McEliece cryptosystem for post-quantum cryptography

Author:

Paul Bagourd

Semester project 1st year master

Supervised by:

Dr Vlad Serban, Chair of number theory

Report accepted on June 9, 2023



École Polytechnique Fédérale de Lausanne

Abstract

Since the 70s, public-key encryption schemes such as RSA or Diffie-Hellman as well as more elaborate cryptosystems have relied on the hardness of underlying mathematical problems. These include factoring large composite numbers or finding discrete logarithms in finite fields. However, the current progress in computer science and quantum computing has raised concerns about the robustness of those cryptosystems. In such considerations, other cryptosystems believed to resist the computational power of quantum technologies have been studied increasingly. One of them is the McEliece cryptosystem, we study this cryptosystem here.

Contents

1 Introduction **3**

2 Introduction to coding theory **4**
2.1 Algebraic coding theory 4
2.2 Linear codes 4
2.3 Generator and check matrix 5
2.4 Decoding techniques for linear codes 6
2.5 The Singleton bound 8

3 McEliece cryptosystem **9**
3.1 Complexity of some code-based problems 9
3.2 Security level 10
3.3 McEliece cryptosystem 11
3.4 A first discussion on security aspects 12
3.5 Generalized Reed-Solomon codes 13
3.6 Alternant codes 14
3.7 Goppa codes 14
3.8 Algebraic Geometry codes 17
3.9 Patterson's algorithm 19

4 Some attacks on the McEliece cryptosystem **22**
4.1 A classical structural attack 22
4.2 Decoding attacks 28
4.3 Information set decoding 28
4.4 Making use of Grover's quantum algorithm 30
4.5 Quantum Fourier Sampling attack 32

5 Conclusion **33**

6 Acknowledgements **33**

1 Introduction

This report studies the McEliece cryptosystem introduced by Robert McEliece in 1978. For now, it remains unbroken despite various attacks and is the only code-based cryptosystem retained by the NIST institution for post-quantum cryptography. It is a public-key cryptosystem or equivalently an asymmetric encryption scheme. An asymmetric encryption scheme relies on mathematical problems computationally difficult to solve in the general case but efficiently solvable with the knowledge of a trapdoor such as a one-way function kept secret in a so-called private key. For example, the RSA cryptosystem relies on the hardness of the factoring problem. However, this cryptosystem could become computationally insecure as a quantum computer can theoretically solve it in polynomial time by harnessing the laws of quantum mechanics, essentially superposition and entanglement, performing probabilistic computations [10].

In the first section, we explain the fundamental aspects of coding theory, a theory that was initially built to correct message errors, but also used to create cryptosystems such as the McEliece cryptosystem. We then analyze the McEliece cryptosystem and some interesting families of codes to which it can be applied. We focus on GRS codes and Goppa codes, which have interesting properties in terms of error correction and information rate. Finally, we present some important attacks and discuss their consequences on the security of McEliece cryptosystem. We study the Sidelnikov-Sheskatov attack that works well for GRS codes but does not seem suitable for Goppa codes. We try to understand why there is such a gap, having in mind that Goppa codes are subfield subcodes of GRS codes. We then investigate a kind of brute-force attack called the information set decoding attack. This attack is interesting in the clever ideas it proposes to reveal the information hidden in the cyphertext. However, its running time is super polynomial and cannot be used in practice, even by mixing it with Grover's quantum algorithm. We briefly mention another quantum-related result explaining why a class of quantum algorithms fails to break McEliece cryptosystem, which comforts us to implement it for post-quantum cryptography.

2 Introduction to coding theory

Coding theory is the part of computer science that studies codes. A code is a transformation of a sequence of symbols to achieve various purposes. This discipline has many applications, from data compression to efficient and secure communications. Let us explain the mathematical background surrounding it. In the following we were inspired by chapters 2&4 of the book "Coding theory: a first course" by S Ling and C Xing [6].

