet, a finite field with q elements where q is a power of a prime integer p. In practice, p is usually 2, but any prime power q is allowed.

Definition 2.1 (Linear code)**.** *Let n, $k \in \mathbb{N}$ with $k < n$. An $[n, k]$* ***linear block code*** *$\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional linear subspace of the vector space $\mathbb{F}_q^n$.*

The elements of $\mathcal{C}$ are called *codewords*. Each message is represented as a vector $\mathbf{u}$ of $\mathbb{F}_q^k$ and mapped to a unique codeword $\mathbf{v} \in \mathbb{F}_q^n$. The integers k and n are the *dimension* and *length* of $\mathcal{C}$, respectively, and the difference $n - k$ is called the *redundancy* of the code. The ratio $R = k/n$ is known as the *transmission rate* and measures the proportion of information transmitted in each codeword.

Formally one can interpret encoding as applying an injective $\mathbb{F}_q$-linear function:

$$\begin{array}{rcl} f : \mathbb{F}_q^k & \longrightarrow & \mathbb{F}_q^n \\ \mathbf{u} & \mapsto & \mathbf{v} \end{array}$$

The vector $\mathbf{u}$ is called the *information vector* and it is encoded into the *codeword* $\mathbf{v} \in \mathbb{F}_q^n$. The message that is originally sent through a given channel, the codeword $\mathbf{v}$, when passing through the transmission channel can be affected by the existence of noise. So, the message received by the decoder may not be exactly $\mathbf{v}$ but rather $\mathbf{y} = \mathbf{v} + \mathbf{e}$, where $\mathbf{e} \in \mathbb{F}_q^n$ is called *error vector*. Therefore, the decoder will have to decide from $\mathbf{y}$ what would have been the most likely vector sent, *i.e.*, the most likely error that could have occurred. As errors are assumed to occur rarely, the decoder search for the codeword that is more similar to $\mathbf{y}$, that is, the $\mathbf{v}$ nearest to $\mathbf{y}$. In this way it is necessary to introduce a notion of distance.

Definition 2.2 (Hamming Distance)**.** *Given two vectors $x, y \in \mathbb{F}_q^n$, we define the* ***Hamming Distance*** *between x and y, $d(x, y)$, to be the number of places where x and y differ, i.e., writing $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$, then*

$$d_H(\boldsymbol{x}, \boldsymbol{y}) = |\{i \mid x_i \neq y_i, 1 \leq i \leq n\}|.$$

Given a code $\mathcal{C}$ of length n and dimension k, let

$$d = \min\{d_H(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}\}.$$

We call d the **(Hamming) distance** of the code $\mathcal{C}$ and we say that $\mathcal{C}$ is an $[n, k, d]$ code. To simplify notation sometimes we will just write $[n, k]$ instead of $[n, k, d]$.

Definition 2.3 (Hamming Weight). *The **Hamming Weight** of a vector $\boldsymbol{x} \in \mathbb{F}_q^n$ is $wt(\boldsymbol{x}) = d_H(\boldsymbol{x}, \boldsymbol{0})$, the distance of $\boldsymbol{x}$ to the zero vector.*

The weight of a vector $\mathbf{x} = (x_1, \ldots, x_n)$, is equal to the number of non-zero positions in it, i.e.,

$$wt(\mathbf{x}) = |\{i \mid x_i \neq 0, 1 \leq i \leq n\}|.$$

Clearly, $d_H(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y})$.

The relation between d and the maximum number of errors that a code $\mathcal{C}$ can correct is given by the following lemma:

Theorem 2.4. *The maximum number of errors that an $[n, k, d]$ code can correct, called the* error correcting capability, *is $\lfloor \frac{d-1}{2} \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer smaller or equal than x.*

Proof. [24, p. 10] □

Proposition 2.5 (Unique decoding). *Let $\mathcal{C}$ be an $[n, k, d]$ code with error correcting capability t. For $\boldsymbol{y} \in \mathbb{F}_q^n$, there exists at most one codeword $\boldsymbol{v} \in \mathcal{C}$ such that $d(\boldsymbol{y}, \boldsymbol{v}) \leq t$.*

Proof. Suppose that there exist $\mathbf{v} \in \mathcal{C}$ and $\mathbf{v}' \in \mathcal{C}$ such that $d(\mathbf{y}, \mathbf{v}) \leq t$ and $d(\mathbf{y}, \mathbf{v}') \leq t$. So, $d(\mathbf{y}, \mathbf{v}) + d(\mathbf{y}, \mathbf{v}') \leq 2t \leq d - 1$, and thus $d(\mathbf{v}, \mathbf{v}') \leq d - 1 < d$ which is a contradiction of d being the minimum distance between two distinct codewords. □

We can visualize a code as a collection of balls of radius t, centered around the codewords. Since the distance between any two codewords is at least d, all these balls do not intersect. Any vector that is obtained by adding a codeword with an error vector with weight at most t lies in exactly one of these balls and can be directly associated with the codeword in the center of the ball. Figure 2.1 illustrates a linear code $\mathcal{C}$ with minimum distance d. Non-intersecting balls of radius $t = \lfloor \frac{d-1}{2} \rfloor$ are drawn around three codewords $\mathbf{v}_1 \neq \mathbf{v}_2 \neq \mathbf{v}_3$ of $\mathcal{C}$. Error vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ of weight at most t are added to $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$. The words $\mathbf{y}_1 = \mathbf{v}_1 + \mathbf{e}_1,\ \mathbf{y}_2 = \mathbf{v}_2 + \mathbf{e}_2,\ \mathbf{y}_3 = \mathbf{v}_3 + \mathbf{e}_3$ remain in the ball of the respective codeword.

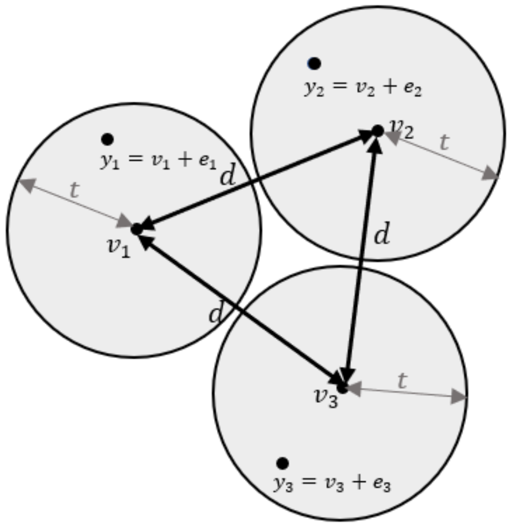

Figure 2.1: Unique decoding

Theorem 2.6 (Singleton bound). *Let $\mathcal{C}$ an $[n, k, d]$ linear code. Then*

$$d \leq n - k + 1.$$

Proof. [24, p. 33] □

Definition 2.7 (MDS Code). *An $[n, k, d]$ linear code $\mathcal{C}$ is said **MDS (Maximum Distance Separable)** if $d = n - k + 1$.*

Having large distance is important because ensures that the code can theoretically correct many errors. However, it is also important the existence of an efficient decoding algorithm.

The two most common ways to represent a code are either the representation by a generator matrix or by a parity-check matrix.

Definition 2.8 (Generator matrix). *An $k \times n$ matrix $\boldsymbol{G}$ is said to be a **generator matrix** of the code $\mathcal{C}$ if its rows form a basis of $\mathcal{C}$.*

A generator matrix $\mathbf{G}$ can be used to discribe the corresponding $[n, k]$ linear code $\mathcal{C}$ as

$$\mathcal{C} = \{\mathbf{v} = \mathbf{u}G \in \mathbb{F}_q^n \,|\, \mathbf{u} \in \mathbb{F}_q^k\}.$$

Remark 2.9. *Since the choice of basis is not unique, there are many generator matrices for a code. In fact, it is easy to see that given a generator matrix G and an invertible matrix S of size k, then SG generates the same code. If*

$G = [I_k \mid A]$, where I_k is the identity matrix and A a matrix $k \times (n-k)$, we say that G is in systematic form *and any code admits an unique generator matrix in systematic form.*

Before defining the parity-check matrix of a code, we introduce the *dual* of a code.

Definition 2.10 (Dual Code)**.** *Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. The **Dual code** of $\mathcal{C}$ is the $[n, n-k]$ code $\mathcal{C}^\perp$ defined by:*

$$\mathcal{C}^\perp = \{y \in \mathbb{F}_q^n \mid \forall x \in \mathcal{C}, xy^T = 0\}$$

Definition 2.11 (Parity-check matrix)**.** *The **parity-check matrix** of $\mathcal{C}$ is an $(n-k) \times n$ matrix such that $\boldsymbol{v} \in \mathcal{C}$ if and only if $H\boldsymbol{v}^T = 0$.*

Clearly an $[n,k]$ linear code $\mathcal{C}$ can also be described as

$$\mathcal{C} = \{\mathbf{v} \in \mathbb{F}_q^n \mid H\mathbf{v}^T = 0\}.$$

Remark 2.12. *From the previous definitions we have to $HG^T = 0$. Hence, a parity-check matrix of a code $\mathcal{C}$ is a generator matrix of its dual code.*

Remark 2.13. *If G is in systematic form, i.e., $G = [I_k \mid A]$, then H is of the form $[-A^T \mid I_{n-k}]$ where I_s is the identity matrix of size s.*

A parity-check matrix of a code $\mathcal{C}$ is very helpful to verify if a word belongs to $\mathcal{C}$ or not, computing the so-called *syndrome*, which is defined next.

Definition 2.14 (Syndrome)**.** *Let H be a parity-check matrix of an $[n,k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ and $\boldsymbol{y} \in \mathbb{F}_q^n$. The **syndrome** $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$ of $\boldsymbol{y}$ is given by:*

$$\boldsymbol{s} = \boldsymbol{y}H^T.$$

Remark 2.15. *Notice that if $\boldsymbol{y} \in \mathcal{C}$, i.e., if no errors occurred, the syndrome of $\boldsymbol{y}$ is the zero vector.*

The next definitions will be needed to analyse the security of the McEliece cryptosystem. In particular, they will be used in the Wieschebrink attack and the distinguisher-based attack, see Sections 3.3.2 and 3.3.3.

Definition 2.16 (Schur Product)**.** *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$. We define the **Schur product** of $\boldsymbol{x}$ and $\boldsymbol{y}$ as their componentwise product*

$$\boldsymbol{x} \star \boldsymbol{y} = (\boldsymbol{x}_1\boldsymbol{y}_1, \cdots, \boldsymbol{x}_n\boldsymbol{y}_n).$$

Definition 2.17 (Star Product of Codes and Square Code)**.** *Let $\mathcal{A}$, $\mathcal{B}$ be two codes of length n. The **Star Product Code** denoted by $< \mathcal{A} \star \mathcal{B} >$ of $\mathcal{A}$ and $\mathcal{B}$ is the vector space spanned by all $\boldsymbol{a} \star \boldsymbol{b}$ with $\boldsymbol{a} \in \mathcal{A}$ and $\boldsymbol{b} \in \mathcal{B}$:*

$$< \mathcal{A} \star \mathcal{B} >=< \boldsymbol{a} \star \boldsymbol{b} \mid \boldsymbol{a} \in \mathcal{A}, \boldsymbol{b} \in \mathcal{B} > .$$

*When $\mathcal{B} = \mathcal{A}$, $< \mathcal{A} \star \mathcal{A} >$ is called the **Square Code of** $\mathcal{A}$ and is denoted by $< \mathcal{A}^2 >$.*

Definition 2.18 (Schur Matrix)**.** *Let G be a $k \times n$ matrix, with rows r_i for $1 \leq i \leq k$. The **Schur Matrix**, denoted by $S(G)$, is a matrix with rows $r_i \star r_j$ for $i \leq i \leq j \leq k$.*

Remark 2.19. *It is easy to see that $S(G)$ is of the size $\frac{1}{2}(k^2 + k) \times n$. Note also that the Schur matrix of the generator matrix of a code is the generator matrix of the square code.*

The square code construction is very useful when we want to distinguish a random code from a structured code like a GRS code, as we will see in Section 3.3.3.

The following result can be found in [15, Prop. 4], for instance.

Proposition 2.20. *Let $\mathcal{A}$, $\mathcal{B}$ be two codes of length n. Then*

$$dim(< \mathcal{A} \star \mathcal{B} >) \leq dim(\mathcal{A}) dim(\mathcal{B}).$$

Proof. Suppose that $dim(\mathcal{A}) = k$ and $dim(\mathcal{B}) = k'$. Let $G_{\mathcal{A}}$ be the generator matrix of the code $\mathcal{A}$, with rows a_i, $1 \leq i \leq k$ and $G_{\mathcal{B}}$ be the generator matrix of the code $\mathcal{B}$, with rows b_j, $1 \leq j \leq k'$. The generator matrix of $< \mathcal{A} \star \mathcal{B} >$ has rows $a_i \star b_j$ with $1 \leq i \leq k$ and $1 \leq j \leq k'$. Thus the generator matrix of $< \mathcal{A} \star \mathcal{B} >$ is a $kk' \times n$ matrix and has at most rank

$$min\{n, kk'\} \leq kk' = dim(\mathcal{A}) dim(\mathcal{B}).$$

$\square$

Proposition 2.21. *Let $n : \mathbb{N} \longrightarrow \mathbb{N}$ be such that $n(k) \geq \frac{k(k+1)}{2}$ for all $k \in \mathbb{N}$ and define $s : \mathbb{N} \longrightarrow \mathbb{N}$ by $s(k) := n(k) - \frac{k(k+1)}{2}$. Then there exits a constant $\tilde{\delta} \in \mathbb{R}_{>0}$ such that, for all large enough k,*

$$Pr\left(\dim < \mathcal{A}^2 >= \frac{k(k+1)}{2}\right) \geq 1 - 2^{-\tilde{\delta} s(k)},$$

where $\mathcal{A}$ is chosen uniformly at random from $\mathcal{A}[n(k), k]$.

Proof. See [12] for details. $\square$

Remark 2.22. *Notice that the previous result implies that if one chooses a linear code at random, of dimension k and length n then the dimension of the square code is typically the minimum of the values n and $\frac{k(k+1)}{2}$.*

2.2 Generalized Reed-Solomon codes

Generalized Reed Solomon codes form a class of codes that was introduced for the first time in 1960 by Reed and Solomon in [53]. They represent an interesting family of codes with a wide range of applications as they are MDS codes and admit several efficient (algebraic) decoding algorithms. Next, we recall some of their important properties that will be used in this thesis, see [40] for more details.

2.2.1 Basic definitions and properties

Definition 2.23 (Generalized Reed-Solomon Code). *Let $\mathbb{F}_q$ be a finite field and $1 \leq k < n \leq q$ integers. Let $(\alpha, \beta) \in \mathbb{F}_q^n \times \mathbb{F}_q^n$, such that $\alpha = (\alpha_1, \cdots, \alpha_n)$ with $\alpha_i \neq \alpha_j, \forall i \neq j \in \{1, \cdots, n\}$ and $\beta = (\beta_1, \cdots, \beta_n)$ with $\beta_i \neq 0, \forall i \in \{1, \cdots, n\}$. The* ***Generalized Reed-Solomon code****, $GRS_{n,k}(\alpha, \beta)$, is given by:*

$$GRS_{n,k}(\alpha, \beta) = \{(\beta_1 f(\alpha_1), \cdots, \beta_n f(\alpha_n)) \mid f \in \mathbb{F}_q[x],\ \deg(f) < k\}.$$

It is easy to see that a generator matrix of $GRS_{n,k}(\alpha, x)$ is given by

$$G = G_{GRS} = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_n \\ \beta_1\alpha_1 & \beta_2\alpha_2 & \cdots & \beta_n\alpha_n \\ \vdots & & \ddots & \\ \beta_1\alpha_1^{k-1} & \beta_2\alpha_2^{k-1} & \cdots & \beta_n\alpha_n^{k-1} \end{bmatrix}$$

where $\alpha_1, \cdots, \alpha_n$ are called the code locators.

Remark 2.24. *Let $\boldsymbol{u} \in \mathbb{F}_q^k$, then*

$$\boldsymbol{u}G = \begin{bmatrix} \beta_1 \left(u_1 + u_2\alpha_1 + u_3\alpha_1^2 + \cdots u_k\alpha_1^{k-1}\right) \\ \beta_2 \left(u_1 + u_2\alpha_2 + u_3\alpha_2^2 + \cdots u_k\alpha_2^{k-1}\right) \\ \vdots \\ \beta_n \left(u_1 + u_2\alpha_n + u_3\alpha_n^2 + \cdots u_k\alpha_n^{k-1}\right) \end{bmatrix}^\top .$$

Therefore, the polynomial f corresponding to the codeword $\boldsymbol{v} = \boldsymbol{u}G$ is

$$f(x) = u_1 + u_2x + \cdots + u_kx^{k-1}.$$

Proposition 2.25. *GRS codes are MDS codes, i.e.,*

$$d(GRS_{n,k}(\alpha, \beta)) = n - k + 1.$$

Proof. Let v be a nonzero codeword of $GRS_{n,k}(\alpha, \beta)$. Let $f \in \mathbb{F}_q[x]$ be the polynomial corresponding to v. Since f has degree less than k, it has at most $k-1$ roots. Since $v = (\beta_1 f(\alpha_1), \cdots, \beta_n f(\alpha_n))$, with $\beta_i \neq 0$ for any $i \in \{1, \cdots, n\}$, a coordinate v_i of v can only be zero if $f(\alpha_i) = 0$. So the minimum weight of a nonzero codeword is $n - (k-1)$, i.e., is MDS. $\square$

Given an n-tuple $v = (\beta_1 f(\alpha_1), \cdots, \beta_n f(\alpha_n))$ it is possible to reconstruct the unique polynomial f of degree less than k associated to v using polynomial interpolation. To show that, we need to introduce some notation. Define

$$L(x) = \prod_{i=1}^{n} (x - \alpha_i)$$

and

$$L_i(x) = \prod_{j \neq i} (x - \alpha_j).$$

The polynomials $L(x)$ and $L_i(x)$ are monic of degrees n and $n-1$, respectively. Since the coordinate v_i of v is $\beta_i f(\alpha_i)$, i.e. $f(\alpha_i) = \frac{v_i}{\beta_i}$, hence, by Lagrange interpolation, the polynomial f can be recovered as

$$f(x) = \sum_{i=1}^{n} \frac{L_i(x)}{L_i(\alpha_i)} f(\alpha_i).$$

Proposition 2.26. *The dual of a GRS code is also a GRS code and we have*

$$GRS_{n,k}(\alpha, \beta)^{\perp} = GRS_{n,n-k}(\alpha, \beta')$$

for $\beta' = (\beta'_1, \ldots, \beta'_n) \in \mathbb{F}_q^n$, *where*

$$\beta'_i = \frac{1}{\beta_i L_i(\alpha_i)}. \tag{2.1}$$

Proof. Let f and g be the polynomials associated to the codes $GRS_{n,k}(\alpha, \beta)$ and $GRS_{n,n-k}(\alpha, \beta')$, respectively. If x is a codeword of $GRS_{n,k}(\alpha, \beta)$, then $x = (\beta_1 f(\alpha_1), \cdots, \beta_n f(\alpha_n))$, if y is a codeword of $GRS_{n,n-k}(\alpha, \beta')$, then $y = (\beta'_1 g(\alpha_1), \cdots, \beta'_n g(\alpha_n))$. We want to show that $xy^T = 0$. As $\deg(f) < k$ and $\deg(g) < n-k$, $\deg(fg) < n-1$. By Lagrange interpolation we can write

$$f(x)g(x) = \sum_{i=1}^{n} \frac{L_i(x)}{L_i(\alpha_i)} f(\alpha_i) g(\alpha_i).$$

Considering the coefficient of x^{n-1} from the two sides, we have

$$0 = \sum_{i=1}^{n} \frac{1}{L_i(\alpha_i)} f(\alpha_i) g(\alpha_i),$$

or equivalently,

$$\sum_{i=1}^{n} \beta_i f(\alpha_i) \frac{1}{\beta_i L_i(\alpha_i)} g(\alpha_i) = 0$$

As $\beta_i' = \frac{1}{\beta_i L_i(\alpha_i)}$ we have that $\sum_{i=1}^{n} \beta_i f(\alpha_i) \cdot \beta_i' g(\alpha_i) = 0$, *i.e.*, $xy^T = 0$ as required.

$\square$

It follows that the parity check matrix of a $GRS_{n,k}(\alpha, \beta)$ code is

$$H = \begin{pmatrix} \beta_1' & \beta_2' & \cdots & \beta_n' \\ \beta_1'\alpha_1 & \beta_2'\alpha_2 & \cdots & \beta_n'\alpha_n \\ \vdots & & \ddots & \\ \beta_1'\alpha_1^{n-k-1} & \beta_2'\alpha_2^{n-k-1} & \cdots & \beta_n'\alpha_n^{n-k-1} \end{pmatrix}$$

where $\beta_i' = \frac{1}{\beta_i L_i(\alpha_i)}$.

With the above notation, the following result can be found in [40].

Proposition 2.27.

$$GRS_{n,k}(\alpha, \beta) = GRS_{n,k}((a\alpha_1 + b, \ldots, a\alpha_n + b), (c\beta_1, \ldots, c\beta_n))$$

for all $a, b, c \in \mathbb{F}_q$ and $a, c \neq 0$.

Proof. See [40]. $\square$

We finish this introduction to the GRS codes with a result about the dimension of the square code, that can be found for example in [19].