2.1 Algebraic coding theory

A code is an injective function $\mathcal{C} : \mathcal{X} \to \Sigma^*$ where Σ^* is a set of finite-length strings of symbols from a Σ-ary alphabet. This way, $C(x)$ denotes the codeword associated with x. In the context of data compression, data being encoded can be seen as a random variable $X : \Omega \to \mathcal{X}$ and codes can be used to compress information carried by the data.
Usually, coding involves a sequence of symbols that is divided into blocks of same length to which a transformation is applied. The term algebraic coding theory denotes the subfield of coding theory where the properties of codes are expressed in algebraic terms, from which follows mathematical research and improvements. Mathematically speaking, this means that the sets $\mathcal{X}$ and Σ are algebraic fields.

2.2 Linear codes

Historically, linear codes were introduced in the field of communication systems to overcome transmission errors when communicating through a channel. Different types of errors exist due to noise and disturbance, in this project we introduce the linear codes which correct changes of symbols in the message sequence. We will later see how this can be used for a cryptosystem.

Let $n, k, q \in \mathbb{N}^*$ with $k \leq n$ and $q \geq 2$.

Definition 2.1 ($[n,k]_q$ linear code)**.** *A linear code of length n and dimension k is a linear map $\mathcal{C} : \mathbb{F}_q^k \to \mathbb{F}_q^n$.*

Intuitively, a linear code is a code for which the sum of two codewords is again a codeword, a property of vector spaces.

Remark 2.2. *$\mathcal{C}$ is usually identified with its image as a linear subspace of dimension k of the vector space $\mathbb{F}_q^n$.*

One of the main feature of linear codes that takes part in the error-correcting process is the minimum distance.

Definition 2.3 (Hamming distance)**.** *Let $x, y \in \mathbb{F}_q^n$, the Hamming distance $dist_H(x, y)$ between x and y is the number of positions at which the corresponding elements are different. Hence $dist_H(x, y) = |\{i : x_i \neq y_i, i \in [n]\}|$*

Definition 2.4 (Weight of a codeword)**.** *The weight $wt(x)$ of a codeword x is the Hamming distance of x from 0: $wt(x) = dist_H(x, 0)$*

Definition 2.5 (Minimum distance)**.** *The minimum distance $d_{\mathcal{C}}$ of a linear code $\mathcal{C}$ is the minimum distance between any two codewords.*

Lemma 2.6. *For a linear code $\mathcal{C}$, $d_{\mathcal{C}} = min_{x \in \mathbb{C}^*}(wt(x))$*

Proof. it follows from the linearity of $\mathcal{C}$. □

When the context is clear, we denote by d the minimum distance of $\mathcal{C}$ and we call the code a $[n, k, d]_q$ code. The minimum distance is closely related to the error-correcting ability of the code. In fact, a code can correct up to $t = \lfloor \frac{d-1}{2} \rfloor$ errors that occurred on the given codeword, since with less than t errors the original codeword is the unique nearest codeword from the corrupted message. We call such a code a t-error correcting code.

2.3 Generator and check matrix

As linear maps from $\mathbb{F}_q^k$ into $\mathbb{F}_q^n$, the image is a linear subspace of dimension k of the vector space $\mathbb{F}_q^n$ and is generated by a basis of k codewords.

Definition 2.7 (Generator matrix)**.** *A generator matrix for a $[n, k]_q$ linear code $\mathcal{C}$ is a matrix $G \in \mathbb{F}_q^{k \times n}$ whose rows form a basis for $\mathcal{C}$.*

We can denote by $< G >$ the code generated by G as being the vector space generated by the row vectors of G. Similarly, we can associate to $\mathcal{C}$ a linear map from $\mathbb{F}_q^n$ to $\mathbb{F}_q^{n-k}$ whose kernel is span(G).

Definition 2.8 (Check matrix)**.** *The check matrix $H \in \mathbb{F}_q^{n \times n-k}$ is the matrix representation of the linear map from $\mathbb{F}_q^n$ to $\mathbb{F}_q^{n-k}$ whose kernel is span(G).*

Thus, any vector w written as sG for some $s \in \mathbb{F}_q^k$ is a codeword of the code $\mathcal{C}$ generated by G. A code generated by the check matrix H is called the dual code $\mathcal{C}^\perp$ of $\mathcal{C}$.

Theorem 2.9. *Let $\mathcal{C}$ be a linear code and let H be a check matrix for $\mathcal{C}$. Then*

- *$\mathcal{C}$ has minimum distance $\geq d$ if and only if any $d-1$ columns of H are linearly independent*
- *$\mathcal{C}$ has minimum distance $\leq d$ if and only if H has d columns that are linearly dependent.*

Proof. The proof can be found in [6], chapter 4. □

2.4 Decoding techniques for linear codes

Suppose that codewords from a code $\mathcal{C}$ are being sent over a communication channel.

Definition 2.10 (Nearest neighbor decoding)**.** *Nearest neighbor decoding, or minimum distance decoding decodes a received word x into the codeword c_x such that c_x is minimal among all the codewords in $\mathcal{C}$, that is*

$$d(x, c_x) = min_{c \in \mathcal{C}} d(x, c_x). \tag{2.1}$$

Of course, depending on the code and the received word, uniqueness might not be guaranteed. We can distinguish between complete and incomplete decoding for the nearest neighbor decoding rule. For a given received word c_x, if two or more codewords satisfy (2.1), then the complete decoding rule arbitrarily selects one of them to be the most likely word sent, while the incomplete decoding rule requests for retransmission.

Definition 2.11 (t-error-correcting code)**.** *Let t be a positive integer. A code $\mathcal{C}$ is t-error-correcting if the nearest neighbor decoding method can correct up to t errors, assuming that the incomplete decoding rule is used.*

Theorem 2.12. *A code $\mathcal{C}$ is t-error-correcting if and only if $d_{\mathcal{C}} \geq 2t + 1$.*

Proof. The proof can be found in [6] □

Definition 2.13 (Coset)**.** *Let $\mathcal{C}$ be a $[n, k]_q$ linear code and let $u \in \mathbb{F}_q^n$ be any vector. We define the coset of $\mathcal{C}$ determined by u to be the set*

$$\mathcal{C} + u = \{v + u : v \in \mathcal{C} = \mathcal{C} + u\}$$

Cosets play a crucial role in the decoding schemes discussed next.

Example 2.14. *Let $q = 2$ and $n = 4$. The cosets of the binary linear code $\mathcal{C} = \{0000, 1011, 0101, 1110\}$ of dimension $k = 2$ are as follows : The size of*

$$\begin{array}{llllll} 0000+\mathcal{C}: & 0000 & 1011 & 0101 & 1110 \\ 0001+\mathcal{C}: & 0001 & 1010 & 0100 & 1111 \\ 0010+\mathcal{C}: & 0010 & 1001 & 0111 & 1100 \\ 1000+\mathcal{C}: & 1000 & 0011 & 1101 & 0110 \end{array}$$

this array is $2^k \times 2^{n-k}$.

Definition 2.15 (Coset leader)**.** *A word of least Hamming weight in a coset is called a coset leader*

Nearest neighbor decoding works for a linear code $\mathcal{C}$ in the following way. Upon receiving the word x, we choose a word e of least weight in the coset $x + \mathcal{C}$ and conclude that $c_x = x - e$ was the codeword transmitted, assuming that there exists a unique codeword c_x at minimum distance less than $t = \lfloor \frac{d-1}{2} \rfloor$ from x. In the example above, if the received word is 1101, then we can see that it

belongs to coset $1000 + \mathcal{C}$ where 1000 is the coset leader. Hence, the original codeword was $1101 - 1000 = 0101$.
Next we will show that some time can be saved in the decoding process by making use of the notion of syndrome to identify the coset to which the received word belongs.