Proposition 2.28. *Let $\mathcal{A}$, be a GRS code of length n and dimension k. If $2k - 1 \leq n$ then*

$$\dim(< \mathcal{A} \star \mathcal{A} >) = 2k - 1.$$

Proof. See for instance [41, Proposition 18]. $\square$

2.2.2 Encoding with GRS codes

Let $u = (u_0, u_1, \cdots, u_{k-1})$ be the message to be sent. We encode u by computing $v = u \cdot G_{GRS}$. So $v = (v_1, v_2, \cdots, v_n)$, with

$$v_i = \sum_{j=0}^{k-1} \beta_i u_j \alpha_i^j.$$

Alternately, we can see the encoding process as an evaluation of the polynomial

$$u(x) = u_0 + u_1 x + \cdots + u_{k-1} x^{k-1} \in \mathbb{F}_q[x]$$

in n different points $\alpha_1, \ldots, \alpha_n$. Hence, the encoding of $u(x)$ is

$$u(x) \mapsto (\beta_1 u(\alpha_1), \cdots, \beta_n u(\alpha_n)).$$

2.2.3 Decoding with GRS codes

A series of decoding algorithms bearing names such as Berlekamp-Massey, Peterson, Welch–Berlekamp, or using the Euclidean Algorithm, have been developed over the years for GRS codes. We briefly present a method based on the Euclidean decoding algorithm, following the ideas of [28, Chapter 5], and an algorithm for decoding the dual of $GRS_{n,r}(\alpha, u)$.

Let $y = v + e$, where $v = (v_1, v_2, \ldots, v_n)$ is the codeword transmitted, $y = (y_1, y_2, \ldots, y_n)$ is the received vector and $e = (e_1, e_2, \ldots, e_n)$ is the error vector. Let J the set of error locations:

$$J = \{i \mid e_i \neq 0\}.$$

The decoder will find the error set J and the error values e_i when the error correcting capability of the code is not exceeded, *i.e.*, when $\mid J \mid \leq \left\lfloor \frac{d-1}{2} \right\rfloor$.

The first step is to compute the syndrome, which is given by $S_y = y \cdot H^T$. We let $S_y = (S_1, \ldots, S_{n-k})$. Then, for every $1 \leq l \leq n-k$, we have

$$S_l = \sum_{j=1}^{n} y_j \beta'_j \alpha_j^{l-1}. \tag{2.2}$$

Definition 2.29. *The* ***syndrome polynomial****, denoted $S(x)$, is defined by*

$$S(x) = \sum_{l=0}^{n-k-1} S_{l+1} x^l. \tag{2.3}$$

Note that $S_y = y \cdot H^T = vH^T + eH^T = eH^T$, because $vH^T = 0$ since $v \in \mathcal{C}$. So, $S(y) = S(e)$, and

$$S_l = \sum_{j=1}^{n} e_j \beta'_j \alpha_j^{l-1} = \sum_{j \in J} e_j \beta'_j \alpha_j^{l-1}. \tag{2.4}$$

In order to find the set of error locations J and the corresponding error values $\{e_i \mid i \in J\}$, we define two polynomials as follows.

Definition 2.30. *Let*

$$\sigma(x) = \prod_{j \in J} (1 - \alpha_j x)$$

and

$$\omega(x) = \sum_{j \in J} e_j \beta'_j \left(\prod_{k \in J, k \neq j} (1 - \alpha_k x) \right).$$

The polynomial $\sigma(x)$ is called the error locator *polynomial, and $\omega(x)$ is the* error evaluator *polynomial. Note that $deg(\sigma) =\mid J \mid$ and $deg(\omega) \leq\mid J \mid -1$.*

Using the polynomials $\sigma(x)$ and $\omega(x)$ we can reconstruct the error vector e. We may assume that none of the α_i are equal to 0, but if $\alpha_i = 0$ then G has column i null, so $v_i = 0$ and then $e_i = y_i$. Thus,

$$J = \{b \mid \sigma(\alpha_b^{-1}) = 0\}. \tag{2.5}$$

Lemma 2.31. *The polynomials $\sigma(x)$ and $\omega(x)$ are relatively prime, and the error values e_b are given by*

$$e_b = \frac{-\alpha_b \omega(\alpha_b^{-1})}{\beta'_b \sigma'(\alpha_b^{-1})} \tag{2.6}$$

where $\sigma'(x)$ is the formal derivative of $\sigma(x)$.

Proof. In order to show that $\sigma(x)$ and $\omega(x)$ are relatively prime, it is enough to observe that they have no roots in common. In effect, if α_b^{-1} is a root of $\sigma(x)$, then $b \in J$. By the previous definition,

$$\begin{aligned}
\omega(\alpha_b^{-1}) &= \sum_{j \in J} e_j \beta'_j \left(\prod_{k \in J, k \neq j} (1 - \alpha_k \alpha_b^{-1}) \right) \\
&= \sum_{j \in J, j \neq b} e_j \beta'_j \left(\prod_{k \in J, k \neq j} (1 - \alpha_k \alpha_b^{-1}) \right) + e_b \beta'_b \left(\prod_{k \in J, k \neq j} (1 - \alpha_k \alpha_b^{-1}) \right) \\
&= e_b \beta'_b \prod_{k \in J, k \neq j} (1 - \alpha_k \alpha_b^{-1}) \\
&\neq 0.
\end{aligned}$$

The first sum is zero since $j \neq b$, $k \neq j$ and $b \in \{k \in J \mid k \neq j\}$. Hence, $\sigma(x)$ and $\omega(x)$ are relatively prime. In order to prove (2.6) notice that

$$e_b = \frac{\omega(\alpha_b^{-1})}{\beta'_b \prod_{k \in J, k \neq j} (1 - \alpha_k \alpha_b^{-1})}$$

and

$$\sigma'(x) = -\sum_{j\in J} \alpha_j \prod_{k\in J, k\neq j} (1-\alpha_k x).$$

So,

$$\begin{aligned}
\sigma'(\alpha_b^{-1}) &= -\sum_{j\in J} \alpha_j \prod_{k\in J, k\neq j} (1-\alpha_k \alpha_b^{-1}) \\
&= -\sum_{j\in J, j\neq b} \alpha_j \prod_{k\in J, k\neq j} (1-\alpha_k \alpha_b^{-1}) - \alpha_b \prod_{k\in J, k\neq b} (1-\alpha_k \alpha_b^{-1}) \\
&= -\alpha_b \prod_{k\in J, k\neq b} (1-\alpha_k \alpha_b^{-1}).
\end{aligned}$$

Therefore,

$$\begin{aligned}
e_b &= \frac{\omega(\alpha_b^{-1})}{\beta_b' \prod_{k\in J, k\neq b}(1-\alpha_k\alpha_b^{-1})} \\
&= \frac{-\alpha_b \omega(\alpha_b^{-1})}{\beta_b' \sigma'(\alpha_b^{-1})}.
\end{aligned}$$

□

The next theorem gives the so called *Key Equation* for decoding GRS codes, and it establishes a fundamental relationship between $\sigma(x)$, $\omega(x)$ and $S(x)$.

Theorem 2.32. *There is a polynomial $\mu(x)$ such that the error locator, the error evaluator and the syndrome polynomials verify the key equation:*

$$\sigma(x)S(x) = \omega(x) + \mu(x)x^{n-k}, \tag{2.7}$$

or, equivalently, using a congruence:

$$\sigma(x)S(x) \equiv \omega(x) \pmod{x^{n-k}}$$

Proof. By Definition 2.29 and equation (2.4), we have

$$\begin{aligned}
S(x) &= \sum_{l=0}^{n-k-1} \left(\sum_{j\in J} e_j \beta_j' \alpha_j^l \right) x^l \\
&= \sum_{j\in J} e_j \beta_j' \sum_{l=0}^{n-k-1} (\alpha_j x)^l \\
&= \sum_{j\in J} e_j \beta_j' \frac{1-(\alpha_j x)^{n-k}}{1-\alpha_j x}.
\end{aligned}$$

Multiplying both sides by $\sigma(x)$, where $\sigma(x)$ is given in Definition 2.30, we obtain

$$\begin{aligned}
\sigma(x)S(x) &= \sum_{j\in J} e_j\beta'_j \frac{1-(\alpha_j x)^{n-k}}{1-\alpha_j x} \prod_{m\in J}(1-\alpha_m x) \\
&= \sum_{j\in J} e_j\beta'_j \left(1-(\alpha_j x)^{n-k}\right) \prod_{m\in J, m\neq j} (1-\alpha_m x) \\
&= \sum_{j\in J} e_j\beta'_j \prod_{\substack{m\in J\\ m\neq j}}(1-\alpha_m x) - \sum_{j\in J} e_j\beta'_j(\alpha_j x)^{n-k} \prod_{\substack{m\in J\\ m\neq j}}(1-\alpha_m x) \\
&= \sum_{j\in J} e_j\beta'_j \prod_{\substack{m\in J\\ m\neq j}}(1-\alpha_m x) - \left(\sum_{j\in J} e_j\beta'_j\alpha_j^{n-k} \prod_{\substack{m\in J\\ m\neq j}}(1-\alpha_m x)\right) x^{n-k} \\
&= \omega(x) + \mu(x)x^{n-k}.
\end{aligned}$$

□

The next step in GRS decoding is to solve the key Equation (2.7) in order to find $\sigma(x)$ and $\omega(x)$. We will describe an algorithm based on Extended Euclidean Algorithm. There are many algorithms that solve this equation and this is not the most efficient, but this is simple and our aim is to show that GRS codes can be decoded in polynomial time.

Given two polynomials $a(x)$ and $b(x)$, the Extended Euclidean Algorithm provides a recursive procedure to find the greatest common divisor $c(x)$ of $a(x)$ and $b(x)$. This algorithm also finds two polynomials $s(x)$ and $t(x)$ such that

$$a(x)s(x) + b(x)t(x) = c(x).$$

Hence, in order to solve the key equation,

$$\mu(x)x^{n-k} + \sigma(x)S(x) = \omega(x), \tag{2.8}$$

we will compute $\gcd(S(x), x^{n-k})$, *i.e.*, $\omega(x)$. First, let us describe the Euclidean Algorithm for polynomials. Consider two polynomials a and b such that $\deg(a) \leq \deg(b)$. We start from the initial conditions $r_{-1} = a$ and $r_0 = b$.

We perform a recursion in steps $1, 2, \dots, m$. At step $n \leq m$ of the recursion, we obtain r_n as the residue of dividing r_{n-2} by r_{n-1}, *i.e.*, $r_{n-2} = q_n r_{n-1} + r_n$, where $\deg(r_n) < \deg(r_{n-1})$. The recursion is then given by

$$r_n = r_{n-2} - q_n r_{n-1}.$$

We also obtain values s_n and t_n such that $r_n = s_n a + t_n b$. Hence, the same recursion is valid for s_n and t_n as well:

$$s_n = s_{n-2} - q_n s_{n-1},$$

$$t_n = t_{n-2} - q_n t_{n-1}.$$

Since $r_{-1} = a = (1)a + (0)b$ and $r_0 = b = (0)a + (1)b$, we set the initial conditions $s_{-1} = 1$, $t_{-1} = 0$, $s_0 = 0$ and $t_0 = 1$.

By an application of the Euclidean Algorithm to x^{n-k} and $S(x)$, at a certain point of the recursion we obtain

$$r_i(x) = s_i(x)x^{n-k} + t_i(x)S(x),$$

where $\deg(r_i) \leq \lfloor \frac{n-k}{2} \rfloor - 1$, and i is the first with this property. Then, $\sigma(x) = t_i(x)$ and $\omega(x) = r_i(x)$.

We illustrate the decoding of GRS codes using the Extended Euclidean Algorithm with an example.

Example 2.33. *Consider the code $\mathcal{C} = GRS_{9,4}(\alpha, \beta)$ over $\mathbb{F}_{11}$, where*

$$\alpha = (2, 4, 1, 5, 3, 7, 9, 8, 6)$$

and

$$\beta = (1, 5, 3, 1, 2, 2, 1, 9, 4).$$

Suppose we want to send the message

$$u = (4, 1, 3, 2).$$

To encode it, we compute $v = uG$:

$$\begin{aligned} v &= \begin{pmatrix} 4 & 1 & 3 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 5 & 3 & 1 & 2 & 2 & 1 & 9 & 4 \\ 2 & 9 & 3 & 5 & 6 & 3 & 9 & 6 & 2 \\ 4 & 3 & 3 & 3 & 7 & 10 & 4 & 4 & 1 \\ 8 & 1 & 3 & 4 & 10 & 4 & 3 & 10 & 6 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 7 & 8 & 4 & 0 & 5 & 9 & 8 & 0 \end{pmatrix} \end{aligned}$$

The error correcting capability, t, *is* 2. *Considering that the error vector is $e = (0, 5, 0, 0, 0, 4, 0, 0)$, i.e., the errors are at the second and seventh positions. The message received is $y = v + e$,*

$$y = (1, 1, 8, 4, 0, 5, 2, 8, 0).$$

Suppose that the receiver wants to decode this message. First he calculates a vector β' for which $\mathcal{C}^{\perp} = GRS_{9,5}(\alpha, \beta')$: Using Equation (2.1) he obtains:

$$\beta' = (5, 7, 3, 3, 5, 5, 9, 3, 6).$$

Next he computes the syndrome polynomial, $S(x)$, using Equations (2.2) and (2.3):

$$S_y = (5, 2, 0, 5, 10).$$

Therefore,

$$S(x) = 5 + 2x + 5x^3 + 10x^4.$$

Now he applies the Extended Euclidean Algorithm with respect to $S(x)$ and x^5. When he finds the first n for which $\deg(r_n) \leq 1$ he stops and obtains the polynomials $\sigma(x)$ and $\omega(x)$. The process is described below.

n	r_n	q_n	$t_n = t_{n-2} - q_n t_{n-1}$
-1	x^5	-	0
0	$10x^4 + 5x_3 + 2x + 5$	-	1
1	$3x^3 + 2x^2 + 4x + 3$	$10x + 6$	$x + 5$
2	$4x + 3$	$7x + 8$	$4x^2 + x + 5$

Therefore he has obtained the error locator and evaluator polynomials $\sigma(x) = 4x^2 + x + 5$ and $\omega(x) = 4x + 3$. The error locations are given by (2.5), so, to find them he must compute the roots of $\sigma(x)$:

$$\begin{aligned} \sigma(x) = 0 &\Leftrightarrow 4x^2 + 4x + 5 = 0 \\ &\Leftrightarrow x = 3 \vee x = 5 \end{aligned}$$

Further, $3^{-1} = 4 = \alpha_2$ and $5^{-1} = 9 = \alpha_7$, so $J = \{2, 7\}$. The error value e_b is given by Equation (2.6) where $\sigma'(x) = 8x + 1$. Thus,

$$\begin{aligned} e_2 &= \frac{-4\omega(3)}{7\sigma'(3)} = 5 \\ e_7 &= \frac{-9\omega(5)}{9\sigma'(5)} = 4. \end{aligned}$$

Then, $e = (0, 5, 0, 0, 0, 0, 4, 0, 0)$ and $v = y - e = (1, 7, 8, 4, 0, 5, 9, 8, 0)$. To conclude the decoding process he computes u using $u = vG_{left}^{-1}$, where G_{left}^{-1} is the left inverse of G.

2.3 Goppa codes

In this section we briefly describe a class of linear error-correcting codes, the Goppa codes. These codes were introduced by V.D. Goppa in [27]. However, the original publication was in Russian and therefore we will refer to Berlekamp's summary in [8] for the definition of Goppa codes. Our interest in these codes stem from the fact that they were used in the first of the McEliece-type cryptosystems proposed and, surprisingly, they have resisted cryptanalysis for more than forty years, since no polynomial-time attack to the system has been devised up to now. Nevertheless, the improvement in computing power and algorithms to optimize the attacks have required an update of its original parameters (see [10]). Irreducible binary Goppa codes form an interesting subclass of these codes as they admit fast decoding algorithms. In fact, the irreducible binary Goppa codes were used in the original construction of the McEliece cryptosystem.

2.3.1 Basic definitions and properties

The existing literature about Goppa codes uses several different ways to introduce these codes. Here we will define them in terms of a Goppa polynomial $g(z) \in \mathbb{F}_{q^m}[z]$.

Definition 2.34 (Goppa codes)**.** *Let m and t be positive integers, $g(z) = \sum_{i=0}^{t} g_i z^i$, with $g_i \in \mathbb{F}_q$, a monic polynomial of degree t, called the **Goppa polynomial** and $\mathcal{L} = \{\alpha_1, \dots, \alpha_n\} \subset \mathbb{F}_q$ a tuple of n distinct elements, called the **support** of, such that $g(\alpha_i) \neq 0, \forall \alpha_i \in \mathcal{L}$. A **Goppa code** $\Gamma(\mathcal{L}, g)$ is defined to be the set of all codewords $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_n) \in \mathbb{F}_q^n$ such that*

$$\sum_{i=1}^{n} \frac{\mathbf{v}_i}{z - \alpha_i} \equiv 0 \mod g(z). \tag{2.9}$$

*If $g(z)$ is irreducible then the code is called an **irreducible Goppa code**.*

Goppa codes are linear codes and, as we can use the notation $[n, k, d]$ to describe a Goppa code with the parameters of length n, dimension k and minimum Hamming distance d.

Theorem 2.35. *[40] Let $\Gamma(\mathcal{L}, g)$ be a Goppa code and $deg(g(z)) = t$. Then, the dimension k and the minimum Hamming distance d satisfy the following properties:*

(i) $k \geq n - mt$,

(ii) $d \geq t + 1$.

We next describe the *Parity Check Matrix* of a Goppa code, see [40] for more details. Recall that a parity check matrix of a code $\mathcal{C}$ is defined to be a matrix H such that $H\mathbf{v}^T = 0$, $\forall \mathbf{v} \in \mathcal{C}$, and it is typically used to decode a message. In order to obtain a parity check matrix of a Goppa code we are going to transform the left side of Equation (2.9) into a polynomial expression. First note that $z - \alpha_i \mid g(z) - g(\alpha_i)$ and

$$\begin{aligned} g(z) - g(\alpha_i) &\equiv -g(\alpha_i) \mod g(z) \\ \frac{-1}{g(\alpha_i)} \frac{g(z) - g(\alpha_i)}{z - \alpha_i} &\equiv \frac{1}{z - \alpha_i} \mod g(z). \end{aligned}$$

Therefore, $\mathbf{v} \in \Gamma(\mathcal{L}, g)$ if and only if

$$-\sum_{i=1}^{n} \frac{\mathbf{v}_i}{g(\alpha_i)} \frac{g(z) - g(\alpha_i)}{z - \alpha_i} \equiv \sum_{i=1}^{n} \frac{\mathbf{v}_i}{z - \alpha_i} \equiv 0 \mod g(z). \tag{2.10}$$

We express $\frac{g(z)-g(\alpha_i)}{z-\alpha_i}$ as a polynomial. Note that we can write

$$g(z) - g(\alpha_i) = g_t z^t + g_{t-1} z^{t-1} + \ldots + g_1 z - \sum_{j=1}^{t} g_j \alpha_i^j = \sum_{j=1}^{t} g_j (z^j - \alpha_i^j.$$

Hence we obtain,

$$\begin{aligned} \frac{g(z) - g(\alpha_i)}{z - \alpha_i} &= g_t z^{t-1} + (g_{t-1} + g_t \alpha_i) z^{t-2} + \ldots + \\ &+ \left(g_1 + g_2 \alpha_i + \ldots + g_t \alpha_i^{t-1}\right) z^0 \\ &= \sum_{k=0}^{t-1} z^k \left(\sum_{j=k+1}^{t} g_j \alpha_i^{j-1-k} \right). \end{aligned}$$

Therefore, it follows that

$$\frac{g(z) - g(\alpha_i)}{z - \alpha_i} g(\alpha_i)^{-1} = g(\alpha_i)^{-1} \sum_{k=0}^{t-1} z^k \left(\sum_{j=k-1}^{t} g_j \alpha_i^{j-1-k} \right). \tag{2.11}$$

From (2.10) and (2.11) we obtain $\mathbf{v} \in \Gamma(\mathcal{L}, g)$ if and only if,

$$-\sum_{i=1}^{n} \mathbf{v_i} \frac{1}{g(\alpha_i)} \sum_{k=0}^{t-1} z^k \left(\sum_{j=k+1}^{t} g_j \alpha_i^{j-1-k} \right) \equiv 0 \mod g(z).$$

That is, $\mathbf{v} \in \Gamma(\mathcal{L}, g)$ if and only if

$$\sum_{i=1}^{n} \left(\frac{1}{g(\alpha_i)} \sum_{j=k+1}^{t} g_j \alpha_i^{j-1-k} \right) \mathbf{v}_i = 0.$$

Considering the fact that $\mathbf{v} \in \Gamma(\mathcal{L}, g)$ if and only if $H\mathbf{v}^T = 0$, we concluded that a parity check H of a Goppa code is a matrix $t \times n$ whose i-th column is:

$$\begin{pmatrix} g_t \\ g_{t-1} + g_t\alpha_i \\ g_{t-2} + g_{t-1}\alpha_i + g_t\alpha_i^2 \\ \vdots \\ g_1 + g_2\alpha_i + \cdots + g_t\alpha_i^{t-1} \end{pmatrix} g(\alpha_i)^{-1}.$$

Clearly, the parity check matrix H can be decomposed as $H = XYZ$, where

$$X = \begin{pmatrix} g_t & 0 & 0 & \dots & 0 \\ g_{t-1} & g_t & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \dots & g_t \end{pmatrix}, \; Y = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \dots & \alpha_n^{t-1} \end{pmatrix}$$

and

$$Z = \begin{pmatrix} g(\alpha_1)^{-1} & 0 & \dots & 0 \\ 0 & g(\alpha_2)^{-1} & 0 & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_n^{t-1} \end{pmatrix}.$$

As X is a invertible matrix $H' = X^{-1}H$ generates the same code as H. So we have a simpler write a parity check matrix H of a Goppa code in a simpler way:

$$H' = \begin{pmatrix} g(\alpha_1)^{-1} & g(\alpha_2)^{-1} & \dots & g(\alpha_n)^{-1} \\ \alpha_1 g(\alpha_1)^{-1} & \alpha_2 g(\alpha_2)^{-1} & \dots & \alpha_n g(\alpha_n)^{-1} \\ \vdots & & \ddots & \\ \alpha_1^{t-1} g(\alpha_1)^{-1} & \alpha_2^{t-1} g(\alpha_2)^{-1} & \dots & \alpha_n^{t-1} g(\alpha_n)^{-1} \end{pmatrix}.$$

2.3.2 Encoding and Decoding

To encoded a message we multiply the message by the generator matrix G of a Goppa code, the matrix $k \times n$ whose rows constitute a basis of the Goppa code and such that $GH^T = 0$. As for the decoding process, we remark that there exist several algebraic decoding algorithms tailor-made for Goppa codes. For instance one can find the decoding algorithm of Patterson [48] or the adaptation of the Berlekamp-Massey to Goppa codes [20]. We shall omit the details here, but one can also find variants of these algorithms. In [29] the author uses a modification of the Patterson algorithm including an optimization in the Euclidean algorithm. In [38] it is shown a way to easily decode classical Goppa codes assuming its representation in the setting of the Fourier transform.

2.3.3 Irreducible binary Goppa codes

The most important subclass of the Goppa codes is the irreducible binary Goppa codes, *i.e.*, the Goppa codes with $q = 2$ and with irreducible polynomial $g(z)$ in $\mathbb{F}_2[z]$. This subclass has some notable advantages over the general family of Goppa codes (and other kinds of codes as well). Not only it can be very efficiently implemented but, more importantly, it has a very good error-correction capability while maintaining a relatively high information rate. Note that all irreducible polynomials over a finite field are separable, *i.e.*, have no roots of multiplicity larger than one (see [30]).

Theorem 2.36. *For an irreducible binary $[n, k, d]$ Goppa code $\Gamma(\mathcal{L}, g)$ with $deg(g(z)) = t$, we have $d \geq 2t + 1$.*

Proof. Let $\mathbf{v} \in \Gamma(\mathcal{L}, g)$ be a nonzero codeword, $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_n)$ with $\mathbf{v}_i \in \{0, 1\}$, and $\mathcal{L} = \{\alpha_1, \dots, \alpha_n\}$. We have

$$\sum_{i=1}^{n} \frac{\mathbf{v}_i}{z - \alpha_i} \equiv 0 \mod g(z),$$

or equivalently,

$$\sum_{i=1}^{n} \mathbf{v}_i \frac{\prod\limits_{\substack{1 \leq k \leq n \\ k \neq i}} (z - \alpha_k)}{\prod\limits_{j=1}^{n} (z - \alpha_j)} \equiv 0 \mod g(z).$$

Let $\mathbf{v}_{i_1}, \dots, \mathbf{v}_{i_w}$ be the nonzero coordinates of $\mathbf{v}$. Since the code is binary, $\mathbf{v}_{i_j} = 1$, for $j \in \{1, \dots, w\}$, and we can write

$$\sum_{j=1}^{w} \frac{\prod\limits_{\substack{1 \leq k \leq w \\ k \neq j}} (z - \alpha_{i_k})}{\prod\limits_{\ell=1}^{w} (z - \alpha_{i_\ell})} \equiv 0 \mod g(z).$$

The denominator has no common factors with $g(z)$, since $g(\alpha_i) \neq 0$, for $i \in \{1, \ldots, n\}$. Therefore it must be a divisor of the numerator, *i.e.*, $g(z) \mid f(z)$, where

$$f(z) = \sum_{j=1}^{w} \prod_{\substack{1 \leq k \leq w \\ k \neq j}} (z - \alpha_{i_k}).$$

Moreover, note that the last expression is the derivative of the polynomial $\prod_{j=1}^{w}(z - \alpha_{i_j})$, and a binary derivative can only have terms with even exponents, thus

$$f(z) = f_0 + f_2 z^2 + \cdots + f_{2u} z^{2u}, \quad \text{with} \quad 2u \leq w - 1.$$

So, $g(z) \mid k(z)^2$, with $k(z)$ a polynomial of degree u and $2u \leq w - 1$. Since $g(z)$ is irreducible, $g(z) \mid k(z)$. Thus $t \leq u$ and $w - 1 \geq 2u \geq 2t$. For the minimum distance d it follows that $d = w \geq 2t + 1$. □

Theorem 2.37. *Let $\Gamma(\mathcal{L}, g(z))$ be a binary Goppa code with $g(z)$ irreducible, then $\Gamma(\mathcal{L}, g(z)) = \Gamma(\mathcal{L}, g^2(z))$.*

Proof. Notice that, since $\mathbb{F}_2$ is finite it follows that $g(z)$ has no roots of multiplicity larger than one. Following the argument and notation of the proof of the previous theorem:

$$\begin{aligned} \mathbf{v} \text{ is a codeword of } \Gamma(\mathcal{L}, g^2(z)) &\Leftrightarrow g^2(z) \mid f(z) \\ &\Leftrightarrow g(z) \mid f(z) \\ &\Leftrightarrow \mathbf{v} \text{ is a codeword in } \Gamma(\mathcal{L}, g(z)) \end{aligned}$$

□

This class of codes admit a particularly efficient decoding algorithm. We refer to [38] for a similar description of the decoding process as the one explained above for GRS.

2.4 Convolutional codes

Convolutional codes were first introduced by Elias in 1955 in [21]. Later they were formalized and further explained by Forney [25, 26], Piret [51] and McEliece [45], among others. In this setting the message is a sequence of messages instead of a single block message as occurred in the block code context. Convolutional codes differ from block codes in that the encoder contains memory and the n encoder outputs at any time unit depend not only on the k inputs but also on m previous input blocks. Thus memory is an important feature of an encoder of a convolutional code.

The use of convolutional codes in code-based cryptography is barely explored. They first appeared in variants of the McEliece cryptosystem in the work of Carl Löndahl and Thomas Johansson in [39], see also [46]. They proposed a new variant which replaces the generator matrix of the Goppa codes by a generator matrix of a block code with the classical Toeplitz structure of convolutional codes. However, as this variant resembled very much the one using block codes, the proposal was broken in a short time (see [35]). Nevertheless, the scheme presented several very appealing features. For instance, the fact that its secret generator matrix contains large parts that were generated completely at random. But the authors had already pointed out that this approach of using convolutional codes suffered from two main problems. The first one being the lack of efficient decoding algorithms, namely, that if one wants a maximum likelihood decoding, such as the Viterbi decoding algorithm, the memory used must be very limited. The second problem stems from the fact that convolutional codes usually start from the all zero state, and therefore the first code symbols that are generated will have low weight parity-checks, which would imply security threats.

In a very different way, we proposed in [1] a new variant of the McEliece cryptosystem where we use a convolutional encoder to be part of the public key. In this section we introduce the necessary background on convolutional codes that is needed to present and understand our variant. Unlike linear block codes, there exist several approaches defining convolutional codes. We shall present convolutional codes using the generator matrix approach. Before that, for a better understanding of the underlying idea we will begin by defining a convolutional encoder.

An (n, k) **convolutional encoder** is a linear device which maps a sequence of information $(u_0, u_1, \dots)$, $u_i \in \mathbb{F}_q^k$ into a sequence of codewords $(v_0, v_1, \dots)$, $v_i \in \mathbb{F}_q^n$ using an internal m-dimensional storage vector, x_i. In a convolutional encoder the i-th codeword v_i is a linear function not only of the i-th input word u_i, but also of the i-th state x_i. In a formal description, called the state space description of a convolutional code: $x_0 = 0$ and for $i \geq 0$

$$\begin{cases} x_{i+1} = x_i A + u_i B \\ \quad v_i = x_i C + u_i D \end{cases} \tag{2.12}$$

where $A_{m\times m}$, $B_{k\times m}$, $C_{m\times n}$ and $D_{k\times n}$ are matrices with entries from a field $\mathbb{F}_q$, q being a prime power. The integer m is called the degree of the encoder. In the case of a convolutional encoder we have four matrices while in block codes we have one, the matrix G. A block code is simply a degree 0 convolutional code, see [45] for a more detailed description.

Let's look at one simple example of a $(2,1)$ convolutional code over $\mathbb{F}_2$:

Example 2.38. *Let* $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 0 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, $D = \begin{pmatrix} 1 & 1 \end{pmatrix}$. *Suppose we want to encode the sequence* $u_0 = 1$, $u_1 = 0$, $u_2 = 1$, $u_3 = 0$, $u_4 = 1$ *and the state* x_i *is zero at time instant* $t = 0$, *i.e.,* $x_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}$.
Then,

$$\begin{cases} x_1 = \begin{pmatrix} 0 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \end{pmatrix} & = \begin{pmatrix} 1 & 0 \end{pmatrix} \\ v_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \end{pmatrix} & = \begin{pmatrix} 1 & 1 \end{pmatrix} \end{cases}$$

i.e., the convolutional encoder receives 1 *and encodes it into the vector* $\begin{pmatrix} 1 & 1 \end{pmatrix}$ *at time instant zero. At time instant* $t = 1$, $u_1 = 0$ *enters as an input in the convolutional encoder to produce* v_1 *as follows:*

$$\begin{cases} x_2 = \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \end{pmatrix} & = \begin{pmatrix} 0 & 1 \end{pmatrix} \\ v_1 = \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 \end{pmatrix}\begin{pmatrix} 1 & 1 \end{pmatrix} & = \begin{pmatrix} 1 & 0 \end{pmatrix} \end{cases}$$

and so on. Hence the sequence $1, 0, 1, 0, 1$ *is encoded to the sequence*

$$\begin{pmatrix} 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \end{pmatrix}, \ldots.$$

A physical realization of such an encoder can be implemented in terms of the so called shift-registers, for more details see [45]. It is easy to verify that in this example corresponding shift-register is:

To illustrate the dynamics of the encoding process of the Example 2.38 in the first two instants see figures 2.3 and 2.4.

For the purpose of this thesis we next consider convolutional codes as a set of generating functions. To this end we represent the information sequence $u_0, u_1, \ldots$ as a polynomial $u(D) = u_0 + u_1 D + \ldots$ where D is called the delay operator, to indicate the time instant in which each information arrived or each codeword was transmitted. Analogously, the codeword is a polynomial $v(D) = v_0 + v_1 D + \cdots$. So, we have:

$$X(D) = \sum_{i \geq 0} x_i D^i, \quad u(D) = \sum_{i \geq 0} u_i D^i \quad \text{and} \quad v(D) = \sum_{i \geq 0} v_i D^i.$$

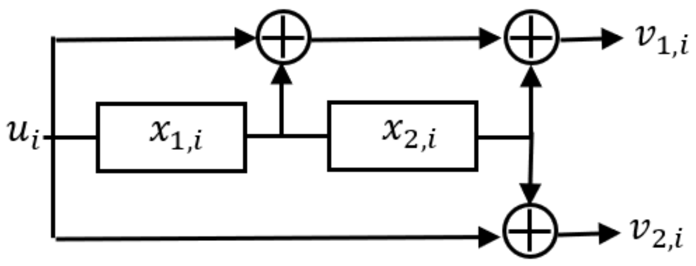

Figure 2.2: Shift register implementation of the convolutional encoder described in Example 2.38.

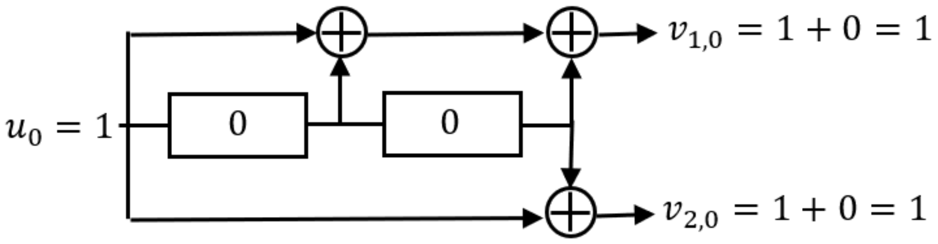

Figure 2.3: Encryption process at time instant $t = 0$

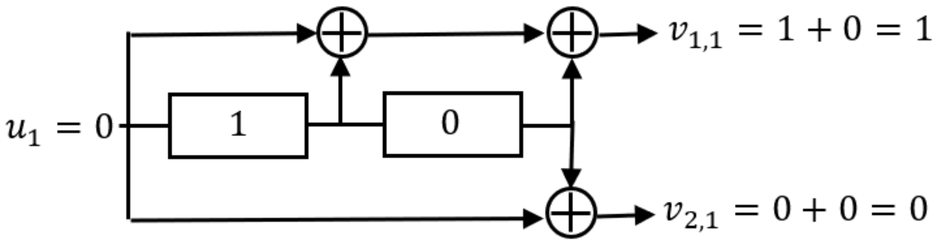

Figure 2.4: Encryption process at time instant $t = 1$

To transform the convolutional encoder described in (2.12) we multiply both sides of (2.12) by D^i and sum over all i (using the fact u_i, x_i, v_i are all zero for $i < 0$), and obtain

$$\begin{cases} X(D) \cdot D^{-1} = X(D) \cdot A + u(D) \cdot B \\ \qquad\quad v(D) = X(D) \cdot C + u(D) \cdot D \end{cases} \tag{2.13}$$

If we solve the system (2.13) to get an explicit expression for $X(D)$ and $v(D)$

in terms of the input $u(D)$ we obtain:

$$\begin{cases} X(D) = u(D) \cdot B \cdot (D^{-1}I_m - A)^{-1} \\ v(D) = u(D) \cdot [B \cdot (D^{-1}I_m - A)^{-1} \cdot C + D] \end{cases}$$

where I_m denotes the $m \times m$ identity matrix. We have concluded that

$$v(D) = u(D)G(D)$$

where $G(D)$ is a generator matrix defined by

$$G(D) = B \cdot (D^{-1}I_m - A)^{-1} \cdot C + D.$$

Example 2.39. *Considering the Example* (2.38)*,*

$$\begin{aligned} G(D) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \left(\begin{pmatrix} D^{-1} & 0 \\ 0 & D^{-1} \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \right)^{-1} \cdot \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 + D + D^2 & 1 + D^2 \end{pmatrix}. \end{aligned}$$

We shall allow the transmission to "start" at any time instant and therefore consider the infinite sequence as a Laurent polynomial

$$u(D, D^{-1}) = \sum_{i > \lambda} u_i D^i \in \mathbb{F}((D, D^{-1})), \quad \lambda \in \mathbb{Z}.$$

Based on this representation, we have the following definition of a convolutional code:

Definition 2.40. *[45, Definition 2.3] A* ***convolutional code*** *$\mathcal{C}$ of rate k/n is an $\mathbb{F}((D))$-subspace of $\mathbb{F}((D))^n$ of dimension k given by a rational* encoder matrix $G(D) \in \mathbb{F}(D)^{k \times n}$*,*

$$\begin{aligned} \mathcal{C} &= \operatorname{Im}_{\mathbb{F}((D))} G(D) \\ &= \left\{ u(D, D^{-1})G(D) \mid u(D, D^{-1}) \in \mathbb{F}^k((D)) \right\}, \end{aligned}$$

where $u(D, D^{-1}) = \sum_{i \geq \lambda} u_i D^i$ is called the information vector. If

$$G(D) = \sum_{i=0}^{m} G_i D^i \in \mathbb{F}^{k \times n}[D]$$

is polynomial, m is called the memory *of $G(D)$, since it needs to "remember" the inputs u_i from m units in the past. Note than when $m = 0$ the encoder is constant and generates the class of linear block codes in a natural way. A dual description of a convolutional code can be given through one of its parity-check matrices which are $(n-k) \times n$ full rank rational matrices $H(D)$ such that*

$$\begin{aligned} \mathcal{C} &= \ker_{\mathbb{F}((D))} H(D) \\ &= \left\{ v(D) \in \mathbb{F}^n((D)) \mid H(D)v(D) = 0 \in \mathbb{F}^{n-k}((D)) \right\}. \end{aligned}$$

Chapter 3

Block code Cryptography

Code-based cryptosystems, namely the McEliece cryptosystem is one promising family of public key cryptosystems which are potentially secure against the threat of quantum computing. Firstly, the problem of decoding a random public code with general decoding algorithms is a NP-hard problem and any attack based on a algorithm for decoding random linear codes, namely the Information-Set Decoding (ISD), can be protected by increasing the size of the code. Secondly, it is very difficulty for an attacker to uncover the underlying structure of the code used in the public key. The choice of the code used in the secret key is very important. The McEliece cryptosystem in its original version used binary irreducible Goppa codes and a generator matrix for it (the secret key) is masked by multiplying it on the right by a permutation matrix and multiplying it on the left by an invertible matrix. Goppa codes are also used in the Niederreiter cryptosystem, but instead of using the generator matrix, it is uses the parity check matrix. According to McEliece, they form a large family, providing a vast number of potential public keys, and there exists an efficient algorithm to decode them. However this cryptosystem has a big disadvantage, the key size is too large. For this reason, many variants replacing the Goppa code by other families of codes have been proposed. Many of these proposals focus on Generalised Reed Solomon (GRS) codes, since they benefit from excellent decoding properties which translates into small keys. On the other hand, their structure is difficult to hide.

In this chapter, we will analyze the three main such proposals. The use of GRS codes to replace Goppa codes in the McEliece cryptosystem was initially suggested by Niederreiter in [47], but this proposal was attacked by Sidelnikov and Shestakov [56].

Berger and Loidreau suggested in [7] a new variant that was resistant to the Sidelnikov and Shestakov attack. The idea was to replace the GRS code by subcodes of GRS but again a structural attack for this proposal was presented by Christian Wieschebrink in [63].

Baldi, Bianchi, Chiaraluce, Rosenthal and Schipani proposed in [6] a new way to mask the structure of the underlying secret GRS code by replacing the original permutation matrix by a denser transformation matrix. This proposal was attacked by Valérie Gauthier, Ayoub Otmani and Jean-Pierre Tillich in [19]. In spite of all these proposals having been broken (for the more interesting parameters), GRS codes are still of the highest interest for cryptography as its inclusion in the McEliece cryptosystem would reduce significantly the key size.

In this chapter, we start with a description of the original proposal of the McEliece cryptosystem, then we will investigate these three variants of that cryptosystem which use the GRS codes and, finally, we present Information Set Decoding (ISD) attacks. All these results are fundamental to understand the security and advantages of our cryptosystem, presented in the next chapter.

3.1 McEliece Cryptosystem

The first application of coding theory in a cryptographic context was presented by Robert McEliece in 1978 [44], which was called the McEliece Cryptosystem. This public-key cryptosystem is based on the hardness of decoding a message with random errors. The original version uses binary Goppa codes. We can describe the McEliece cryptosystem by means of three algorithms: a key generation algorithm which produces a public and a private key, an encryption algorithm and a decryption algorithm.

Key generation

Let $\mathcal{C}$ be a $[n, k, 2t+1]$ linear code that has an efficient decoding algorithm $\mathcal{D}$ that can correct up to t errors.

- Compute a $k \times n$ generator matrix G for $\mathcal{C}$;
- Generate a random $k \times k$ invertible matrix S;
- Generate a random $n \times n$ permutation matrix P;
- Compute the $k \times n$ matrix $G' = SGP$.

The ***public key*** is (G', t) and the ***private key*** is (S, G, P).

Encryption

To encrypt a message $\mathbf{u}$ of lenght k:

- Compute $\mathbf{u}G'$ and add a random error vector $\mathbf{e}$ of weight t and length n;
- Send $\mathbf{y} = \mathbf{u}G' + \mathbf{e}$.

Decryption

To decrypt $\mathbf{y}$:

- Compute $\mathbf{y}P^{-1} = (\mathbf{u}S)G + \mathbf{e}P^{-1}$;
- Since $(\mathbf{u}S)G$ is a codeword in $\mathcal{C}$ and the permuted error vector $\mathbf{e}P^{-1}$ has weight t, the decoding algorithm $\mathcal{D}$ can be applied to $\mathbf{y}P^{-1}$ to obtain $\mathbf{u}S$;
- Calculate $\mathbf{u}$ using S^{-1}.

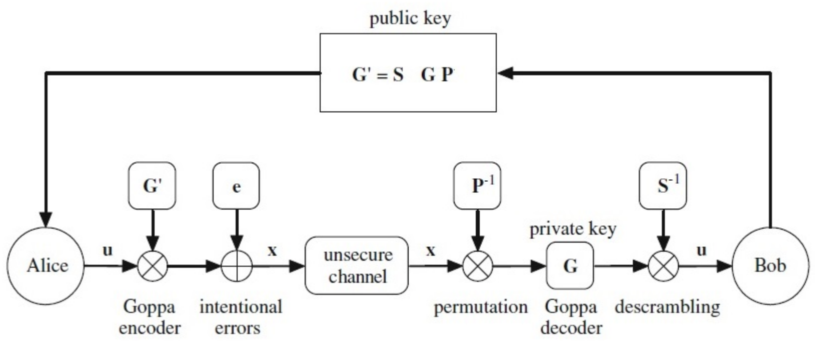

Figure 3.1: The original McEliece cryptosystem

This cryptosystem has the advantage that there exists a fast algorithm to encrypt and decrypt the message (compared, for instance, with the RSA) and no effective quantum algorithm is known to break McEliece cryptosystem. However, it has the disadvantage of having large keys and low transmission rate. For that, these cryptosystem have been barely used in practice. Several variants of these proposals focus on the inclusion of GRS codes as it would reduce the key. However, these variant suffer from structural attacks, *i.e.*, attacks aiming to discover the underlying secret GRS structure of the public key.

3.2 Niederreiter cryptosystem using the parity check matrix

Niederreiter proposed in 1986 [47] a cryptosystem which is similar to the McEliece system, but using the parity check matrix instead of the generator matrix and syndrome decoding for decryption. The Niederreiter cryptosystem works as follows.

Key generation
Let $\mathcal{C}$ be a $[n, k, 2t+1]$ linear code that has an efficient decoding algorithm that can correct up to t errors.

- Compute a $(n-k) \times n$ parity matrix H for $\mathcal{C}$;
- Generate a random $k \times k$ invertible matrix S;
- Generate a random $n \times n$ permutation matrix P;
- Compute the $k \times n$ matrix $H' = SHP$.

The ***public key*** is (H', t) and the ***private key*** is (S, H, P).

Encryption
To encrypt a message m of lenght k:

- Compute $m(H')^T$ and add a random error vector e of weight t and length n;
- Send $m' = m(H')^T + e$.