Definition 2.16. *Let $\mathcal{C}$ be a $[n,k]_q$ linear code and let H be a check matrix for $\mathcal{C}$. For any $w \in \mathbb{F}_q^n$, the syndrome of w is the word $S(w) = wH^T \in \mathbb{F}_q^{n-k}$.*

It directly follow that S is a linear map and $S(u) = 0$ if and only if $u \in \mathcal{C}$

Definition 2.17 (syndrome look-up table)**.** *A table that matches each coset leader with its syndrome is called a syndrome look-up table (or standard decoding array).*

A quick way to construct a syndrome look-up table, given the check matrix H and distance d for the code $\mathcal{C}$, is to generate all the error patterns e with $wt(e) \leq \lfloor \frac{d-1}{2} \rfloor$ as coset leaders and compute the syndrome $S(e)$ for each of them.

Example 2.18. *Using the same code example as before, we obtain the following syndrome look-up table The size of this array is $2^k \times 2$ with $k = 2$.*

Coset leader u	Syndrome $S(u)$
0000	00
0001	01
0010	10
1000	11

Now, for any received word x, one just has to compute the syndrome $S(x)$ and look to the corresponding coset leader u_x. If less than $\lfloor \frac{d-1}{2} \rfloor$ errors occured then the original codeword is $c_x = x - u_x$.
This method is faster since now the size of the array is $q^k \times 2$ compared to the array from 2.14 of size $q^k \times q^{n-k}$.

Two important mathematical problems of coding theory naturally derives from those two decoding techniques :

Definition 2.19 (General Decoding Problem for linear codes (GDP))**.** *Let $\mathcal{C}$ be an $[n,k]_q$ linear code over $\mathbb{F}_q$ and $y \in \mathbb{F}_q^n$. The GPD consists in finding $x \in \mathcal{C}$ such that $dist(y,x)$ is minimal.*

Definition 2.20 (Syndrome Decoding Problem (SDP))**.** *Given a matrix $H \in \mathbb{F}_q^{n \times n-k}$, a vector $s \in \mathbb{F}_q^n$, and a non negative integer t_0, find a vector $u_x \in \mathbb{F}_q^{n-k}$ with Hamming weight $wt(u_x) = t_0$ such that $Hu_x = s$*

The GDP and SDP can be solved using a syndrome look-up table as in 2.18. Given a t_0-error correcting code and a vector x we compute $S(x)$. Finding u_x, we can retrieve the codeword $c_x = x - u_x$ at a minimal distance to x and the equality $Hu_x = S(x)$ is satisfied.

2.5 The Singleton bound

Finally, let us present the Singleton bound which can describe in some sense how good an error-correcting code can be.

Lemma 2.21 (Singleton bound)**.** *Let $A_q(n,d)$ denote the maximum number of possible codewords in a q-ary code of length n. Then $A_q(n,d) \leq q^{n-d+1}$. Moreover, if the code is linear of dimension k, then $d \leq n-k+1$.*

Proof. Since the minimum distance is d, we can delete the first $d-1$ elements of every codeword and the resulting punctured codewords must still be pairwise different. The code being q-ary, the maximum number of such resulting codewords of length $n-(d-1)$ is q^{n-d+1}. Since the code is arbitrary and the number of punctured codewords is the same as the number of codewords, we obtain $A_q(n,d) \leq q^{n-d+1}$. If the code is linear of dimension k, the number of codewords is q^k so $k \leq n-d+1$ and finally $d \leq n-k+1$ □

Definition 2.22 (Maximum distance separable code (MDS))**.** *A maximum distance separable code is a linear $[n,k]_q$ code which reaches equality for the Singleton bound, that is $d = n-k+1$.*

In fact, MDS codes are interesting in the context of error correction since they reach the equality for the Singleton bound so they optimise the trade-off between information rate $R = \frac{k}{n}$ and the minimal distance d. A famous example of MDS code is the generalised Reed-Solomon code which we will study in section 3.

3 McEliece cryptosystem

In this section we will present the McEliece cryptosystem and the mathematical objects it relies on. As discussed in the Introduction 1, the McEliece cryptosystem is a code-based cryptosystem. Let us first give some notions of computational complexity related to code-based cryptography, this will help us understand what makes the McEliece cryptosystem robust.