Decryption To decrypt m':

$$S^{-1}m^T = HPu^T = H(uP^T)^T.$$

Since $wt(uP^T) \leq t$, we can apply syndrome decoding to get uP^T, and by multiplying the inverse of P^T we get the message u. The security of the original proposals of McEliece and the Niederreiter cryptosystem is equivalent as are their computational costs. An attacker who can break one is able to break the other [64]. The Niederreiter cryptosystem can, in certain situations, have some advantages. Using the systematic form of the public key, the public key in the Niederreiter system is then $n/(n-k)$ times smaller than in the McEliece version (since the public key in the Niederreiter system has $(n-k) \times k$ bits and in the McEliece system it has $n \times k$). The systematic form of the public matrix H and the low-weight of vector m significantly reduce the computational cost involved in the encryption in Niederreiter's version.

3.3 Structural Attacks for variants of McEliece cryptosystems using GRS codes

There are two main classes of attacks to the McEliece cryptosystem: key-recovery attacks and general attacks. It was McEliece himself that suggested this classification in the final section of the original paper. The first class contains attacks directed at the private key of the cryptosystem. Since the aim is to reconstruct the private key, exploiting the particular structure of the code, they are known as *structural attacks*. These attack obviously heavily depend on the type of the code that is being use. The other class of attacks aim to decipher the ciphertext in order to obtain the plaintext directly from the public key without using any particular structure of the code. The main subclass of these attack are the so-called Information Set Decoding (ISD) attacks. In the next two sections we describe these two classes in detail and will establish the basis for the cryptanalysis of our proposal in the next chapter. We start this section with the structural attacks. In particular we focus on variants of the McEliece cryptosystem that use GRS codes as that is what we need for our proposal.

3.3.1 Sidelnikov-Shestakov attack to the Niederreiter variant

Niederreiter was the first to propose a cryptosystem based on the GRS codes [47]. However, six years after the article was published, Sidelnikov and Shestakov in [56] proposed a polynomial time attack against this variant.

Let's look at the general ideia of the description this attack, which we can find in [63].

Let $SG_{\alpha,\beta}P$ be the public key, where P is a permutation matrix of order n, S is an invertible matrix of order k and $G_{\alpha,\beta}$ is a generator matrix of the code $GRS_{n,k}(\alpha,\beta)$ (see Definition 2.23). $G_{\alpha,\beta}P$ is a column permutation of $G_{\alpha,\beta}$, so $G_{\alpha,\beta}P = \tilde{G}_{\tilde{\alpha},\tilde{\beta}}$, with $\tilde{\alpha} = (\tilde{\alpha}_1,\dots,\tilde{\alpha}_n)$, $\tilde{\beta} = (\tilde{\beta}_1,\dots,\tilde{\beta}_n)$ and for each i there exist one and only one j such that $\tilde{\alpha}_i = \alpha_j$ and $\tilde{\beta}_i = \beta_j$. Since $\tilde{G}_{\tilde{\alpha},\tilde{\beta}}$ is the generator matrix of a GRS code, we have

$$\begin{pmatrix} \tilde{\beta}_1 & \tilde{\beta}_2 & \dots & \tilde{\beta}_n \\ \tilde{\beta}_1\tilde{\alpha}_1 & \tilde{\beta}_2\tilde{\alpha}_2 & \dots & \tilde{\beta}_n\tilde{\alpha}_n \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\beta}_1\tilde{\alpha}_1{}^{k-1} & \tilde{\beta}_2\tilde{\alpha}_2{}^{k-1} & \dots & \tilde{\beta}_n\tilde{\alpha}_n{}^{k-1} \end{pmatrix}.$$

Let $M = S\tilde{G}_{\tilde{\alpha},\tilde{\beta}}$,where S is a $k \times k$ non-singular matrix. In the first step, $\tilde{\alpha}$ is reconstructed. Compute the echelon form $E(M)$ of M:

$$E(M) = \begin{pmatrix} 1 & 0 & \dots & 0 & b_{1,k+1} \dots & b_{1,n} \\ 0 & 1 & \dots & 0 & b_{2,k+1} \dots & b_{2,n} \\ & & \ddots & & \vdots & \vdots \\ 0 & \dots & 0 & 1 & b_{k,k+1} \dots & b_{k,n} \end{pmatrix}.$$

Consider the i-th row b_i of $E(M)$:

$$b_i = (0 \dots 0 \quad 1 \quad 0 \dots 0 \quad b_{i,k+1} \dots b_{i,n}).$$

Let f_{b_i} be the associated polynomial to b_i (see Remark 2.24). We know that f_{b_i} has $k-1$ roots and $f_{b_i}(\tilde{\alpha_j}) = 0$, for all $j \leq k$ with $j \neq i$. So f_{b_i} is a polynomial of degree $k-1$ and has the form

$$f_{b_i}(x) = c_{b_i} \cdot \prod_{j=1, j\neq i}^{k} (x - \tilde{\alpha}_j) \tag{3.1}$$

with $c_{b_i} \in \mathbb{F} \setminus \{0\}$. Now pick two arbitrary rows of $E(M)$, for example b_1 and b_2, and divide the entries of the first row by the corresponding entries in the second row as long as these are different from zero. Using (3.1) we get

$$\frac{b_{1,j}}{b_{2,j}} = \frac{\tilde{\beta}_j \cdot f_{b_1}(\tilde{\alpha}_j)}{\tilde{\beta}_j \cdot f_{b_2}(\tilde{\alpha}_j)} = \frac{c_{b_1} \prod_{l=1,l\neq 1}^{k} (\tilde{\alpha}_l - \tilde{\alpha}_j)}{c_{b_2} \prod_{l=1,l\neq 2}^{k} (\tilde{\alpha}_l - \tilde{\alpha}_j)} = d \cdot \frac{\tilde{\alpha}_j - \tilde{\alpha}_2}{\tilde{\alpha}_j - \tilde{\alpha}_1} \tag{3.2}$$

for $j = k+1, \dots, n$. Using Proposition 2.27, we can assume that $\tilde{\alpha}_1 = 0$ and $\tilde{\alpha}_2 = 1$. Since the $\frac{b_{1,j}}{b_{2,j}}$ are known, the $\tilde{\alpha}_j$ can uniquely be reconstructed from (3.2) through attempts at the correct value of d.

$$\tilde{\alpha_j} = \frac{b_{1,j}\tilde{\alpha}_1 - d\tilde{\alpha}_2 b_{2,j}}{b_{1,j} - d b_{2,j}}$$

It remains to find $\tilde{\alpha}_3, \dots, \tilde{\alpha}_k$. Therefore, we replace the row b_2 by b_i in 3.2, where $i \in \{3, \dots, k\}$, and get

$$\tilde{\alpha_j} = \frac{b_{1,j}\tilde{\alpha}_1 - d\tilde{\alpha}_i b_{i,j}}{b_{1,j} - d b_{i,j}}$$

Note that d and $\tilde{\alpha}_i$ are unknown, but letting $j = k+1$ and $j = k+2$, for example, those values can uniquely be reconstructed by solving a system of two linear equations.

Now, in a second step, $\tilde{\beta}$ and the matrix S can be recovered: Let $M'_{k\times(k+1)}$ be the matrix consisting of the first $k+1$ columns of the public key M. Firstly, find a non-trivial solution $c = (c_1, \ldots, c_{k+1})$ of the linear system

$$M'c = 0.$$

Let $\tilde{G}'_{k\times k+1}$ be obtained from $\tilde{G}$ by considering its first $k+1$ columns,

$$\begin{aligned}\tilde{G}' &= \begin{pmatrix} \tilde{\beta}_1 & \tilde{\beta}_2 & \ldots & \tilde{\beta}_{k+1} \\ \tilde{\beta}_1\tilde{\alpha}_1 & \tilde{\beta}_2\tilde{\alpha}_2 & \ldots & \tilde{\beta}_{k+1}\tilde{\alpha}_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\beta}_1\tilde{\alpha}_1^{k-1} & \tilde{\beta}_2\tilde{\alpha}_2^{k-1} & \ldots & \tilde{\beta}_{k+1}\tilde{\alpha}_{k+1}^{k-1} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \tilde{\alpha}_1 & \tilde{\alpha}_2 & \ldots & \tilde{\alpha}_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\alpha}_1^{k-1} & \tilde{\alpha}_2^{k-1} & \ldots & \tilde{\alpha}_{k+1}^{k-1} \end{pmatrix} \begin{pmatrix} \tilde{\beta}_1 \\ \tilde{\beta}_2 \\ \vdots \\ \tilde{\beta}_{k+1} \end{pmatrix}.\end{aligned}$$

Since $M' = S\tilde{G}'$, if $M'c = 0$ then $\tilde{G}'c = 0$. Therefore,

$$\begin{pmatrix} c_1 & c_2 & \ldots & c_{k+1} \\ c_1\tilde{\alpha}_1 & c_2\tilde{\alpha}_2 & \ldots & c_{k+1}\tilde{\alpha}_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ c_1\tilde{\alpha}_1^{k-1} & c_2\tilde{\alpha}_2^{k-1} & \ldots & c_{k+1}\tilde{\alpha}_{k+1}^{k-1} \end{pmatrix} \begin{pmatrix} \tilde{\beta}_1 \\ \tilde{\beta}_2 \\ \vdots \\ \tilde{\beta}_{k+1} \end{pmatrix} = 0.$$

We have a system of k equations and $k+1$ unknowns, so we can consider, for example, $\tilde{\beta}_1 = 1$ and compute the unique solution. Hence, we have the matrix $\tilde{G}'$ completely determined. Let $\tilde{G}''$ be the matrix consisting of the first k columns of $\tilde{G}'$ and M'' the matrix consisting of the first k columns of M. We have $S = M''(\tilde{G}'')^{-1}$. So we can obtain $\tilde{G}$ by computing $\tilde{G} = S^{-1}M$. Notice that this attack works if $2 \leq k \leq n-2$.

Next, we will see two other variants of the McEliece cryptosystem that uses GRS codes. We will study their structural attacks to better understand the modifications introduced in our proposal of cryptosystem.

3.3.2 Wieschebrink attacks on the BL variant

The idea of using GRS codes was reconsidered more than ten years later by Berger and Loidreau. In 2005, Berger and Loidreau [49] proposed a variant of the Niederreiter scheme which resists the Sidelnikov-Shestakov attack, using subcodes of GRS of small codimension to mask the structure. Unfortunately this technique was also successfully attacked in two steps by Wieschebrink [62], [63].

In the Berger-Loidreau variant (BL variant), the plaintext has length $k-\ell$, S is a $(k-\ell)\times k$ random matrix of rank $k-\ell$ and the public key is $G' = SGP$, where G is a $k\times n$ generator matrix of a GRS code. Note that the Sidelnikov-Shestakov attack cannot be directly applied to this scheme.

Wieschebrink presents two attacks to the BL variant. One presented in [62] can be considered as an extension of the Sidelnikov - Shestakov attack. But this attack is only feasible for small values of q and ℓ. In fact, it becomes infeasible if $q \geq 64$ and $\ell \geq 8$. For this reason we will not describe it, as it is not relevant for the main goals of this dissertation. The second attack was presented in [63]. Here, Wieschebrink presents a structural attack using the componentwise product. In what follows, we will see the general ideas of this powerful second attack that has been also applied to many other variants.

Let us consider $\tilde{G}_{\tilde{\alpha},\tilde{\beta}} = G_{\alpha,\beta}P$ as defined above, and set $M = G' = S\tilde{G}_{\tilde{\alpha}\tilde{\beta}} = [m_{ij}]$ the public matrix of the Berger-Loidreau cryptosystem. Note that the public key gives rise to a subcode of GRS of large dimension and then a generic attack to GRS cannot be applied (see [62] for details). It is easy to see that $m_{ij} = \tilde{\beta}_j f_i(\tilde{\alpha}_j)$ with $f_i(\tilde{\alpha}_j) = \sum_{t=1}^{k} s_{it}\tilde{\alpha}_j^{t-1}$, $i = 1,\cdots,k-\ell$ and $j = 1,\cdots n$.

Let b_i the i-th row of M, $i = 1,\cdots,k-\ell$ and f_i the associated polynomial to b_i. The attack is divided into two cases: $2k-1 \leq n-2$ and $2k-1 > n-2$.

Suppose, $2k-1 \leq n-2$. We calculate the component-wise product between any two rows, b_i and b_j of M, $i,j \in \{1,\cdots,k-\ell\}, i \leq j$:

$$b_i \star b_j = (\tilde{\beta}_1^2 f_i(\tilde{\alpha}_1)\cdot f_j(\tilde{\alpha}_1),\cdots,\tilde{\beta}_n^2 f_i(\tilde{\alpha}_n)\cdot f_j(\tilde{\alpha}_n)).$$

Since $\deg(f_i) \leq k-1$ and $\deg(f_j) \leq k-1$, we have $\deg(f_i\cdot f_j) \leq 2k-2$. Therefore, the code $\mathcal{C}$ generated by $b_i \star b_j$ is a subcode of $GRS_{n,2k-1}(\tilde{\alpha},\tilde{\beta}')$ where $\tilde{\beta}' = \tilde{\beta}\star\tilde{\beta}$. For ℓ small, more specifically, for

$$\frac{k-\ell(k-\ell+1)}{2} > 2k-1, \tag{3.3}$$

we have $\mathcal{C} = GRS_{n,2k-1}(\tilde{\alpha},\tilde{\beta}')$, where $\tilde{\beta}' = \tilde{\beta}\star\tilde{\beta}$, and therefore the Sidelnikov-Shestakov attack can be applied to discover the parameters $\tilde{\alpha}$ and $\tilde{\beta}'$.

If ℓ is so large that equation (3.3) is not satisfied, it is possible to apply the attack described in [62]. However this case is unlikely to occur (see [63, Section 6], for details).

Suppose now that $2k - 1 > n - 2$. In this case a similar attack could be applied to a shortened code. We will explain the details below. First we define this type of codes.

Definition 3.1 (d-shortened Code)**.** *Let $\mathcal{C}$ an $[n, k]$ linear code and $d \in \mathbb{N}$, $d \leq k$. The **d-shortened code** $S_d(\mathcal{C})$ consists of all codewords $(s_1, \cdots, s_{n-d})$ with $s_i \in \mathbb{F}_q$, for $1 \leq i \leq n - d$, such that*

$$(0, \cdots, 0, s_1, \cdots, s_{n-d}) \in \mathcal{C}.$$

Suppose that the generator matrix $G = [I_k \mid T]$ of $\mathcal{C}$ is in systematic form, where T denotes a $k \times (n - k)$ matrix. Then, a basis of $S_d(\mathcal{C})$ can be easily obtained by extracting the $n-d$ rightmost components of the last $k-d$ rows of G.

Let M be again the public key of the Berger-Loidreau cryptosystem, denote by $\mathcal{C}_M$ the code generated by the rows of M and by G_{S_d} the generator matrix of $S_d(\mathcal{C}_M)$.

A row $s = (s_1, \cdots, s_{n-d})$ of G_{S_d} can be written

$$s = (\tilde{\beta}_{d+1} h(\tilde{\alpha}_{d+1}), \cdots, \tilde{\beta}_n h(\tilde{\alpha}_n))$$

with $h(x) = g(x) \prod_{j=1}^{d} (x - \tilde{\alpha}_j)$, with $\deg(g) \neq k - d - 1$. Therefore,

$$< G_d > \subseteq GRS_{n-d,k-d}(\tilde{\alpha}', \tilde{\beta}')$$

where $\tilde{\alpha}' = (\tilde{\alpha}_{d+1}, \cdots, \tilde{\alpha}_n)$ and $\tilde{\beta}' = \beta_{d+i} \prod_{j=1}^{d} (\alpha_{d+i} - \alpha_j)$, $i = 1, \cdots, n - d$. Hence, we are in the same case as above and it is possible to retrieve $\tilde{\alpha}'$ and $\tilde{\beta}'$.

Wieschebrink used square codes and shortened codes to attack the Berger-Loidreau variant. His method was extended by A. Couvreur et al. to attack the variants of the McEliece cryptosystem proposed by Baldi et al. as we will see in the next subsection.

Both Wieschebrink and Sidelnikov-Shestakov attacks use the vulnerability that the public key is permutation equivalent to a GRS code. In the next chapter, we present a new cryptosystem where the matrix P will not be a permutation anymore. It will be more dense matrix in order to protect the encoder of a GRS code against structural attacks.

3.3.3 Distinguisher-Based Attack on the BBCRS variant

Baldi, Bianchi, Chiaraluce, Rosenthal and Schipani in [6] also used GRS codes in a variant, which we will designate by the BBCRS cryptosystem, replacing the permutation matrix used in the original proposal by a denser transformation matrix. This proposal was also attacked, this time by A. Couvreur, A., V. Otmani, J. P. Tillich and V. Gauthier-Umana in [19].

We start by describing the BBCRS scheme. The main element that differentiates this proposal from the original cryptosystem is the replacement of the permutation matrix P with an inverse of a dense transformation matrix Q. The construction of this matrix is crucial for the protection of the attack of Sidelnikov and Shestakov. Given G, a generator matrix of a $GRS_{n,k}$ code, S a $k \times k$ non-singular matrix, and Q an $n \times n$ non-singular transformation matrix, the public key is $G' = S^{-1}GQ^{-1}$. The matrix Q is constructed as $Q = R + T$ where R is a dense $n \times n$ matrix with rank z, T is a sparse $n \times n$ matrix of average row weight m (below, we explain what this means) and the elements of these matrices belong $\mathbb{F}_q$. The matrix R is obtained as follows:

$$R = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_w \end{bmatrix}^T \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_w \end{bmatrix} \tag{3.4}$$

where A_i and B_i, $i = 1, \cdots, w$, are $z \times n$ matrices, with $z \leq n$. In their proposal they consider the following two particular cases:

- $A_1 = A$, $A_2 = \mathbf{0}$, where $\mathbf{0}$ represent the all-zero $z \times n$ matrix

$$R = \begin{bmatrix} A \\ \mathbf{0} \end{bmatrix}^T \cdot \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}.$$

- $B_1 = B$ and $B_2 = \mathbf{1} + B$, where $\mathbf{1}$ represent the all-one $z \times n$ matrix

$$R = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}^T \cdot \begin{bmatrix} B \\ \mathbf{1} + B \end{bmatrix}.$$

The matrices R and T are built in such way that $Q = R + T$ is a $n \times n$ invertible matrix.

Relatively to the error correction capability, there is a change from the previous proposal, since they define $t^* = \left\lfloor \frac{n-k}{2m} \right\rfloor$ and impose specific conditions on the vector $\mathbf{e}$, namely, $A\mathbf{e}^T = 0$. Since the public key is $G' = S^{-1}GQ^{-1}$, to encrypt a message $\mathbf{u} \in \mathbb{F}_q^k$, we selected a random error vector $\mathbf{e} \in \mathbb{F}_q^n$, with $wt(\mathbf{e}) \leq t^*$ and compute $y = \mathbf{u}G' + \mathbf{e}$. To decrypt the message we compute $yQ = \mathbf{u}S^{-1}G + \mathbf{e}Q$ where $\mathbf{e}Q = \mathbf{e}R + \mathbf{e}T$.

In the first case, with $A_1 = A$, $A_2 = \mathbf{0}$, we have $\mathbf{e}R = 0$. Since $wt(\mathbf{e}T) \leq t^*$, we can apply a decoding algorithm and obtain $\mathbf{u}S^{-1}$, and then multiplying by S we get $\mathbf{u}$.

In the second case, we need to have $wt(\mathbf{e}Q) \leq t^*$ to do the same process. We have $\mathbf{e}Q = \mathbf{e}A_2(\mathbf{1} + B) + \mathbf{e}T$.

In this proposal they consider that there are advantages if the matrix T has columns with weight 1 and R has rank 1, because when these values increase, the key size increases, as well as the complexity of the decoding algorithm. The case with $m = 1$ and $z = 1$ is the case which gives the smallest key sizes. However the fact that there are columns with weight one would became a weakness that was used by Tillich in [15] to attack this cryptosystem. Once again it is possible to distinguish a GRS code from a random code. In this case, that is achieved by computing the square code and that can be used to unravel the algebraic structure of the public code. In [19], A. Couvreur, A., V. Otmani, J. P. Tillich and V. Gauthier-Umana were able to attack the BBCRS variant even when $1 < m < 2$. For this reason, other proposal emerged later in which they avoid columns with weight one (see [6]).

Next, we describe the attack in [19] that used the square code of different shortenings of the public code. The idea is to compute the dimension of the square code of shortened versions of the dual of the public code and, using this,to try to reduce this new problem to the original one when T is a permutation matrix. Before describing the attack, we summarize the $BBCRS$ scheme for $1 < m < 2$:

Secret key:

- G a $k \times n$ generator matrix of a GRS code;
- Q an $n \times n$ invertible matrix, $Q = R + T$, where:
- R an $n \times n$ rank-z matrix, such that Q is invertible, it's means that $R = a^T b$, for $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, with a_i and b_i being $z \times 1$ full rank matrices, $\forall i \in \{1, \ldots n\}$;
- T an $n \times n$ non-singular sparse matrix, with row weight $1 < m < 2$ (for example if $m = 1.4$, means that 40% of the rows have weight equal to 2 and the other 60% have weight equal to 1);
- S a $k \times k$ invertible matrix.

The authors considered $z = 1$, since a larger z would imply a larger public key (see [15] for details)

Public key: $G' = S^{-1}GQ^{-1}$.

Encryption: Let $\mathbf{u} \in \mathbb{F}_q^k$ a message, $\mathbf{e} \in \mathbb{F}_q^n$ a random error vector with $wt(\mathbf{e}) \leq t^*$. Compute $y = \mathbf{u}G' + \mathbf{e}$.

Decryption:

1. Guess the value of $\mathbf{e}R$, trying q elements since $\mathbf{e}R = \mathbf{e}a^T b$ and $\mathbf{e}a^T \in \mathbb{F}_q$;
2. Compute $yQ - \mathbf{e}R = \mathbf{u}S^{-1}G + \mathbf{e}Q - \mathbf{e}R = \mathbf{u}S^{-1}G + \mathbf{e}T$;
3. Since $wt(\mathbf{e}T) \leq t^*$, using a decoding algorithm of the GRS code, we obtain $\mathbf{u}S^{-1}$;
4. Multiplying by S we obtain $\mathbf{u}$.