3.1 Complexity of some code-based problems

Definition 3.1 (The complexity class P)**.** *The class P is the set of decision problems that can be decided by a deterministic Turing machine in a polynomial number of steps in the size of the input.*

Definition 3.2 (The complexity class NP)**.** *The class NP is the set of decision problems that can be verified by a deterministic Turing machine in a polynomial number of steps in the size of the input.*

Definition 3.3 (Polynomial time computable function)**.** *A function f is polynomial time computable if there is a Turing Maching that runs in polynomial time such that on every input x outputs $f(x)$.*

Definition 3.4 (Polynomial-time reduction)**.** *A decision problem $\mathcal{A}$ is said to be polynomial time reducible to a decision problem $\mathcal{B}$ if there exists a polynomial time computable function f such that for every instance x of $\mathcal{A}$, $\mathcal{A}(x)$ is true if and only if $\mathcal{B}(f(x))$ is true. We denote by $\mathcal{A} \leq_P \mathcal{B}$*

Intuitively, this means that if we know how to solve $\mathcal{B}$, we can efficiently transform each instance x of $\mathcal{A}$ to an instance of $\mathcal{B}$ and verify if it is true to know if $\mathcal{A}(x)$ is true.

Definition 3.5 (NP-complete problem)**.** *A NP-complete problem is a problem $\mathcal{A} \in NP$ such that for every problem $\mathcal{B} \in NP$ one has $\mathcal{B} \leq_P \mathcal{A}$.*

Intuitively, a NP-complete problem is a decision problem that belong to NP and is at least as hard as all the problems in this class.

Theorem 3.6. *The GDP and SDP from 2.19 and 2.20 are NP-complete.*

Proof. We can show this by reducing those problems to the NP-complete three-dimensional matching problem [1] □

The GDP and SDP are not properly formulated as decision problems but essentially they are decision problems since here deciding if there exists a true instance is equivalent to finding it. Based on the hardness of those decoding problems, some code-based cryptosystems have been invented such as the McEliece cryptosystem.

3.2 Security level

In the context of secure communications, it is important to define a mathematical measure to express the security of a given cryptosystem. This will also be a convenient way to compare different cryptosystems and is useful when combining multiple primitives in a hybrid cryptosystem.
We can simply define the security of a cryptosystem by the number of operations that an attacker should perform to decipher a message. The number of operations to perform is a measure of security because it is proportional to the time the attacker should spend on the attack. If the message cannot be deciphered for a period of time as long as the two parties desire to make effective the message information, then the two parties consider the cryptosystem to be secure. In the realm of classical computation, operations are called bits and the duration could directly be calculated in terms of the number of bits per second that a classical computer could achieve and the number of operations needed to decipher a message.

As far as public key cryptosystems have evolved, there is naturally a trade-off to make between the size of the key and the security level desired. Of course, we might never know with certainty the true security level of a cryptosystem since someone could always come up with an innovative attack. Intrinsically, that should be the risk to take when sending information. But we can still make some heuristics on the security level of a cryptosystem based on the best-known attacks and the theoretical assumptions about computational complexity. The number of operations of such attacks gives the estimated security level such that if 2^b operations are needed, then the cryptosystem has b-bit security.
Currently, the fastest classical supercomputers can perform approximately 10^{17} bits per second. Hence, if a cryptosystem has 80-bit security, then the time needed to decipher the message with a supercomputer would be

$$\frac{2^{80}}{10^{17}} \approx 2^{29} sec \approx 17 years$$

For that reason, 80-bit security can be considered as a limit between practical and impractical attacks on a cryptosystem. As mentioned before, the security also depends on the key size that the users accept to store in memory. For example, for the RSA cryptosystem at a 128-bit security level, NIST and ENISA recommend using 3072-bit keys. Elliptic curve cryptography meanwhile requires shorter keys, so the recommendations are around 300-bit keys for the same security level. In comparison, to obtain a 128-bit security the Goppa codes must have length 2960 and dimension 2288 (with a Goppa polynomial of degree 56). Hence the public size is $2960 \times 2288 \approx 67 Mb$.