Below we will give an idea of the distinguisher attack as described in [19]. But first, we need to describe two ways of constructing new codes from a given code. The first is just a generalization of the d-shortened code defined in the previous section.

Definition 3.2 (Shortened code)**.** *Let $\mathcal{C}$ an $[n,k]$ linear code and $\mathcal{I}$ a subset of $\{1,\ldots,n\}$. The **shortened code**, $S_{\mathcal{I}}(\mathcal{C})$, is defined as*

$$S_{\mathcal{I}}(\mathcal{C}) = \{(u_i)_{i\notin\mathcal{I}} \mid \textit{exists } \boldsymbol{u} = (u_i)_i \in \mathcal{C} \textit{ such that } u_j = 0 \textit{ when } j \in \mathcal{I}\}.$$

Clearly, a d-shortened code is a particular case of a shortened code, obtained when $\mathcal{I} = \{1,\ldots,d\}$.

Definition 3.3 (Punctured code)**.** *Let $\mathcal{C}$ an $[n,k]$ linear code and $\mathcal{I}$ a subset of $\{1,\ldots,n\}$. The **punctured code**, $P_{\mathcal{I}}(\mathcal{C})$, is defined as*

$$P_{\mathcal{I}}(\mathcal{C}) = \{(u_i)_{i\notin\mathcal{I}} \mid \boldsymbol{u} = (u_i)_i \in \mathcal{C}\}.$$

Consider the following notation: $\mathcal{C}$ is the code generated by G, $\mathcal{C}'$ the code generated by G', $\mathcal{I}_1$ the set of positions which correspond to rows of T of Hamming weight 1 and $\mathcal{I}_2$ the set of positions which correspond to rows of T of Hamming weight 2. The goal of the attack is to recover the matrices T and R. To this end, the parity-check matrix, that is the generator matrix of the dual of the public code, is considered. The attack follows four steps:

1. Detecting columns of H' that belong to $\mathcal{I}_2$ and, therefore, those that belong to $\mathcal{I}_1$;

2. Transforming columns of $\mathcal{I}_2$ into degree 1 columns by linear combinations with columns of $\mathcal{I}_1$;

3. Then, the public code has been transformed into another code $\mathcal{C}'_*$ such that exists a secret GRS code $\mathcal{C}_*$ and a matrix $\Pi R'$ where Π is a permutation matrix and R' is rank-1 matrix such that:

$$\mathcal{C}'_* = \mathcal{C}_*(\Pi + R');$$

4. Finally, applying the attack developed in [15], it is possible to recover Π and R'.

The first step of this attack uses the fact that the dimension of the square code of structured codes like the GRS codes is linear in k (namely $2k-1$, when $2k-1 \leq n$), while the dimension of the square code of random codes is quadratic in k (see [19, Section 1]).

Firstly, one tries to find a set $\mathcal{I}$ such that $(S_{\mathcal{I}}(\mathcal{H}'))^2$ has low dimension in the following sense:

$$\dim\left(S_{\mathcal{I}}(\mathcal{H}')\right)^2 < \min\left\{n- \mid \mathcal{I} \mid, \binom{n-k- \mid \mathcal{I}+1}{2}\right\}$$

then, using Propositions 2.21 and 2.28, to distinguish the public code from a random code (since the public code, in the positions not in $\mathcal{I}$, behaves like a structured GRS code).

The second part of the first step consists in puncturing $S_{\mathcal{I}}(\mathcal{C}'^{\perp})$ in a position $i \notin \mathcal{I}$ such that

$$\dim\left(S_{\mathcal{I}}(\mathcal{C}'^{\perp})\right)^2 = \dim\left(P_i(S_{\mathcal{I}}(\mathcal{C}'^{\perp}))\right)^2 + 1.$$

It turns out that, when the above happens, i belongs to $\mathcal{I}_2$, but when the above equality is not valid then

$$\dim\left(S_{\mathcal{I}}(\mathcal{C}'^{\perp})\right)^2 = \dim\left(P_i(S_{\mathcal{I}}(\mathcal{C}'^{\perp}))\right)^2,$$

and we i can be an element of $\mathcal{I}_1$ or of $\mathcal{I}_2$. So, in this case we cannot decide if column i of H' corresponds to a row of T with weight 1 or weight 2. This phenomenon depends on the choice of $\mathcal{I}$, however by choosing several random subsets $\mathcal{I}$ one is able to decide if i corresponds to rows of weight 1 or of weight 2.

From this attack we can infer that the existence of lines with weight one in the construction of the matrix T is a vulnerability to this attack. This is a very important fact as it follows that to protect any variant of McEliece with GRS codes we need to use weight of at least two [32]. We will use this two-weight masking in our proposal.

3.4 ISD attacks

In this section we deal with the other type of attack to the McEliece cryptosystem, namely, general attacks. As opposed to the structural attack that try to recover the secret key, these attacks aim to decipher the ciphertext in order to obtain the plaintext directly from the public key. To this purpose, the cryptanalyst is faced with the problem of decoding a linear code with an unknown structure, *i.e.*, a random linear code, which is known to be an NP-hard problem [9]. These algorithms are usually called *decoding attacks.* Decoding attacks require exponential time, and therefore still represent only a non-critical threat to McEliece, in the sense that it is enough to enlarge the parameter size in order to make them infeasible. As a consequence, decoding attacks are often used as a tool to determine the minimum parameter size required to achieve the desired security level (e.g. 2^{128} or 2^{256}). The most renowned and highly regarded attack is the technique known as *Information-Set Decoding (ISD)* and all the best decoding attacks are derived from it.

Eugene Prange [52] was the pioneer in the early 1960 in this area and after him a significant research effort has been put into finding more efficient methods in order to solve the random linear code decoding problem. More concretely, Lee and Brickell [36], Leon [37], and Stern [57], proposed decoding attacks through a family of ISD algorithms tailor-made for binary codes. Algorithms for codes over arbitrary finite fields were proposed by Christiane Peters in [50]. In this article a generalization of two information-set decoding algorithms were presented: the Lee-Brickell's and the Stern's algorithms. Moreover, for these attacks, an analysis of the computational cost, *i.e.*, concrete formulas to estimate the *work factor.*

Next, we succinctly explain the idea of the ISD attack. Basically, the Information-Set Decoding tries to solve the following NP-hard problem:

Given a random looking generator matrix $G \in \mathbb{F}^{k\times n}$ of a linear code $\mathcal{C}$ and a vector $\mathbf{y} = \mathbf{u}G + \mathbf{e}$, recover $\mathbf{u}$.

The first step of any ISD algorithm is to find a so-called information-set. Before we introduce its formal definition we consider the following notation:

Notation 3.4. *For a non-empty set $I \subseteq \{1, \cdots, n\}$ we let $\pi_I : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^{|I|}$ be the projection on the coordinates indexed by I.*

Definition 3.5. *Let $\mathcal{C}$ be an $[n, k]$ code over $\mathbb{F}_q$ and $G \in \mathbb{F}_q^{k\times n}$ an arbitrary generator matrix for $\mathcal{C}$. Consider a non-empty set $I \subseteq \{1, \cdots, n\}$ and $\pi_I(G)$ the submatrix of G formed by the columns indexed by I. If $|I| = k$ and $\pi_I(G)$ has rank k, the set I is called an* ***information-set*** *and the I-indexed entries of a codeword $\mathbf{v}$ are* ***information symbols****.*

Lemma 3.6. *Let $\mathcal{C}$ be an $[n,k]$ code over $\mathbb{F}_q$ and let $G \in \mathbb{F}_q^{k\times n}$ be a generator matrix of $\mathcal{C}$. Let $I \subseteq \{1,\cdots,n\}$ be an information-set for $\mathcal{C}$. Then $\pi_I(G) \in \mathbb{F}_q^{k\times k}$ is an invertible matrix and, for all $\boldsymbol{v} \in \mathcal{C}$ we have*

$$\boldsymbol{v} = \pi_I(\boldsymbol{v}) \cdot \pi_I(G)^{-1} \cdot G,$$

that is, G and $\pi_I(G)^{-1}G$ generate the same code, and for any codeword $\boldsymbol{u}\pi_I(G)^{-1}G$, the I-indexed entries will carry the information symbols.

Proof. By the definition given above of information-set, the matrix $\pi_I(G)$ has size $k \times k$ and rank k, therefore invertible. If $\mathbf{v} \in \mathcal{C}$, then $\mathbf{v} = \mathbf{u}G$ for $\mathbf{u} \in \mathbb{F}_q^k$. We trivially have

$$\mathbf{v} = \mathbf{u} \cdot \pi_I(G) \cdot \pi_I(G)^{-1} \cdot G \tag{3.5}$$

Applying π_I to both sides of (3.5) we obtain

$$\pi_I(\mathbf{v}) = \mathbf{u} \cdot \pi_I(G) \cdot \pi_I(G)^{-1} \cdot \pi(G) = \mathbf{u} \cdot \pi_I(G).$$

Therefore,

$$\mathbf{u} = \pi_I(\mathbf{v}) \cdot \pi_I(G)^{-1},$$

and consequently

$$\mathbf{v} = \pi_I(\mathbf{v}) \cdot \pi_I(G)^{-1} \cdot G.$$

□

Proposition 3.7. *Let $\mathcal{C}$ be an $[n,k,d]$ code and $G \in \mathbb{F}_q^{k\times n}$ be a generator matrix of $\mathcal{C}$. Let $\boldsymbol{v} \in \mathcal{C}$, t the error correction capability and $\boldsymbol{y} = \boldsymbol{v} + \boldsymbol{e}$ the message received, where $\boldsymbol{e} \in \mathbb{F}_q^n$ is the error vector. There exits a information-set $I \subseteq \{1,\cdots,n\}$ of $\mathcal{C}$ for which*

$$\boldsymbol{v} = \pi_I(\boldsymbol{y}) \cdot \pi_I(G)^{-1} \cdot G.$$

In particular, there exists an information-set $I \subseteq \{1,\cdots,n\}$ of $\mathcal{C}$ with

$$d(\boldsymbol{y}, \pi_I(\boldsymbol{y}) \cdot \pi_I(G)^{-1} \cdot G) \leq \left\lfloor \frac{d-1}{2} \right\rfloor.$$

Proof. Let $T := \{1,\cdots,n\} \setminus \sigma(\mathbf{e})$, where $\sigma(\mathbf{e})$ is the set of the nonzero positions of $\mathbf{e}$. We want to prove that $\pi_T(G)$ has rank k. Suppose, by contradiction, that $\pi_T(G)$ has rank less than or equal to $k-1$. Then, there exists $\mathbf{v} \in \mathcal{C}$ such that $\mathbf{v} \neq 0$ and $\sigma(\mathbf{v}) \cap T = \emptyset$. Therefore, $1 \leq wt(\mathbf{v}) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$, contradicting the fact that $\mathcal{C}$ has minimum distance d.

Since $\pi_T(G)$ has rank k, there exists an information-set I for $\mathcal{C}$ with $I \subseteq T$. In particular, $I \cap \sigma(\mathbf{e}) = \emptyset$ and so $\pi_I(\mathbf{y}) = \pi_I(\mathbf{v})$. Therefore,

$$\pi_I(\mathbf{y}) \cdot \pi_I(G)^{-1} \cdot G = \pi_I(\mathbf{v}) \cdot \pi_I(G)^{-1} \cdot G = \mathbf{v},$$

where the latter identity follows from (3.6). □

We can describe a decoding algorithm for a linear code based on information sets in the following way:
Algorithm of Information-Set decoding
Inputs:

1. The generator matrix $G \in \mathbb{F}_q^{k\times n}$ of an $[n,k,d]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$;
2. The collection $\mathcal{I}(\mathcal{C})$ of all information-sets of $\mathcal{C}$;
3. The received vector $\mathbf{y} \in \mathbb{F}_q^n$.

Procedure:

1. For all $I \in \mathcal{I}(\mathcal{C})$ compute $\mathbf{v}_I := \pi_I(\mathbf{y}) \cdot \pi_I(G)^{-1} \cdot G \in \mathcal{C}$;
2. If $d(\mathbf{y}, \mathbf{v}_I) \leq \lfloor \frac{d-1}{2} \rfloor$, then return $\mathbf{v}_I$ and terminate the algorithm.

Christiane Peters presented in [50] a generalization of the Stern algorithm for codes over arbitrary fields. In Stern's algorithm there are two parameters: p and ℓ. The algorithm uses the idea of allowing a fixed number of errors in the information-set and search for the error vector, restricting the possible candidates to vectors having ℓ zeros outside the I-indexed columns. Stern's algorithm divides the information-set I into two equal-size subsets X and Y, and looks for words having exactly weight p among the columns indexed by X, words with exactly weight p among the columns indexed by Y, and words with exactly weight 0 on a fixed uniform random set of l positions outside the I-indexed columns. Hence, in the Stern's algorithm, the aim is to find error vectors with weights p, p, 0 when restricted to certain index sets X, Y, Z, respectively. Here X and Y are sets of size $\frac{k}{2}$, and Z has size ℓ. The parameters p and ℓ need to be chosen as to optimize the average run time. For more details of the algorithm and work factor's estimates, see [57].

The *work-factor* is the amount of effort or work required to break a cryptosystem, *i.e.*, the number of bits operations that an adversary needs in order to break it. In modern encryption algorithms the threshold of 2^{128} or 2^{256} bit operations is considered secure. In general, we can say that

$$WF = mS \times \frac{1}{P_r}$$

where S is the cost of one iteration, P_r is the probability of success of each iteration (where we assume this probability does not depend on the iteration) and m is the number of bits needed to describe an element of $\mathbb{F}_q$.

In the case of the Stern's algorithm for codes over $\mathbb{F}_q$, with $q = 2^m$, r the number of errors and the chosen parameters p and ℓ, Peters presented in [50] the following work factor estimate:

$$WF_{p,\ell} = mS_{p,\ell} \frac{\binom{n}{r}}{\binom{k/2}{p}^2 \binom{n-k-\ell}{r-2p}}, \tag{3.6}$$

where

$$S_{p,\ell} = (n-k)^2(n+k) + \left[k/2 - p + 1 + 2\binom{k/2}{p}(q-1)^p\right]\ell + \\ + \frac{2pq(r-2p+1)(2q-3)(q-1)^{2p-2}}{q^\ell}\binom{k/2}{p}^2.$$

In the next chapter we will use this work factor estimate formula in order to obtain parameters sufficiently large for the security of 2^{128}, 2^{256} and 2^{512} bit operations.

Chapter 4

McEliece Cryptosystems with Convolutional Encoders

In this chapter we present a variant of de McEliece cryptosystem that instead of block codes uses convolutional encoders. The idea of this new variant is different from all previous constructions, even the ones that used convolutional codes, since it is based on building a polynomial encoder as the product of a block code encoder and two invertible Laurent polynomial matrices. Hence, the decoding is performed by inverting the polynomial matrices and decode the corrupted data using the block code. This new variant aimed at opening the possibility of trying to use again different classes of codes such as GRS in alternative to the Goppa codes. In this proposed scheme the message is not a block vector but a stream of vectors sent in a sequential fashion. Again the security relies in the difficulty of decoding a general convolutional code (specially hard when the degree of the code is large).
Our construction uses large parts of randomly generated matrices in order to mask the secret key and the truncated sliding matrices of the encoders are not full row rank (i.e., they are not generator matrices of a block code).

In the first section we introduce some definitions and results. In the second we describe the cryptosystem and finally in the third section we analyse the security with respect to decoding attacks and structural attacks.

4.1 Definitions

For the construction of the new proposal, the general ingredients are the following. Let $G \in \mathbb{F}_q^{k \times n}$ be an encoder of an $[n, k, 2t+1]$ block code that admits an efficient decoding algorithm and can correct up to t errors,

$$S(D) = \sum_{i=\mu}^{\nu} S_i D^i \in \mathbb{F}_q^{k\times k}[D] \tag{4.1}$$

where ν, μ are integers such that $\nu \geq \mu$, $S_\mu \in \mathbb{F}_q^{k\times k}$ is an invertible constant matrix and

$$T(D^{-1}, D) = \sum_{j=-\mu}^{\mu} T_j D^j \in \mathbb{F}_q^{n\times n}(D^{-1}, D) \tag{4.2}$$

is an invertible (in $\mathbb{F}_q((D))$) Laurent polynomial matrix such that

1. each row of T_i has at most ρ nonzero elements, for $i = -\mu, \dots, 0, \dots, \mu$;
2. $|T(D^{-1}, D)| \in \mathbb{F}_q \setminus \{0\}$.

We propose to construct a convolutional encoder $G'(D)$ as:

$$G'(D) = S(D)\ G\ T^{-1}(D^{-1}, D). \tag{4.3}$$

Note that the encoder $G'(D)$ is a polynomial, and therefore can be used as a blueprint for building a physical encoder for the code, i.e., a device which can be used to transform k parallel streams of information symbols into n parallel streams of coded symbols.

Recall that the Hamming weight of a vector $\mathbf{v} \in \mathbb{F}_q^n$, $\mathrm{wt}(\mathbf{v})$, is the number of the nonzero components of $\mathbf{v}$. This definition can be extended to polynomial vectors $\mathbf{v}(D^{-1}, D) = \sum_{i\in\mathbb{Z}} \mathbf{v}_i D^i$ in a natural way as $\mathrm{wt}(\mathbf{v}(D^{-1}, D)) = \sum_{i\in\mathbb{Z}} \mathrm{wt}(\mathbf{v}_i)$.

For the sake of simplicity we shall consider information vectors that start at time instant zero and have finite support, i.e., $\mathbf{u}(D) \in \mathbb{F}_q^k[D]$. We will also consider the error vectors $\mathbf{e}(D) \in \mathbb{F}_q^n[D]$ to be polynomials.

Lemma 4.1. *Let $T(D^{-1}, D)$ be a Laurent polynomial matrix as described in (4.2) and let $\boldsymbol{e}(D) = \sum_{i\geq 0} \boldsymbol{e}_i D^i \in \mathbb{F}_q^n[D]$ be a random error vector satisfying*

$$\mathrm{wt}((\boldsymbol{e}_i, \boldsymbol{e}_{i+1}, \dots, \boldsymbol{e}_{i+2\mu})) \leq \frac{t}{\rho} \tag{4.4}$$

for all $i \geq 0$. Then all the coefficients of $\boldsymbol{e}(D)T(D^{-1}, D)$ have weight less than or equal to t.

Proof. It readily follows from the condition 1. above, that if each row of T_j has at most ρ nonzero elements, then $\mathrm{wt}(\mathbf{e}_i T_j) \leq \rho \cdot \mathrm{wt}(\mathbf{e}_i)$ for all $i \geq 0$ and $-\mu \leq j \leq \mu$. Take $\mathbf{e}_i = 0$ for $i < 0$, then the ℓ-th coefficient of $\mathbf{e}(D)T(D^{-1}, D)$ is given by $\displaystyle\sum_{j=-\mu}^{\mu} \mathbf{e}_{\ell-j}T_j$, for $\ell \geq -\mu$, and the result follows. □

Condition (4.4) describes the maximum number of errors allowed within a time interval and is similar to the sliding window condition introduced in [4] to describe the possible error patterns that can occur in a given channel.

Theorem 4.2. *Let $G'(D)$ be the encoder as described in (4.3), t the correcting error capability of G, $\mathbf{u}(D)$ the information sequence and $\boldsymbol{e}(D)$ an error vector satisfying (4.4). Then, the received data*

$$\boldsymbol{y}(D) = \mathbf{u}(D)G'(D) + \boldsymbol{e}(D) \in \mathbb{F}_q^n[D]$$

can be successfully decoded.

Proof. Multiplying $\mathbf{y}(D)$ by $T(D^{-1}, D)$ from the right yields the polynomial equation

$$\mathbf{y}(D)T(D^{-1}, D) = \mathbf{u}(D)S(D)G + \mathbf{e}(D)T(D^{-1}, D).$$

Therefore, if $\mathbf{u}(D) = \mathbf{u}_0 + \mathbf{u}_1 D + \cdots + \mathbf{u}_s D^s$ and $\mathbf{e}(D) = \mathbf{e}_0 + \mathbf{e}_1 D + \cdots + \mathbf{e}_{s+\mu+\nu} D^{s+\mu+\nu}$, we can write

$$\mathbf{u}(D)S(D) = \sum_{i=\mu}^{\nu+s} \widehat{\mathbf{u}}_i D^i \tag{4.5}$$

and

$$\mathbf{e}(D)T(D^{-1}, D) = \sum_{i=-\mu}^{s+2\mu+\nu} \widehat{\mathbf{e}}_i D^i,$$

for some coefficients $\widehat{\mathbf{u}}_i$ and $\widehat{\mathbf{e}}_i \in \mathbb{F}_q^n$. So, each coefficient of $\mathbf{y}(D)T(D^{-1}, D)$ is of the form $\widehat{\mathbf{u}}_i G + \widehat{\mathbf{e}}_i$. By Lemma 4.1 it follows that $\mathrm{wt}(\widehat{\mathbf{e}}_i) \leq t$, for $-\mu \leq i \leq s + 2\mu + \nu$ and, therefore, each $\widehat{\mathbf{u}}_i$ can be recovered. Furthermore, from (4.5) we have that

$$\begin{aligned}
&\left[\begin{array}{cccc} \widehat{\mathbf{u}}_{\mu} & \widehat{\mathbf{u}}_{\mu+1} & \cdots & \widehat{\mathbf{u}}_{\nu+s} \end{array}\right] = \\
&= \left[\begin{array}{cccc} \mathbf{u_0} & \mathbf{u_1} & \cdots & \mathbf{u_s} \end{array}\right]
\left[\begin{array}{cccccccccc}
S_{\mu} & S_{\mu+1} & \cdots & S_{\nu} & & & & & & \\
 & S_{\mu} & S_{\mu+1} & \cdots & S_{\nu} & & & & & \\
 & & \ddots & \ddots & \ddots & \ddots & & & & \\
 & & & S_{\mu} & \cdots & \cdots & S_{\nu} & & & \\
 & & & & \ddots & & \vdots & \ddots & & \\
 & & & & & \ddots & \vdots & & \ddots & \\
 & & & & & & S_{\mu} & S_{\mu+1} & \cdots & S_{\nu}
\end{array}\right],
\end{aligned}$$

so, one can recover each $\mathbf{u}_i$ sequentially as

$$\begin{aligned}
\mathbf{u}_0 &= \widehat{\mathbf{u}}_{\mu} S_{\mu}^{-1}, \\
\mathbf{u}_1 &= \left(\widehat{\mathbf{u}}_{\mu+1} - \mathbf{u}_0 S_{\mu+1}\right) S_{\mu}^{-1}, \\
\mathbf{u}_2 &= \left(\widehat{\mathbf{u}}_{\mu+2} - \mathbf{u}_0 S_{\mu+2} - \mathbf{u}_1 S_{\mu+1}\right) S_{\mu}^{-1}, \qquad (4.6) \\
&\vdots \\
\mathbf{u}_i &= \left(\widehat{\mathbf{u}}_{\mu+i} - \sum_{j=\max\{0, i-(\nu-\mu+1)\}}^{i-1} \mathbf{u}_j S_{\mu+i-j}\right) S_{\mu}^{-1}.
\end{aligned}$$

□

Notice that if $s \leq \nu - \mu$ we do not need to use all the S_j to retrieve $\mathbf{u}(D)$. If we let

$$\sum_{i \geq -\mu} \widehat{\mathbf{y}}_i D^i = \mathbf{y}(D) T(D^{-1}, D),$$

then, at each time instant i, with $i \geq -\mu$, we compute each

$$\widehat{\mathbf{y}}_i = \sum_{j=-\mu}^{\min\{i,\mu\}} \mathbf{y}_{i-j} T_j$$

by performing at most $2\mu + 1$ multiplications of vectors of size n by a matrix of order n, then we decode $\widehat{\mathbf{y}}_i$ using the decoding algorithm corresponding to G to obtain $\widehat{\mathbf{u}}_i$, e.g., using Berlekamp-Massey algorithm requires $\mathcal{O}(n^2)$ field operations. Finally we retrieve $\mathbf{u}_j$ by performing at most $\nu - \mu + 1$ multiplications of vectors of size k by a matrix of order k and at most $\nu - \mu$ sums of vectors of order k. Recall that the complexity of each vector-matrix product is $\mathcal{O}(n^2)$.