3.3 McEliece cryptosystem

The McEliece cryptosystem is a public key cryptosystem based on the difficulty of decoding linear codes. It was introduced in 1978 by Robert McEliece and it is the first which uses error-correcting codes as a trapdoor for a cryptosystem, the trapdoor being the knowledge of an efficient error-correcting algorithm.
See [8]
The McEliece cryptosystem works as follows if user Bob wants to secretly send messages to user Alice :

- **system parameters :** $n, k, t \in \mathbb{N}$ with $t \ll n$ and $k \leq n$. In practical implementations, $n \approx 2k$ and $k \approx 10 * t$.
- **key generation :** Alice generates the following matrices:
  G : $k \times n$ generator matrix of a $[n, k]_q$ code $\mathcal{G}$ with minimum distance $d \geq 2t + 1$
  S: $k \times k$ random binary non-singular matrix (the "scrambler" matrix)
  P: $n \times n$ random permutation matrix
  Then, compute the $k \times n$ matrix $G' = SGP$
- **public key:** Alice publicly communicates (G', t)
- **private key:** Alice secretly keeps the knowledge of $(S, \mathcal{D}_\mathcal{G}, P)$, where $\mathcal{D}_\mathcal{G}$ is an efficient (polynomial time) decoding algorithm for $\mathcal{G}$.
- **encryption:** Alice encrypts a message $m \in \{0,1\}^k \equiv \mathbb{F}_2^k$, randomly choose a vector $e \in \{0,1\}^n$ of weight t (the error vector) and compute the ciphertext c as follows :

  $$c = mG' \oplus e$$

- **decryption:** to decrypt the ciphertext c, Alice calculates :

  $$cP^{-1} = (mS)G \oplus eP^{-1}$$

  and applies the decoding algorithm $\mathcal{D}_\mathcal{G}$ for $\mathcal{G}$ to it. Noticing that cP^{-1} has a Hamming distance of t to $\mathcal{G}$ we obtain the codeword

  $$mSG = \mathcal{D}_\mathcal{G}(cP^{-1})$$

  Then Alice finds a set $J \subseteq \{1, \ldots, n\}$ of size k such that $G_{.J}$ is invertible, where $G_{.J}$ denotes the matrix G restrained to the columns indexed by the set J. Such a set always exists since G is of rank k and it can be found in polynomial time by solving a set of linear equations.
  Finally Alice can compute the original message $m = (mSG)_J(G_{.J})^{-1}S^{-1}$

Looking at the protocol, we can already attempt to grasp what makes it secure. G' is in fact the generator matrix of a linear code, but it has no obvious structure due to the scrambler matrix S so decoding from the noise e would

be only possible by going through an efficient general decoding technique, that is solving the GDP in polynomial time. This is unlikely since this problem is NP-complete and it would prove that $P = NP$, which is unlikely. Hence, unless some alternative private keys are found, the only way to decipher c seems to be the decryption method explained above which requires the knowledge of S, P and the Goppa polynomial.

3.4 A first discussion on security aspects

Definition 3.7 (Kerkhoff's principle)**.** *Kerkhoff's principle states that what makes the security of a cryptosystem is the ingenuity of its design and not its secrecy.*

This principle is widely embraced by cryptographers since cryptographic primitives are long and tedious to come up with, to share, and to implement. In our largely interconnected world, the assumption that the cryptosystem of a communication system is known is relevant.

Definition 3.8 (McEliece problem (MP))**.** *Given a McEliece public key* (G', t) *where* $G' = SGP \in \mathbb{F}_2^{k\times n}, S \in \mathbb{F}_2^{k\times k}, P \in \mathbb{F}_2^{n\times n}$ *and a cyphertext* $c \in \{0,1\}^n$*, find the message* $m \in \{0,1\}^k$ *such that dist*$(mG', c) = t$

Clearly, someone who can solve the GDP will come up with the codeword mG', then he can easily recover m by inverting G'_I for some $I \in \{1, \dots, n\}$ of size k. It follows that GDP is at least as hard as MP, the converse is not known to be true since $\mathcal{G}' =< G' >$ is not a random one but comes from $G' = SGP$ where G is the generator matrix of a known class of code, for instance Goppa codes.
An important aspect is scrambling the rows of GP by a random matrix S, which is a completely random binary matrix but it is a $k \times k$ matrix and not $n \times n$ like P. So depending on the rate $R = \frac{k}{n}$ of the code, we obtain a generator matrix $G = SG'P$ more or less random. Of course in the extreme case, if $R = 1$, then the McEliece problem is as hard as the GDP. But for instance the original proposal of McEliece using Goppa code was $n = 1024, k = 524$ so $R \approx 0.51$. In fact for applications, R should be not too close to 0 or 1 to keep the key sizes reasonable compared to the size of the message while transmitting some information. Up to now, it is unknown whether MP is equivalent to the GDP but one hopes that S makes things random enough.
We also mention that the permutation matrix P should remain secret. Indeed, when Goppa codes are used, if H' denotes the dual matrix of $G' = SGP$ then we can use that $S_c(x) \equiv 0 \bmod g(x)$ from Definition 3.17 for every vector c such that $HP^{-1} = 0$ and retrieve the Goppa polynomial $g(x)$.

In its original version, McEliece used Goppa codes which makes the cryptosystem particularly robust and still unbroken. In the following, we will explain the construction of some important families of linear codes to eventually better understand how Goppa codes are constructed and what makes them "special".

3.5 Generalized Reed-Solomon codes

An important family of codes is the generalized Reed-Solomon codes (GRS). Historically, it was introduced in the 60s to code information on a vast variety of technologies, for instance on CDs and DVDs to maintain robustness again potential damages and for data transmission technologies such as DSL and satellite communication where the extreme environment could induce a lot of errors.

Since computers manipulate bits, in the following constructions of those codes we work in fields of characteristic 2 and let $n \leq q$ with typically $q = 2^m$ for some $m \geq 1$. We mostly follow the general introduction proposed in chapter 9 of [6]. Let us define some parameters $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_{2^m}^n$ and $v = (v_1, \ldots, v_n) \in \mathbb{F}_{2^m}^n$ where $v_i \neq 0$ and the $\alpha_i, i = 1, \ldots, n$ are distinct.

Definition 3.9 ($GRS_k(\alpha, v)$ code)**.** *Let $k \leq n$, $GRS_k(\alpha, v)$ is the linear code generated by the generator matrix*

$$G_{\alpha,v} = \begin{bmatrix} v_1 & \ldots & v_n \\ v_1\alpha_1 & \ldots & v_n\alpha_n \\ \vdots & & \vdots \\ v_1\alpha_1^{k-1} & \ldots & v_n\alpha_n^{k-1} \end{bmatrix} = \begin{bmatrix} 1 & \ldots & 1 \\ \alpha_1 & \ldots & \alpha_n \\ \vdots & & \vdots \\ \alpha_1^{k-1} & \ldots & \alpha_n^{k-1} \end{bmatrix} \begin{bmatrix} v_1 & & \\ & \ddots & \\ & & v_n \end{bmatrix}$$

From the definition, it is easy to see that $GRS_k(\alpha, v)$ is the set of codewords $\{(v_1 f(\alpha_1), \ldots, v_n f(\alpha_n)) : f(x) \in \mathbb{F}_q[X] \text{ and } \deg(f(x)) < k\}$. Hence, given a codeword $c \in \mathbb{F}_q^n$, we can associate a unique polynomial $f_c(x) \in \mathbb{F}_q[X]$ of degree less than k.