Remark 4.3. *Let see in detail what happens when $\mu = 1$, $\nu = 2$ and $s = 2$. We have:*

$$\begin{aligned}
\mathbf{u}(D) &= \mathbf{u}_0 + \mathbf{u}_1 D + \mathbf{u}_2 D^2 \\
S(D) &= S_1 D + S_2 D^2 \\
T(D^{-1}, D) &= T_{-1} D^{-1} + T_0 + T_1 D \\
P(D^{-1}, D) &= T^{-1}(D^{-1}, D).
\end{aligned}$$

Since $P(D^{-1}, D) = T^{-1}(D^{-1}, D)$:

$$\begin{aligned}
I &= P(D^{-1}, D)T^{-1}(D^{-1}, D) \\
I &= (P_{-1}D^{-1} + P_0 + P_1 D)(T_{-1}D^{-1} + T_0 + T_1 D) \\
I &= P_{-1}T_{-1}D^{-2} + (P{-}1T_0 + P_0 T_{-1})D^{-1} + P_{-1}T_1 + P_0 T_0 + P_1 T_{-1} + \\
&+ (P_0 T_1 + P_1 T_0)D + P_1 T_1 D^2.
\end{aligned}$$

So, $P_{-1}T_{-1} = 0$, $P_{-1}T_0 + P_0 T_{-1} = 0$, $P_0 T_1 + P_1 T_0 = 0$, $P_1 T_1 = 0$ and $P_{-1}T_1 + P_0 T_0 + P_1 T_{-1} = I$.
In this case, the convolutional encoder $G'(D)$ is given by

$$\begin{aligned}
&G'(D) = (S_1 D + S_2 D^2)G(P_{-1}D^{-1} + P_0 + P_1 D) \\
&\qquad = S_1 G P_{-1} + (S_1 G P_0 + S_2 G P_{-1})D + (S_1 G P_1 + S_2 G P_0)D^2 + S_2 G P_1 D^3.
\end{aligned}$$

The encryption of the message $\mathbf{u}(D)$ is given by:

$$\begin{aligned}
&\boldsymbol{y}(D) = \mathbf{u}(D)G'(D) + \boldsymbol{e}(D) \\
&\qquad = [\mathbf{u}_0 S_1 G P_{-1} + \boldsymbol{e}_0] + [\mathbf{u}_0 S_1 G P_0 + \mathbf{u}_0 S_2 G P_{-1} + \mathbf{u}_1 S_1 G P_{-1} + \boldsymbol{e}_1]D + \\
&\quad +[\mathbf{u}_0 S_1 G P_1 + \mathbf{u}_0 S_2 G P_0 + \mathbf{u}_1 S_1 G P_0 + \mathbf{u}_1 S_2 G P_{-1} + \mathbf{u}_2 S_1 G P_{-1} + \boldsymbol{e}_2]D^2 + \\
&\quad +[\mathbf{u}_0 S_2 G P_1 + \mathbf{u}_1 S_1 G P_1 + \mathbf{u}_1 S_2 G P_0 + \mathbf{u}_2 S_1 G P_0 + \mathbf{u}_2 S_2 G P_{-1} + \boldsymbol{e}_3]D^3 + \\
&\quad +[\mathbf{u}_1 S_2 G P_1 + \mathbf{u}_2 S_1 G P_1 + \mathbf{u}_2 S_2 G P_0 + \boldsymbol{e}_4]D^4 + [\mathbf{u}_2 S_2 G P_1 + \boldsymbol{e}_5]D^5
\end{aligned}$$

Let's look at the decoding process next.

Multiplying $\boldsymbol{y}(D)$ by $T(D^{-1}, D)$ we obtain:

$$
\begin{aligned}
&\boldsymbol{y}(D)T(D^{-1}, D) = \\
&\qquad (\boldsymbol{e}_0 T_{-1})D^{-1} + [\mathbf{u}_0 S_1 G(P_{-1}T_0 + P_0 T_{-1}) + \boldsymbol{e}_0 T_0 + \boldsymbol{e}_1 T_{-1}] + \\
&\qquad + [\mathbf{u}_0 S_1 G(P_{-1}T_1 + P_0 T_0 + P_1 T_{-1}) + \mathbf{u}_0 S_2 G(P_{-1}T_0 + P_0 T_{-1}) + \\
&\qquad + \mathbf{u}_1 S_1 G(P_{-1}T_0 + P_0 T_{-1}) + \boldsymbol{e}_0 T_1 + \boldsymbol{e}_1 T_0 + \boldsymbol{e}_2 T_{-1}]\, D + \\
&\qquad + [\mathbf{u}_0 S_1 G(P_0 T_1 + P_1 T_0) + \mathbf{u}_0 S_2 G(P_{-1}T_1 + P_0 T_0 + P_1 T_{-1}) + \\
&\qquad + \mathbf{u}_1 S_1 G(P_{-1}T_1 + P_0 T_0 + P_1 T_{-1}) + \mathbf{u}_1 S_2 G(P_{-1}T_0 + P_0 T_{-1}) + \\
&\qquad + \mathbf{u}_2 S_1 G(P_{-1}T_0 + P_0 T_{-1}) + \boldsymbol{e}_1 T_1 + \boldsymbol{e}_2 T_0 + \boldsymbol{e}_3 T_{-1}]\, D^2 + \\
&\qquad + [\mathbf{u}_0 S_2 G(P_0 T_1 + P_1 T_0) + +\mathbf{u}_1 S_1 G(P_0 T_1 + P_1 T_0) \\
&\qquad + \mathbf{u}_1 S_2 G(P_{-1}T_1 + P_0 T_0 + P_1 T_{-1}) + u_2 S_1 G(P_{-1}T_1 + P_0 T_0 + P_1 T_{-1}) + \\
&\qquad + \mathbf{u}_2 S_2 G(P_{-1}T_0 + P_0 T_{-1}) + \boldsymbol{e}_2 T_1 + \boldsymbol{e}_3 T_0 + \boldsymbol{e}_4 T_{-1}]\, D^3 + \\
&\qquad + [\mathbf{u}_1 S_2 G(P_0 T_1 + P_1 T_0) + \mathbf{u}_2 S_1 G(P_0 T_1 + P_1 T_0) + \\
&\qquad + \mathbf{u}_2 S_2 G(P_{-1}T_1 + P_0 T_0 + P_1 T_{-1}) + \boldsymbol{e}_3 T_1 + \boldsymbol{e}_4 T_0 + \boldsymbol{e}_5 T_{-1}]\, D^4 + \\
&\qquad + [\mathbf{u}_2 S_2 G(P_0 T_1 + P_1 T_0) + \boldsymbol{e}_4 T_1 + \boldsymbol{e}_5 T_0]\, D^5 + [\boldsymbol{e}_5 T_1]\, D^6.
\end{aligned}
$$

4.2 Construction

We propose a new scheme of the McEliece PKC where a secret encoder of a block code is masked by polynomial matrices yielding a polynomial encoder of a convolutional code, which constitutes the public key. We shall consider the type of encoders described in Section 2. In this context, the information vector $\mathbf{u}(D)$ represents the message to be interchanged (usually, in the applications, it is a secret key for a symmetric cryptosystem) and thus will be a polynomial with designed fixed degree. The proposed scheme works as follows:
Secret key: $\{S(D), G, T(D^{-1}, D)\}$.
Public key: $\{G'(D) = S(D)GT^{-1}(D^{-1}, D),\ t/\rho,\ \mu\}$.
Encryption: Alice selects an error vector $\mathbf{e}(D)$ satisfying (4.4) and encrypts the message $\mathbf{u}(D) = \mathbf{u}_0 + \mathbf{u}_1 D + \mathbf{u}_2 D^2 + \cdots + \mathbf{u}_s D^s \in \mathbb{F}_q^k[D]$ as

$$\mathbf{y}(D) = \mathbf{u}(D)G'(D) + \mathbf{e}(D). \tag{4.7}$$

Decryption: Bob multiplies (4.7) from the right by the matrix $T(D^{-1}, D)$ to obtain

$$\mathbf{u}(D)S(D)G + \mathbf{e}(D)T(D^{-1}, D), \tag{4.8}$$

he then decodes using G and then recovers the message $\mathbf{u}(D)$ from $\mathbf{u}(D)S(D)$, as explained in the proof of Theorem 4.2. Note that this can be done sequentially since, if we write

$$\mathbf{y}(D)T(D^{-1},D) = \mathbf{u}(D)S(D)G + \mathbf{e}(D)T(D^{-1},D) = \sum_{i=-\mu}^{s+2\mu+\nu} \widehat{\mathbf{y}}_i D^i,$$

then each coefficient $\widehat{\mathbf{y}}_i$, for $-\mu \leq i \leq s+2\mu+\nu$ is of the form

$$\widehat{\mathbf{y}}_i = \sum_{j=-\mu}^{\min\{i,\mu\}} \mathbf{y}_{i-j} T_j,$$

and hence, Bob can compute each $\widehat{\mathbf{y}}_i$ when he has received the necessary $\mathbf{y}_j$'s, that is, the, at most $2\mu+1$, necessary vectors. Now, since we have that

$$\mathbf{u}(D)S(D) = \sum_{i=\mu}^{\nu+s} \widehat{\mathbf{u}}_i D^i \qquad \text{and} \qquad \mathbf{e}(D)T(D^{-1},D) = \sum_{i=-\mu}^{s+2\mu+\nu} \widehat{\mathbf{e}}_i D^i,$$

then, for every $j = \mu, \ldots, \mu+s$, the j-th coefficient of $\mathbf{y}(D)T(D^{-1},D)$ is $\widehat{\mathbf{y}}_j = \widehat{\mathbf{u}}_j G + \widehat{\mathbf{e}}_j$, and hence if $\widehat{\mathbf{e}}_j$ has t or less errors, Bob can recover $\widehat{\mathbf{u}}_j$, from which he recovers the first $s+1$ coefficients of $\mathbf{u}(D)S(D)$, that are the ones he needs. Notice that Bob does not need to calculate $\widehat{\mathbf{y}}_j$, for $-\mu \leq j \leq \mu-1$. Also he only needs the errors from $\widehat{\mathbf{e}}_\mu$ to $\widehat{\mathbf{e}}_{\mu+s}$, and these errors are of the form

$$\widehat{\mathbf{e}}_i = \sum_{j=-\mu}^{\mu} \mathbf{e}_{i-j} T_j.$$

In order to reduce the key size, we select G to be an encoder of a GRS code and construct $T(D^{-1},D)$ in such a way that it can protect the structure of G and at the same time remain secure to ISD attacks. Recall that a GRS generator matrix is given by two vectors $\alpha \in (\mathbb{F}_q \setminus \{0\})^n$ with $\alpha_i \neq \alpha_j$ for every $i \neq j$, and $\mathbf{x} \in (\mathbb{F}_q \setminus \{0\})^n$. The generator matrix is constructed as

$$G = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_1\alpha_1 & x_2\alpha_2 & \cdots & x_n\alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1\alpha_1^{k-1} & x_2\alpha_2^{k-1} & \cdots & x_n\alpha_n^{k-1} \end{pmatrix}. \tag{4.9}$$

We denote these codes by $GRS_{n,k}(\alpha, \mathbf{x})$. GRS are MDS codes, and therefore can correct up to $t = \left\lfloor \frac{n-k}{2} \right\rfloor$ errors. Due to security reasons, we must consider ρ to be at least 2 when using GRS (see [32, 11]). Properties 4.4 present some desirable conditions that $T(D^{-1},D)$ should fulfill in order to construct a secure public key.

Properties 4.4. *Let*

$$T(D^{-1}, D) = \sum_{j=-\mu}^{\mu} T_j D^j \in \mathbb{F}_q^{n\times n}(D^{-1}, D),$$

be a Laurent polynomial matrix. We want $T(D^{-1}, D)$ to satisfy the following properties:

a) $|T(D^{-1}, D)| \in \mathbb{F}_q \setminus \{0\}$, *i.e., invertible (in $\mathbb{F}_q(D^{-1}, D)$);*

b) *Each nonzero row of T_j has two nonzero elements, for $-\mu \leq j \leq \mu$ (i.e., $\rho = 2$);*

c) *The positions of the nonzero columns of T_j form a partition of n;*

d) *If we write*

$$T^{-1}(D^{-1}, D) = P(D^{-1}, D) = \sum_{j=-\mu}^{\mu} P_j D^j \in \mathbb{F}_q^{n\times n}(D^{-1}, D),$$

then the nonzero columns of P_j have at least 2 nonzero entries.

Remark 4.5. *Let $-\mu \leq j \leq \mu$ and d_j be the number of nonzero columns of T_j. Conditions* **a)** *and* **c)** *in Properties 4.4 imply that*

$$\mu d_{-\mu} + \cdots + 2d_{-2} + d_{-1} = d_1 + 2d_2 \cdots + \mu d_\mu. \tag{4.10}$$

Remark 4.6. *Note that $G'_0 = S_\mu G P_{-\mu}$, and therefore if we assume $d_{-\mu} = d_\mu$, we have*

$$\operatorname{rank}(G'_0) \leq \operatorname{rank}(P_{-\mu}) \leq d_\mu.$$

In particular, if $\sigma \leq s-\mu-\nu$ we have that the rank of the so-called truncated sliding matrix *of $G'(D)$,*

$$G'_{truc}(\sigma) = \begin{bmatrix} G'_0 & G'_1 & \cdots & G'_{\mu+\nu} & & \\ & G'_0 & \cdots & G'_{\mu+\nu-1} & \ddots & \\ & & \ddots & \vdots & \ddots & G'_{\mu+\nu} \\ & & & G'_0 & \cdots & G'_{\mu+\nu-1} \\ & & & & \ddots & \vdots \\ & & & & & G'_0 \end{bmatrix} \in \mathbb{F}_q^{(\sigma+1)k\times(\sigma+1)n}$$

is also less than or equal to $\sigma k + d_\mu$, and we have a rank deficiency of at least $k - d_\mu$.

4.2.1 Constructing the matrices $T(D^{-1}, D)$

In this subsection we present a class of matrices $T(D^{-1}, D)$ that satisfies the conditions given above and study some of the properties that will provide the desired security (discussed in Section 4.3). We consider a construction of $T(D^{-1}, D)$ where the nonzero columns of P_i will always have an even number of nonzero elements, and having, in general, many columns with more than two nonzero elements.

We start by constructing a matrix $A = A(D^{-1}, D)$ from which we later construct the matrix $T(D^{-1}, D)$. In this construction we take n, k and all d_j (defined in Remark 4.5) to be even. Define $\widetilde{n} = n/2$, $\widetilde{k} = k/2$ and $\widetilde{d_j} = d_j/2$, for $-\mu \leq j \leq \mu$.

Properties 4.7. *The matrix $A = A(D^{-1}, D) \in \mathbb{F}(D^{-1}, D)^{\widetilde{k}\times\widetilde{n}}$ satisfies the following conditions:*

1. *A is an upper triangular matrix;*
2. *In the i-th entry of the principal diagonal of A, there is an element of the form $\beta_i D^j$, where $\beta_i \in \mathbb{F} \setminus \{0\}$, for $1 \leq i \leq \widetilde{n}$, and $j \in \{-\mu, \ldots, 0, \ldots, \mu\}$ in such a way that there are $\widetilde{d_j}$ entries with power D^j, satisfying*
$$\mu\widetilde{d}_{-\mu} + \cdots + 2\widetilde{d}_{-2} + \widetilde{d}_{-1} = \widetilde{d}_1 + 2\widetilde{d}_2 + \cdots + \mu\widetilde{d}_\mu; \tag{4.11}$$
3. *Each row of A has at most one entry of the form γD^j for each $-\mu \leq j \leq \mu$, with $\gamma \in \mathbb{F} \setminus \{0\}$;*
4. *All nonzero entries of a column of A have the same exponent of D.*

Condition (4.11) implies
$$|\, A \,| = \prod_{i=1}^{\widetilde{n}} \beta_i,$$
i.e, $|A| \in \mathbb{F} \setminus \{0\}$. All the β_i can be randomly chosen, as well as all the $\gamma \in \mathbb{F}$ that appear above the principal diagonal.

Example 4.8. *In $\mathbb{F}_3$ the following matrix satisfies Properties 4.7:*
$$A = \begin{bmatrix} 2D & 2 & D^{-2} & 0 \\ 0 & 1 & 0 & D \\ 0 & 0 & 2D^{-2} & 2D \\ 0 & 0 & 0 & D \end{bmatrix}$$

Before creating our $T(D^{-1}, D)$ we need a technical lemma about the determinant and inverse of a block matrix first obtained by I. Schur [55].

Lemma 4.9. *[14, Formulas (2) and (4)] Let T be a block matrix of the form*

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

where A and D are nonsingular. Then

a) $|T| = |A|\,|D - CA^{-1}B|$.

b) *If T is invertible, the inverse of T is*

$$T^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix}.$$

We are now in position to construct a matrix $T(D^{-1}, D)$ with all the desirable properties.

Lemma 4.10. *Suppose $|\mathbb{F}| > 2$. Let $A = A(D^{-1}, D)$ be a matrix satisfying the conditions of Properties 4.7. Take $\beta \in \mathbb{F} \setminus \{0, 1\}$, and define $B = \beta A$. Consider the matrices Γ and $\Delta(D^{-1}, D)$ where*

1. *$\Gamma \in \mathbb{F}^{k\times k}$ is a permutation matrix;*
2. *$\Delta(D^{-1}, D)$ is the block matrix*

$$\Delta(D^{-1}, D) = \left[\begin{array}{c|c} A & B \\ \hline A & A \end{array}\right] \in \mathbb{F}^{n\times n}(D^{-1}, D).$$

Then $T(D^{-1}, D) = \Gamma\,\Delta(D^{-1}, D)$ satisfies the conditions in Properties 4.4.

Proof. Using Lemma 4.9 **a)**, we have

$$|\Delta(D^{-1}, D)| = |A|\,|A - AA^{-1}B| = \left(\prod_{i=1}^{\tilde{n}} \beta_i\right)^2 (1-\beta)^{\tilde{n}}.$$

Therefore, $|T(D^{-1}, D)| \in \mathbb{F} \setminus \{0\}$.

Conditions **b)** and **c)** of Properties 4.4 hold due to Properties 4.7 and our construction of $\Delta(D^{-1}, D)$. Condition **d)** is obtained using Lemma 4.9 **b)**, since we have

$$\Delta^{-1}(D^{-1}, D) = \begin{bmatrix} \frac{1}{1-\beta}A^{-1} & \frac{\beta}{\beta-1}A^{-1} \\ \frac{1}{\beta-1}A^{-1} & \frac{1}{1-\beta}A^{-1} \end{bmatrix}.$$

If we write $\Delta(D^{-1}, D) = \sum_{j=-\mu}^{\mu} \Delta_j D^j$ then, by construction, all rows of Δ_j, and hence all columns of P_j, will have an even number of nonzero entries, for $-\mu \le j \le \mu$. □

Example 4.11. *Take A as in Example 4.8, $\Gamma = I_8$ the identity matrix of size 8, and $\beta = 2$. Then,*

$$T(D^{-1}, D) = \left[\begin{array}{c|c} A & B \\ \hline A & A \end{array}\right] = \left[\begin{array}{cccc|cccc} 2D & 2 & D^{-2} & 0 & D & 1 & 2D^{-2} & 0 \\ 0 & 1 & 0 & D & 0 & 2 & 0 & 2D \\ 0 & 0 & 2D^{-2} & 2D & 0 & 0 & D^{-2} & D \\ 0 & 0 & 0 & D & 0 & 0 & 0 & 2D \\ \hline 2D & 2 & D^{-2} & 0 & 2D & 2 & D^{-2} & 0 \\ 0 & 1 & 0 & D & 0 & 1 & 0 & D \\ 0 & 0 & 2D^{-2} & 2D & 0 & 0 & 2D^{-2} & 2D \\ 0 & 0 & 0 & D & 0 & 0 & 0 & D \end{array}\right].$$

In this case

$$P(D^{-1}, D) = \left[\begin{array}{cccc|cccc} D^{-1} & D^{-1} & D^{-1} & 0 & D^{-1} & D^{-1} & D^{-1} & 0 \\ 0 & 2 & 0 & 1 & 0 & 2 & 0 & 1 \\ 0 & 0 & D^2 & D^2 & 0 & 0 & D^2 & D^2 \\ 0 & 0 & 0 & 2D^{-1} & 0 & 0 & 0 & 2D^{-1} \\ \hline 2D^{-1} & 2D^{-1} & 2D^{-1} & 0 & D^{-1} & D^{-1} & D^{-1} & 0 \\ 0 & 1 & 0 & 2 & 0 & 2 & 0 & 1 \\ 0 & 0 & 2D^2 & 2D^2 & 0 & 0 & 2D^2 & 2D^2 \\ 0 & 0 & 0 & D^{-1} & 0 & 0 & 0 & 2D^{-1} \end{array}\right].$$

4.3 Attacks Against the Proposed Cryptosystem

In this section, we investigate possible attacks to the proposed cryptosystem, and show how the properties analyzed in previous sections provide increased key security with respect to previous variants of the McEliece cryptosystem. Thus, such scheme allows the use of GRS codes, and therefore reduce the key size of the public key.

We will consider the structural and ISD attacks. Each one of these classes of attacks will be analysed considering two different situations. One when the attacker deals with the whole message, and another when only truncated parts (intervals) of the sequence that forms the message are considered. This is explained next.
The ciphertext is generated as $\mathbf{y}(D) = \mathbf{u}(D)G'(D) + \mathbf{e}(D)$ or equivalently,

$$\left[\begin{array}{ccccccc} \mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_s & \cdots & \mathbf{y}_{s+\mu+\nu} \end{array}\right] \tag{4.12}$$

is equal to the multiplication of

$$\left[\begin{array}{cccc} \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_s \end{array}\right]$$

with

$$\begin{bmatrix} G'_0 & G'_1 & \cdots & G'_{\mu+\nu} & & & & & \\ & G'_0 & G'_1 & \cdots & G'_{\mu+\nu} & & & & \\ & & \ddots & \ddots & & \ddots & & & \\ & & & G'_0 & G'_1 & \cdots & G'_{\mu+\nu} & & \\ & & & & \ddots & \ddots & & \ddots & \\ & & & & & G'_0 & G'_1 & \cdots & G'_{\mu+\nu} \end{bmatrix} \tag{4.13}$$

and by then adding the error vector

$$\begin{bmatrix} \mathbf{e}_0 & \mathbf{e}_1 & \cdots & \mathbf{e}_{s+\mu+\nu} \end{bmatrix}.$$

An attacker could consider the full row rank matrix in (4.13) as the generator matrix of a block code. Note, however, that the size of this block code is $k(s+1) \times n(s+1+\mu+\nu)$ and therefore it will be very large for large values of s, even if n, k and μ are small. Precise numerical examples are provided in the next section. Notice that, to decode, Bob uses just matrices of size $k \times n$.

Another possibility is to consider an interval of the ciphertext sequence in (4.12) instead of the whole sequence. However, convolutional codes have *memory* and therefore each $\mathbf{y}_i$ depends on previous data. In order to discover $\mathbf{y}_i$ one needs to compute previous $\mathbf{y}_j$, i.e., one must estimate the state of the convolutional code at time instant i.

Thus, it seems natural to try to attack the first vectors $[\mathbf{y}_0\ \mathbf{y}_1\ \cdots\ \mathbf{y}_\sigma]$ for different values of $\sigma \le s$, as the initial state is assumed to be zero. Indeed, this possible weakness when using convolutional codes in the McEliece cryptosystem was already pointed out in [39]. For these reasons we are particularly interested in the security of the following interval of data

$$\begin{bmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_\sigma \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_\sigma \end{bmatrix} \begin{bmatrix} G'_0 & G'_1 & \cdots & G'_\sigma \\ & G'_0 & \cdots & G'_{\sigma-1} \\ & & \ddots & \vdots \\ & & & G'_0 \end{bmatrix} + \begin{bmatrix} \mathbf{e}_0 & \mathbf{e}_1 & \cdots & \mathbf{e}_\sigma \end{bmatrix} \tag{4.14}$$

for $\sigma \le s$ and taking, obviously, $G'_i = 0$ for $i > \mu + \nu$. Note that s can be taken to be very large without increasing the size of the public key nor the operations performed at each time instant.

4.3.1 Structural attacks

One general observation about the security of the proposed cryptosystem is about the way the public key is generated. As opposed to previous McEliece schemes, $G'(D) = S(D)GP(D^{-1}, D)$ is constructed using large parts randomly generated, and this makes, *a priori*, structure attacks more difficult. Effectively, the coefficients of $S(D)$ are totally randomly generated except for the S_μ, which is required to be invertible. Moreover, we will show next that, by counting the number of their inverses $T(D^{-1}, D)$, the set of admissible $P(D^{-1}, D)$ is considerably large, and therefore $P(D^{-1}, D)$ also introduces additional randomness into the system.

Lemma 4.12. *Let $A = [a_{i\ell}]$, $\widetilde{n}$ and $\widetilde{d}_j$, for $-\mu \leq j \leq \mu$, as in Subsection 4.2.1. Let $\widehat{d}_{i,j}$ be the number of elements in the main diagonal with the power D^j in the first i rows, and suppose j_i is such that $a_{ii} = \beta_i D^{j_i}$, for $1 \leq i \leq \widetilde{n}$ and $-\mu \leq j \leq \mu$. Then the number of possible $T(D^{-1}, D)$ matrices is*

$$\frac{(q-2)(q-1)^{\widetilde{n}} n!\, \widetilde{n}!}{\displaystyle\prod_{j=-\mu}^{\mu} \widetilde{d}_j!} \prod_{i=1}^{\widetilde{n}} \prod_{\substack{j=-\mu \\ j \neq j_i}}^{\mu} \big((\widetilde{d}_j - \widehat{d}_{i,j})(q-1) + 1\big). \tag{4.15}$$

in particular, the number of possible $T(D^{-1}, D)$ matrices is always greater than

$$\frac{(q-2)(q-1)^{\widetilde{n}} n!\, \widetilde{n}!}{\displaystyle\prod_{j=-\mu}^{\mu} \widetilde{d}_j!} \tag{4.16}$$

Proof. The number of possibilities for the principal diagonal of A satisfying Condition *2.* of Properties 4.7 is

$$\frac{(\widetilde{n})!}{\displaystyle\prod_{j=-\mu}^{\mu} \widetilde{d}_j!}(q-1)^{\widetilde{n}},$$

where we used the formula for the number of permutations with repetitions and since, for $1 \leq i \leq \widetilde{n}$, the β_i can be any element of $\mathbb{F}_q \setminus \{0\}$.

Although the number of possibilities for A to satisfy Conditions 3. and 4. of Properties 4.7 vary widely depending on the arrangement of the elements of the main diagonal of A, it is easy to see that in the row i we have $\widetilde{d}_j - \widehat{d}_{i,j}$ places after a_{ii} to put at most one element of the form γD^j in them, for $j \in \{-\mu, \ldots, 0, \ldots, \mu\} \setminus \{j_i\}$ such that $\gamma \in \mathbb{F}_q \setminus \{0\}$. We must consider also the possibility that no element of the form γD^j appears with $\gamma \neq 0$ for each j. Therefore there are

$$\prod_{i=1}^{\widetilde{n}} \prod_{\substack{j=-\mu \\ j \neq j_i}}^{\mu} \left((\widetilde{d}_j - \widehat{d}_{i,j})(q-1) + 1 \right) \tag{4.17}$$

different possibilities above the main diagonal, for each one of the possible main diagonals we can take satisfying the condition 2.. Now, the number of different matrices $T(D^{-1}, D)$ that we can obtain with this construction is equal to $(q-2)n!$ times the number of different matrices A. Therefore, the number of possible matrices $T(D^{-1}, D)$, is (4.15). □

Example 4.13. *Suppose $n = 16$, $q = 17$, $\mu = 1$, $d_{-1} = d_1 = 6$ and $d_0 = 4$. If in the main diagonal of A we have the sequence of powers*

$$(D^{-1}, D, 1, D^{-1}, D, 1, D^{-1}, D),$$

then by (4.17) we have

$$(2 \cdot 3)(2 \cdot 2)(2 \cdot 2)(1 \cdot 2)(1 \cdot 1)(1 \cdot 1)(1) = 192$$

different possibilities of arranging the elements above the diagonal. But if in the main diagonal of A we have the sequence of powers

$$(1, 1, D^{-1}, D^{-1}, D^{-1}, D, D, D),$$

then we have

$$(3 \cdot 3)(3 \cdot 3)(3)(3)(3) = 2187$$

different possibilities.

Lemma 4.14. *The number of possible $S(D)$ matrices is*

$$q^{k^2(\nu-\mu)} \prod_{j=1}^{k-1} (q^k - q^j).$$

Proof. In $S(D) = \sum_{i=\mu}^{\nu} S_i D^i$ we have $\nu - \mu + 1$ matrices S_i. While S_μ must be invertible in order to decode $\mathbf{u}(D)$, we impose no conditions over the other $\nu - \mu$ matrices S_i. The number of invertible $k \times k$ matrices defined over $\mathbb{F}_q$ is

$$\prod_{j=0}^{k-1} (q^k - q^j),$$

while the number of $k \times k$ matrices over $\mathbb{F}_q$ with no restrictions is q^{k^2}. Hence, we obtain the stated result. □

The number of possible matrices $S(D)$ is really large even for small values of k, but considering $S(D)$ in such generality will yield large private keys. One way of reducing the size of the private keys is by considering sparse matrices. The next result gives a subfamily of the matrices $S(D)$ that can be used for this purpose.

Lemma 4.15. *Suppose $S(D)$ is a block diagonal matrix with r blocks, i.e.*

$$S(D) = \left(\begin{array}{c|c|c|c} \Sigma_1(D) & 0 & \cdots & 0 \\ \hline 0 & \Sigma_2(D) & \cdots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & \Sigma_r(D) \end{array}\right),$$

where $r \mid k$ and let $\epsilon = k/r$. Then each row of $S(D)$ has at most ϵ nonzero entries and the number of possible $S(D)$ matrices of this form is

$$q^{\epsilon k(\nu-\mu)} \left(\prod_{j=0}^{\epsilon-1} (q^{\epsilon} - q^{j}) \right)^{r}.$$

Proof. Follows immediately from the previous Lemma. □

Next we look at possible structural attacks to the truncated sliding generator matrix $G'_{truc}(\sigma)$, that can be decomposed as

$$\underbrace{\begin{bmatrix} S_\mu & S_{\mu+1} & \cdots & S_\nu & & & \\ & S_\mu & S_{\mu+1} & \cdots & S_\nu & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & S_\mu & \cdots & \cdots & S_\nu \\ & & & & \ddots & & \vdots \\ & & & & & \ddots & \vdots \\ & & & & & & S_\mu \end{bmatrix}}_{=:S_{truc}(\sigma)} \begin{bmatrix} G & & & \\ & G & & \\ & & \ddots & \\ & & & G \end{bmatrix} \times$$

$$\underbrace{\begin{bmatrix} P_{-\mu} & \cdots & P_0 & \cdots & P_\mu & & \\ & P_{-\mu} & \cdots & P_0 & \cdots & P_\mu & \\ & & \ddots & & \ddots & & \ddots \\ & & & \ddots & & \ddots & P_\mu \\ & & & & \ddots & \ddots & \vdots \\ & & & & & \ddots & P_0 \\ & & & & & \ddots & \vdots \\ & & & & & & P_{-\mu} \end{bmatrix}}_{=:P_{truc}(\sigma)},$$

where $S_{truc}(\sigma) \in \mathbb{F}_q^{(\sigma+1)k\times(\sigma+1)k}$ and $P_{truc}(\sigma) \in \mathbb{F}_q^{(\sigma+1)n\times(\sigma+1)n}$. As all the matrices are constant, then it may seem to describe a similar situation as the original McEliece PKC using block codes. Note, however, that $P_{truc}(\sigma)$ is neither a permutation nor an invertible matrix.

As discussed in [11, 32, 33] the algebraic structure of a generator matrix G of a GRS is removed in GP if the weight of the columns of P is larger than or equal to 2. Due to condition **d)** in Properties 4.4, this is the case of the nonzero columns of $P_{truc}(\sigma)$. Obviously, its zero columns will not reveal any structure of the matrix G either, and thus we conclude that the attacks presented in [19] and [16] will not succeed against the proposed scheme. Similarly, one can computed the parity-check matrix $H'(D)$ from $G'(D)$ via $H'(D)(G'(D))^T = 0$ and obtain that $H'(D) = H(T(D^{-1}, D))^T$, where H is the parity-check of G which is also of the form (4.9). Again, the attacks fail as the weight of the rows of $T_{truc}(\sigma)$ are zero or ≥ 2.

4.3.2 Plaintext recovery

The plaintext recovery attacks try to decode a random linear code without requiring any knowledge of the secret key. However, trying to decode directly a convolutional code using ML decoding (Maximum-Likelihood decoding), e.g., the Viterbi decoding algorithm (see [58]), seems very difficult due to the large number of trellis transitions (its computational complexity increases exponentially with the increasing constraint length, $\nu + \mu$ in our case) .

The plaintext recovery type of attack is typically performed using information set decoding algorithms (ISD).

Next, we analyse how Stern ideas could be adapted to our context. Consider the full row rank matrix in (4.13) and ISD attacks using, for instance, the Stern algorithm [50].

A detailed description of the algorithm for codes over arbitrary finite fields can be found in [50, Sec. 3] by C. Peters. In the same paper, Peters also presents a work factor estimate. For $\mathbb{F}_q$ with $q = 2^m$, r the number of errors, G a $k \times n$ matrix and the parameters p and ℓ,

$$\mathrm{WF}_{p,\ell} = m\mathcal{S}_{p,\ell}\frac{\binom{n}{r}}{\binom{n-k-\ell}{r-2p}\binom{\frac{k}{2}}{p}^2}, \tag{4.18}$$

where

$$\begin{aligned}\mathcal{S}_{p,\ell} =&(n-k)^2(n+k) + \ell\left(\frac{k}{2} - p + 1 + 2\binom{\frac{k}{2}}{p}(q-1)^p\right)\\ &+ \frac{2pq(r-2p+1)(2q-3)(q-1)^{2p-2}}{q^\ell}\binom{\frac{k}{2}}{p}^2.\end{aligned}$$

The parameters p and ℓ have to be determined to minimize $\mathrm{WF}_{p,\ell}$. A few improvements were made to this algorithm (see for example [23, 31]) but as was pointed out in [11] new information set decoding algorithms will not significantly change the security of McEliece PKC and its variants.

Let $k_s = \mathrm{rank}(G'_{truc}(s))$ and select a size-k_s information set $I = \{a_1, \dots, a_{k_s}\} \subset \{1, \dots, k(s+1)\}$ and let t_s be the maximum number of errors that the code can tolerate within the time interval $[a_1, \dots, a_{k_s}]$. Let $P(k, n, t)$ be the probability of determining the secret message when using a (n, k) block code with error-correcting capability t in the classical McEliece PKC. Then, it follows that the probability of recovering $\widehat{\mathbf{u}}$ in (4.14) is

$$\frac{1}{q^{k(s+1)-k_s}}P(k_s, n(s+1), t_s).$$

4.4 Improvements: higher rates and message length reduction

In order to protect this PKC against structural attacks, we considered the column weight of each P_j, with $-\mu \leq j \leq \mu$, to be at least ρ, but this implies that at each consecutive $2\mu+1$ instants, we can only send t/ρ errors. One consequence will be that if we want to protect it against ISD attacks (at time instant $\sigma = s$, when $G'_{truc}(s)$ is full rank) we need to send large amount of information with very poor rate. In this section, we develop a new method of sending the messages in such a way that the message length is reduced. This method will allow the use of more general parameters n and k and, therefore, higher rates.

Since our decoding is sequential, we do not need to receive the whole $\mathbf{y}(D)$ to start decoding. In reality we just need to receive $(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{2\mu+\sigma})$ in order to decode $(\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_\sigma)$. This will improve the performance of the scheme, but notice that to calculate $(\mathbf{y}_{\sigma+1}, \dots, \mathbf{y}_{2\mu+\sigma})$, Alice also needs to have $(\mathbf{u}_{\sigma+1}, \dots, \mathbf{u}_{2\mu+\sigma})$. Hence, Alice can proceed in the following way:
Alternative Encryption: Alice wants to send the message $(\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_\sigma)$ to Bob. She randomly generates the vector $(\mathbf{u}_{\sigma+1}, \dots, \mathbf{u}_{2\mu+\sigma})$, selects an error vector

$$\mathbf{e}(D) = \sum_{i=0}^{3\mu+\nu+\sigma} \mathbf{e}_i D^i$$

satisfying (4.4), and encrypts the message $\mathbf{u} = \sum_{i=0}^{2\mu+\sigma} \mathbf{u}_i D^i$ as

$$\mathbf{y}(D) = \mathbf{u}(D) G'(D^{-1}, D) + \mathbf{e}(D) = \sum_{i=0}^{3\mu+\nu+\sigma} \mathbf{y}_i D^i. \tag{4.19}$$

Now, instead of sending $\mathbf{y}(D)$, Alice just sends

$$\widetilde{\mathbf{y}}(D) = \sum_{i=0}^{2\mu+\sigma} \mathbf{y}_i D^i.$$

Note that in this modified scheme the parameter σ has to be part of the public key. As was noticed in Remark 4.6, the rank of $G'_{truc}(\sigma)$ is less than or equal to $\sigma k + d_\mu$, therefore, $G'_{truc}(\sigma)$ is not full rank. If q^{k-d_μ} is sufficiently large, an attacker cannot recover $\mathbf{u}(D)$.

Remark 4.16. *Note that the security of this alternative encryption heavily relies on the rank deficiency of $G'_{truc}(\sigma)$. However, the addition of errors increases the security and moreover avoids the right inversion of the complete sliding generator matrix of G', as described in equation (4.13), by an attacker.*

The process just described allows Alice to send much smaller messages and still be confident that they are protected. But Alice needs to send $(2\mu + \sigma + 1)n$ elements of $\mathbb{F}_q$ if her message is put in $(\sigma + 1)k$ elements of $\mathbb{F}_q$, so the rate of transmission is reduced. But since with this alternative process of encryption we don't need to worry about ISD attacks on full rank matrices, we can have larger rates k/n, specially considering d_μ very low, but sufficiently large so that there are enough randomly generated T_μ.

In the next section, we exhibit a few concrete examples where the advantages of this PKC can be seen.

4.5 Examples

We will now explain in detail the construction of two specific examples. We finish this section displaying a table with thess and other examples with different parameters, where we explore different work-factors and different sizes of the plaintext $\mathbf{u}$.

Example 4.17. *We choose the field $\mathbb{F}_q$ with $q = 2^8$. Let $m = \log_2(q) = 8$. Our generator matrix G is the $k \times n$ generator matrix of a GRS over $\mathbb{F}_q$ of size $n = 180$ and dimension $k = 96$. Therefore, we can correct up to $t = \left\lfloor \frac{n-k}{2} \right\rfloor = 42$ errors. In order to construct our $G'(D)$, we fix $\mu = 1$, $\nu = 2$ and $(d_{-1}, d_0, d_1) = (60, 60, 60)$, so, clearly, condition (4.10) is satisfied. With these parameters, we construct the matrices $S(D)$, $T(D^{-1}, D)$ and $P(D^{-1}, D)$ as described in Section 4.2 and compute*

$$G'(D) = S(D)GP(D^{-1}, D).$$

To reach 2^{256}-bit security level against ISD attacks using the full rank matrix in (4.13), we need to send $\mathbf{u}(D) = \sum_{i=0}^{s} \mathbf{u}_i D^i$ with $s \geq 30$. Taking $t = 42$ and $s = 30$, the value

$$t' = \left\lfloor \frac{t(s+1+\mu+\nu)}{2(2\mu+1)} \right\rfloor = 238,$$

represents the total number of errors Alice will send during the transmission of all $\mathbf{u}(D)$. This means that Alice sends the errors in such a way that the sum of the weights of three consecutive errors is 21, i.e., can send about $t^* = \frac{238}{34} = 7$ errors at each instant. Hence each $\widehat{\boldsymbol{y}}_j$ has at most 42 errors and can be always corrected.

The parameters $t/2$, s are public and the $G'(D)$ is part of the public key and of size (in bits)

$$m \cdot n \cdot k \cdot (\mu + \nu + 1) = 8 \cdot 180 \cdot 96 \cdot (1 + 2 + 1) = 552960.$$

The private key ($S(D)$, G, $T(D, D^{-1})$) can be stored (in bits) using

$$\begin{aligned}\underbrace{m \cdot (\nu - \mu + 1) \cdot k^2}_{S(D)} + \underbrace{m \cdot n \cdot k}_{G} + \underbrace{m \cdot (2\mu + 1) \cdot n^2}_{T(D,D^{-1})} &= \\ &= 8 \cdot 2 \cdot 96^2 + 8 \cdot 180 \cdot 96 + 8 \cdot 3 \cdot 180^2 \\ &= 1063296.\end{aligned}$$

This key can be reduced considering the following facts. We can store G as two vectors α, $\mathbf{x}$ such that $\mathrm{GRS}_{n,k}(\alpha, \mathbf{x})$ is the generalized Reed-Solomon code generated by G, so we only need to store $2 \cdot m \cdot n = 2880$ bits for G. The way to store $T(D, D^{-1})$ is done by storing the information of Γ and $\Delta(D, D^{-1})$. Since Γ is a permutation matrix, we can represent it as a permutation vector $\mathbf{w} \in \{1, 2, \ldots, n\}^n$. Since $n = 180$ and we only need $\lfloor \log_2(180) \rfloor + 1 = 8$ bits to store each component, we store Γ using $8 \cdot 180 = 1440$ bits. Now, we explain how to store $\Delta(D, D^{-1})$. The main diagonal of the matrix $A = A(D, D^{-1})$ can be seen as $A = \widetilde{\Gamma}^T \, \Pi \, \widetilde{\Gamma} \, \Theta$, where $\widetilde{\Gamma}$ is a permutation matrix of order $\widetilde{n} = n/2$, Π is a diagonal matrix of order $\widetilde{n}$ having the first 60 entries equal to D^{-1}, the next 60 entries equal to 1 and the last 60 entries equal to D, and Θ is a diagonal matrix with β_i in row i, for $1 \leq i \leq \widetilde{n}$. Since $\lfloor \log_2(90) \rfloor + 1 = 7$, to store this diagonal we just need to store $7 \cdot 90 = 630$ bits for $\widetilde{\Gamma}$, since $\lfloor \log_2(60) \rfloor + 1 = 6$ we need $6 \cdot 3 = 18$ bits for Π, since we just need to store the partition $(\widetilde{d}_{-1}, \widetilde{d}_0, \widetilde{d}_1) = (60, 60, 60)$, and $8 \cdot 90 = 720$ bits for Θ. To finish with the matrix A we still need to store which and where to put the nonzero elements above the main diagonal. Since in each row we put at most two nonzero elements and they are of the form an element of $\mathbb{F}_q$ times D^{-1}, times 1 or times D, we can just use $7 \cdot 7 = 49$ bits for each position and $8 + 2 = 10$ bits for each element (8 bits for each $\gamma \in \mathbb{F}_q$ and it could be 10 for D^{-1}, 00 for 1 and 01 for D). So we have $2 \cdot 49 \cdot 10 = 980$ bits. Therefore, to store $\Delta(D, D^{-1})$ we need $630 + 18 + 720 + 980 = 2348$ bits. We also need to store $S(D)$, which is a matrix of order 96. Then, to store the private key we need at most (in bits)

$$\underbrace{8 \cdot 2 \cdot 96^2}_{S(D)} + \underbrace{2880}_{G} + \underbrace{1440 + 2348}_{T(D,D^{-1})} = 154124.$$

Notice that 95% of the private key is in $S(D)$. We can reduce it tremendously if instead of having completely random matrices $S(D)$, we consider matrices with many zeros in prescribed places. Using the construction of $S(D)$ given in Lemma 4.15 with $r = 12$ blocks and where each row has at most $\epsilon = 8$ nonzero entries, we just need

$$\underbrace{m \cdot (\nu - \mu + 1) \cdot r \cdot \epsilon^2}_{S(D)} = 8 \cdot 2 \cdot 12 \cdot 8^2 = 12288$$

bits to store $S(D)$. Therefore, to store the private key we need 18956 *bits instead. The length of the message (in bits) that Alice would send, if she sends all* $\boldsymbol{y}(D)$ *is*

$$m \cdot n \cdot (s + 1 + \mu + \nu) = 8 \cdot 180 \cdot (30 + 1 + 1 + 2) = 48960.$$

But if she prefers to send a smaller message, she can use the alternative encryption algorithm explained in Section 4.4. If she takes $\sigma = 5$*, her message* $\mathbf{u}(D)$ *would only have* $mk(\sigma + 1) = 8 \cdot 96 \cdot (5 + 1) = 4608$ *bits, and she would send* $mn(\sigma + 3) = 11520$ *bits. In this case the rate of transmission is* 0.4*. The original McEliece scheme has a communication rate around 0.5.*

Comments on security:

One can think that with such a relatively small field of only $q = 2^8$ *elements and with our particular construction of* Δ*, which has many constraints, it is feasible to consider a brute force attack to construct* $T(D^{-1}, D)$*. However, using (4.16) (instead of (4.15)) we have that the total number of* $T(D^{-1}, D)$ *is greater than*

$$\begin{aligned} (q-2)n!\frac{(\widetilde{n})!}{\prod_{i=-\mu}^{\mu} \widetilde{d}_i!}(q-1)^{\widetilde{n}} &= 254 \cdot 180! \cdot \frac{90!}{(30!)^3} \cdot 255^{90} \\ &> 2^{1463.99} \end{aligned}$$

The total number of matrices $S(D)$, using the construction given in Lemma 4.15, with $r = 12$ *and* $\epsilon = 8$ *is*

$$\begin{aligned} q^{\epsilon k(\nu-\mu)} \left(\prod_{j=0}^{\epsilon-1} (q^{\epsilon} - q^j) \right)^r &= 2^{8 \cdot 8 \cdot 96(2-1)} \left(\prod_{j=0}^{7} \left(2^{8 \cdot 8} - 2^{8j}\right) \right)^{12} \\ &= 2^{12287.93} \end{aligned}$$

If we try to attack the truncated matrix $G'(D)$, from Remark 4.6 we conclude that the work factor is

$$\mathrm{WF} \geq q^{k - \mathrm{rank}(G'_0)} = 2^{8(96-60)} = 2^{288}.$$

Hence, one can try an Information Set Decoding algorithm over the full rank matrix. Using formulas from [11] and [50], (see Subsection 4.3.2) we have that the work factor for the full rank matrix is

$$\mathrm{WF} \geq 2^{257.249}.$$

Although the reported WFs are the lowest ones obtained for each set of parameters, it is important to remark that we are not using the optimized version of ISD or new versions that take into consideration the constrains of the error distribution in this system, so this work factor may be smaller. If this is the case, we can take a bigger s (for example, for $s = 32$ *the work factor increases to* $2^{272.909}$*). If we use the alternative encryption proposed in Section 4.4, we never have full rank matrices, so the work factor comes always from the rank deficiency, which is* 2^{288}.

Next, we exhibit an example that reduces the public key enormously, taking into account that the d_i's do not need to be all equal. Note that decreasing the value of d_1 increases the WF, but the scheme may be susceptible to attacks that take into consideration the fact that the number of possible matrices T_1 (and so P_{-1}) is reduced.

Example 4.18. *Take the field* $\mathbb{F}_q$ *with* $q = 2^6$*, so* $m = 6$*. Take G a generator matrix of a GRS over* $\mathbb{F}_q$ *of size* $n = 2^6 - 2 = 62$ *and dimension* $k = 30$*. Here* $t = 16$*. Take* $\mu = 1$*,* $\nu = 2$ *and* $(d_{-1}, d_0, d_1) = (8, 46, 8)$*. In order to be protected against ISD attacks over the full rank matrix in (4.13) we need to take* $s = 46$*. The total number of errors is* $t' = 133$ *and the sum of the weights of three consecutive errors is* 8*. The size of the public key is (in bits)*

$$m \cdot n \cdot k \cdot (\mu + \nu + 1) = 6 \cdot 62 \cdot 30 \cdot (1 + 2 + 1) = 44640.$$

If we consider $S(D)$ *to be a block diagonal matrix with* 6 *blocks, then the size of the private key is (in bits)*

$$\underbrace{6 \cdot 2 \cdot 6 \cdot 5^2}_{S(D)} + \underbrace{744}_{G} + \underbrace{372 + 966}_{T(D,D^{-1})} = 3882.$$

The length of the message (in bits) is

$$m \cdot n \cdot (s + 1 + \mu + \nu) = 6 \cdot 62 \cdot (46 + 1 + 1 + 2) = 18600.$$

This value can be reduced by sending a smaller message using the alternative encryption scheme. If Alice message $\mathbf{u}$ *only has* $mk(\sigma + 1) = 6 \cdot 30 \cdot (5 + 1) =$ 1080 *bits, she would send* $mn(\sigma + 3) = 2976$ *bits. In this case the rate of transmission is about* 0.363*.*

Comments on security:

In this case, the total number of $T(D^{-1}, D)$ *is greater than*

$$\begin{aligned}(q-2)n!\frac{\widetilde{n}!}{\prod_{i=-\mu}^{\mu}\widetilde{d}_i!}(q-1)^{\widetilde{n}} &= 60\cdot 62!\frac{31!}{4!\cdot 23!\cdot 4!}\cdot 63^{31}\\ &> 2^{504.26}\end{aligned}$$

The total number of matrices $S(D)$*, using the construction given in Lemma 4.15, with* $r = 6$ *and* $\epsilon = 5$ *is*

$$\begin{aligned}q^{\epsilon k(\nu-\mu)}\left(\prod_{j=0}^{\epsilon-1}(q^{\epsilon}-q^{j})\right)^{r} &= 2^{6\cdot 5\cdot 30(2-1)}\left(\prod_{j=0}^{4}\left(2^{6\cdot 5}-2^{6j}\right)\right)^{6}\\ &= 2^{1799.86}\end{aligned}$$

If we try to attack the truncated matrix $G'(D)$*, from Remark 4.6 we conclude that the work factor is*

$$\mathrm{WF} \geq q^{k-\mathrm{rank}(G_0')} = 2^{6(30-8)} = 2^{132}.$$

The Information Set Decoding algorithm over the full rank matrix has a work factor

$$\mathrm{WF} \geq 2^{136.055}.$$

The table 4.1 shows various examples with different parameters, all with $\mu = 1$ and $\nu = 2$ and $d_i = n/3$, for $i \in \{-1, 0, 1\}$, except the first one which is explained in the previous example. The value s is the smallest for given parameters n and k that gives a WF full rank larger than 2^{128} or 2^{256}. In the second part of the table, it is assumed that the alternative encryption scheme was used and so we don't have a WF full rank. The third part of the table exhibits the parameters for the classic McEliece scheme using Goppa codes.

In the example with $n = 78$ and $k = 66$, the rate k/n is above 0.6 for a security of 2^{256} and a public key size of just 144144, but notice that if the errors are evenly distributed, each $\mathbf{e}_i$ has weight 1.

n	k	s	WF Truncated	WF Full Rank	Public Key
Main Scheme					
62	30	46	2^{132}	$2^{136.0}$	44640
126	66	20	2^{168}	$2^{128.4}$	232848
174	78	15	2^{160}	$2^{128.2}$	434304
264	108	10	2^{180}	$2^{128.7}$	1026432
180	96	30	2^{288}	$2^{257.2}$	552960
228	108	25	2^{256}	$2^{260.2}$	787968
254	122	22	2^{298}	$2^{257.7}$	991616
Alternative Scheme					
66	42		2^{140}		77616
78	54		2^{196}		117936
78	66		2^{280}		144144
138	78		2^{256}		344448
156	84		2^{256}		419328
132	108		2^{512}		456192
Classical McEliece with Goppa codes					
2960	2288			2^{128}	1537536
8192	6528			2^{256}	10862592

Table 4.1: Parameters, work factors and public key sizes

Chapter 5

Conclusions and future work

In this dissertation we have proposed a new variant of the McEliece cryptosystem using convolutional codes instead of block codes. This scheme is in many aspects different from the previous proposed variants as the message is not a block vector anymore but a stream sequence of vectors. Trying to adapt the existing attacks seems not straightforward. The class of PKC introduced in this work has many parameters that can be explored further.

Here, we highlight some remarks on the present work and on the research we plan to do in our future work:

1. In all examples we considered $\nu = 2$ which is a very simplified version of this class of cryptosystems, but notice that the public key only changes by a factor of $\frac{\nu+2}{4}$ if we consider a different ν, so in order to create a more convolutional version we just need to increase ν.

2. If we consider different values for the d_i, the public key size can be reduced, keeping the same security if one tries to attack the truncated matrices, as has been exemplified in the first detailed example.

3. We also plan to explore other constructions for $T(D^{-1}, D)$, satisfying the Properties 4.4.

4. Instead of using the encoder G' as in the McEliece cryptosystem, in the future we will explore the possibilities of using the associated parity check H' and create a convolutional version of the Niederreiter cryptosystem.

Bibliography

[1] P. Almeida, M. Beltrá, D. Napp, and C. Sebastião. Smaller keys for code-based cryptography: McEliece cryptosystems with convolutional encoders. *arXiv, submitted for publication in the IEEE Transactions on Information Theory*, https://arxiv.org/abs/2104.06809, 2021.

[2] P. Almeida, J. Brandão, D. Napp, and C. Sebastião. Convolutional code-based cryptosystem. In *Proceedings of the meeting of the Thematic Network of Linear Algebra, Matrix Analysis and Applications (ALAMA), Alicante (Spain)*, volume ISBN: 978-84-16724-96-3, pages 117–119, 2018.

[3] P. Almeida, J. Brandão, D. Napp, and C. Sebastião. Cryptography based on convolutional codes. In *Workshop on Graph Spectra, Combinatorics and Optimization, Aveiro (Portugal), January 25-27, 2018*, 2018.

[4] A. Badr, A. Khisti, W.-T. Tan, and J. Apostolopoulos. Layered constructions for low-delay streaming codes. *IEEE Transactions on Information Theory*, 63(1):111–141, 2017.

[5] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini. Ledakem: A post-quantum key encapsulation mechanism based on QC-LDPC codes. In T. Lange and R. Steinwandt, editors, *Post-Quantum Cryptography*, pages 3–24. Springer International Publishing, 2018.

[6] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, and D. Schipani. Enhanced public key security for the McEliece cryptosystem. *Journal of Cryptology*, 29(1):1–27, 2016.

[7] T. Berger and P. Loidreau. How to mask the structure of codes for a cryptographic use. *Designs, Codes and Cryptography*, 35:63–79, 2005.

[8] E. R. Berlekamp. Goppa codes. *IEEE Trans. Inf. Th*, 19(5):590–592, 1973.

[9] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inf. Th*, 24(3):384–386, 1978.

[10] D. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *Post-Quantum Cryptography*, pages 31–46, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[11] J. Bolkema, H. Gluesing-Luerssen, C. A. Kelley, K. E. Lauter, B. Malmskog, and J. Rosenthal. Variations of the McEliece cryptosystem. In J. W. E. Howe, K. Lauter, editor, *Algebraic Geometry for Coding Theory and Cryptography. Association for Women in Mathematics Series*, volume 9. Springer Cham., 2017.

[12] I. Cascudo, R. Cramer, D. Mirandola, and G. Zémor. Squares of random linear codes. *IEEE Transactions on Information Theory*, 61(3):1159–1173, 2015.

[13] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. Report on post-quantum cryptography. Report NISTIR 8105, National Institute of Standards and Technology (NIST), https://csrc.nist.gov/publications/detail/nistir/8105/final, February 2016.

[14] R. W. Cottle. Manifestations of the Schur complement. *Linear Algebra Appl.*, 8:189–211, 1974.

[15] A. Couvreur, P. Gaborit, V. Gauthier-Umaña, A. Otmani, and J.-P. Tillich. Distinguisher-based attacks on public-key cryptosystems using reed-solomon codes, 2014.

[16] A. Couvreur and M. Lequesne. On the security of subspace subcodes of Reed-Solomon codes for public key encryption. *arXiv: https://arxiv.org/abs/2009.05826*, 2020.

[17] A. Couvreur, I. Márquez-Corbella, and R. Pellikaan. Cryptanalysis of McEliece cryptosystem based on algebraic geometry codes and their subcodes. *IEEE Transactions on Information Theory*, 63(8):5404–5418, 2017.

[18] A. Couvreur, A. Otmani, and J.-P. Tillich. Polynomial time attack on wild McEliece over quadratic extensions. *IEEE Transactions on Information Theory*, 63(1):404–427, 2017.

[19] A. Couvreur, A. Otmani, J.-P. Tillich, and V. Gauthier-Umana. A polynomial-time attack on the BBCRS scheme. *Conference Public Key Cryptography (PKC), 2015 https://arxiv.org/pdf/1501.03736.pdf*, 2015.

[20] M. Elia, E. Viterbo, and G. Bertinetti. Decoding of binary separable goppa codes using berlekamp-massey algorithm. *Electronics Letters*, 35:1720–1721(1), 1999.

[21] Elias. P. coding for noisy channels. *IRE Conv. Rec.*, 4:37–46, 1955.

[22] J. Faugere, V. Gauthier-Umaña, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high-rate McEliece cryptosystems. *IEEE Transactions on Information Theory*, 59(10):6830–6844, 2013.

[23] M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in cryptology ASIACRYPT - Lecture Notes in Computer Science*, volume 5912, pages 88–105. Springer International Publishing, 2009.

[24] F.J.Macwilliams and N. Sloane. *The Theory of Error-Correcting Codes*, volume 16. North-Holland, 1977.

[25] G. Forney. Convolutional Codes I: Algebraic Structure. *IEEE Trans. Inform. Theory*, 16:720–738, 1970. Correction, Ibid., IT-17,pp. 360, 1971.

[26] G. Forney. Structural Analysis of Convolutional Codes via Dual Codes. *IEEE Trans. Inform. Theory*, 19:512–518, 1973.

[27] V. Goppa. A new class of linear correcting codes. *Problemy Peredaci Informacii*, 6:24–30, 1970.

[28] J. I. Hall. Notes on coding theory. Chapter 5: GRS Codes. Department of Mathematics, Michigan State University. available online on. https://users.math.msu.edu/users/halljo/classes/codenotes/coding-notes.html, 2010.

[29] S. Heyse. Code-based cryptography: Implementing the mceliece scheme on reconfigurable hardware. diploma thesis, Faculty of Electrical Engineering and Information Technology, Ruhr-University Bochum, 2009.

[30] T. Hungerford. *Algebra.* Springer, eighth edition, 2003.

[31] C. Interlando, K. Khathuria, N. Rohrer, J. Rosenthal, and V. Weger. Generalization of the ball-collision algorithm. *Journal of Algebra Combinatorics Discrete Structures and Applications*, pages 197–209, 2020.

[32] K. Khathuria, J. Rosenthal, and V. Weger. Two masking of the Reed-Solomon structure in conjugation with list decoding. In *Proceedings of 23rd International Symposium on Mathematical Theory of Networks and Systems, Hong Kong University of Science and Technology, Hong Kong*, pages 309–314, 2018.

[33] K. Khathuria, J. Rosenthal, and V. Weger. Encryption scheme based on expanded Reed-Solomon codes. *Advances in Mathematics of Communications*, 15(2):207–218, 2021.

[34] K. Kimball. Announcing request for nominations for public-key post-quantum cryptographic algorithms. Report NISTIR 8105, National Institute of Standards and Technology (NIST), https://csrc.nist.gov/news/2016/public-key-post-quantum-cryptographic-algorithms, December 20 2016.

[35] G. Landais and J.-P. Tillich. An efficient attack of a McEliece cryptosystem variant based on convolutional codes. In P. Gaborit, editor, *Post-Quantum Cryptography*, pages 102–117. Springer Berlin Heidelberg, 2013.

[36] P. J. Lee and E. F. Brickell. An observation on the security of mceliece's public-key cryptosystem. In D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and C. G. Günther, editors, *Advances in Cryptology — EUROCRYPT '88*, pages 275–280, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[37] J. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.

[38] M. Loeloeian and J. Conan. A transform approach to goppa codes. *IEEE Transactions on Information Theory*, 33(1):105–115, 1987.

[39] C. Löndahl and T. Johansson. A new version of McEliece PKC based on convolutional codes. In T. W. Chim and T. H. Yuen, editors, *Information and Communications Security*, pages 461–470. Springer Berlin Heidelberg, 2012.

[40] F. J. MacWilliams and N. J. Sloane. *The Theory of Error-Correcting Codes.* North Holland, Amsterdam, 1977.

[41] I. Márquez-Corbella, E. Martínez-Moro, and R. Pellikaan. The non-gap sequence of a subcode of a generalized reed—solomon code. *Des. Codes Cryptography*, 66(1–3):317–333, 2013.

[42] I. Márquez-Corbella, E. Martínez-Moro, R. Pellikaan, and D. Ruano. Computational aspects of retrieving a representation of an algebraic geometry code. *Journal of Symbolic Computation*, 64:67–87, 2014. Mathematical and computer algebra techniques in cryptology.

[43] I. Márquez-Corbella and J.-P. Tillich. Using Reed-Solomon codes in the $(U|U+V)$ construction and an application to cryptography. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 930–934, 2016.

[44] R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, Jan. 1978.

[45] R. J. McEliece. The algebraic theory of convolutional codes. In V. Pless and W. Huffman, editors, *Handbook of Coding Theory*, volume 1, pages 1065–1138. Elsevier Science Publishers, Amsterdam, The Netherlands, 1998.

[46] H. Moufek and K. Guenda. A new variant of the mceliece cryptosystem based on the smith form of convolutional codes. *Cryptologia*, 42(3):227–239, 2018.

[47] N. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15:159–166, 1986.

[48] N. Patterson. The algebraic decoding of goppa codes. *IEEE Transactions on Information Theory*, 21(2):203–207, 1975.

[49] T. P.Berger and P. Loidreau. How to mask the structure of codes for a cryptographic use. *Designs, Codes and Cryptography*, 35(1):63–79, 2005.

[50] C. Peters. Information-set decoding for linear codes over $\mathbb{F}_q$. In N. Sendrier, editor, *Post-Quantum Cryptography*, pages 81–94, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[51] P. Piret. Structure and constructions of cyclic convolutional codes. *IEEE Trans. Inf. Th*, 22(2):147–155, 1976.

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

[53] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8, june 1960.

[54] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.

[55] I. Schur. Potenzreihen im Innern des Einheitskreises. *J. Reine Angew. Math.*, 147:205–232, 1917.

[56] V. M. Sidelnikov and S. O. Shestakov. On the insecurity of cryptosystems based on generalized reed-solomon codes. *Discrete Math. Appl.*, 2:439–444, 1992.

[57] J. Stern. A method for finding codewords of small weight. In G. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, pages 106–113, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

[58] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[59] Y. Wang. Quantum resistant random linear code based public key encryption scheme RLCE. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2519–2523, 2016.

[60] Y. Wang. First round of submission to the NIST post.quantum cryptography call. In *RLCE-KEM http://quantumca.org*, 2017.

[61] C. Wieschebrink. An attack on a modified niederreiter encryption scheme. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography - PKC 2006*, pages 14–26, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[62] C. Wieschebrink. How to mask the structure of codes for a cryptographic use. *Public-Key Cryptography-PKC*, 3958:14–26, 2006.

[63] C. Wieschebrink. Cryptanalysis of the niederreiter public key scheme based on grs subcodes. *IACR Cryptology ePrint Archive, Report*, 452, 2009.

[64] R. H. D. Yuan Xing Li and X. mei Wang. On the equivalence os mceliece's and niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40:271–273, 1994.