Theorem 3.10. *The dual of $GRS_k(\alpha, v)$ is $GRS_{n-k}(\alpha, v')$ for some $v' \in \mathbb{F}_q$.*

Proof. First, if $k = n - 1$, $GRS_k(\alpha, v)$ is a $[n, 1]$ code so it is generated by a vector $v' = (v'_1, \ldots, v'_n)$, where $v_i \in \mathbb{F}_q^*$. This dual code is exactly $GRS_1(\alpha, v')$. In particular, for all $f(x) \in \mathbb{F}_q[X]$ of degree less than $n - 1$ we have :

$$v_1 v'_1 f(\alpha_1) + \cdots + v_n v'_n f(\alpha_n) = 0 \tag{3.1}$$

Now let us prove that $GRS_k(\alpha, v)^\perp = GRS_{n-k}(\alpha, v')$ for arbitrary k. Be definition, a typical codeword in $GRS_k(\alpha, v)$ is $(v_1 f(\alpha_1), \ldots, v_n f(\alpha_n))$ with $f(x) \in \mathbb{F}_q[X]$ and $\deg(f(x)) \leq k - 1$. Similarly a typical codeword in $GRS_{n-k}(\alpha, v')$ is $(v'_1 g(\alpha_1), \ldots, v'_n g(\alpha_n))$ with $g(x) \in \mathbb{F}_q[X]$ and $\deg(f(x)) \leq n - k - 1$. Since $deg(f(x)g(x)) < n - 1$, using 3.1 we have :

$$(v_1 f(\alpha_1), \ldots, v_n f(\alpha_n)) \cdot (v'_1 g(\alpha_1), \ldots, v'_n g(\alpha_n)) = 0$$

Therefore $GRS_{n-k}(\alpha, v') \subseteq GRS_k(\alpha, v)^\perp$. Comparing the dimension of both codes, we obtain equality. □

Lemma 3.11. *(corollary) a check matrix of $GRS_k(\alpha, v)$ has the form :*

$$H = \begin{bmatrix} 1 & \dots & 1 \\ \alpha_1 & \dots & \alpha_n \\ \vdots & & \vdots \\ \alpha_1^{n-k-1} & \dots & \alpha_n^{n-k-1} \end{bmatrix} \begin{bmatrix} v'_1 & & \\ & \ddots & \\ & & v'_n \end{bmatrix}$$

Lemma 3.12. *$GRS_k(\alpha, v)$ has parameters $[n, k, d-k+1]$, so it is an MDS code.*

Proof. The proof can be found in [6], chapter 9. □

Remark 3.13. *The Berlekamp–Welch algorithm corrects up to $\lfloor \frac{n-k}{2} \rfloor$ errors of a GRS code in polynomial time.*

3.6 Alternant codes

An interesting family of codes arising from the generalized RS codes of the previous section is the class of alternant codes.

Definition 3.14 (Alternant code)**.** *An alternant code $\mathcal{A}_k(\alpha, v)$ over the finite field $\mathbb{F}_2$ is the subfield subcode $GRS_k(\alpha, v)|_{\mathbb{F}_2^n}$, where $GRS_k(\alpha, v)$ is a GRS code over $\mathbb{F}_{2^m}$.*

Theorem 3.15. *$\mathcal{A}_k(\alpha, v)$ has parameters $[n, k, d]$ where $mk-(m-1)n \leq k' \leq k$ and $d \geq n-k+1$.*

Proof. $k' \leq k$ directly follows from $\mathcal{A}_k(\alpha, v)$ being a subfield subcode of $GRS_k(\alpha, v)$. The rest of the proof can be found in [6], chapter 9. □

Considering the check matrix H of $GRS_k(\alpha, v)$, we deduce that $\mathcal{A}_k(\alpha, v)$ is exactly the code $\{c \in \mathbb{F}_2 : cH^\perp = 0\}$. Since H is determined by α and v' from 3.11, we can also use the notation $\mathcal{A}_k(\alpha, v')$ for the alternant code.

3.7 Goppa codes

Goppa codes were defined by V. D. Goppa in 1970. The definition we give here derives from the McElice summary paper [4]
Let m and t be positive integers and let

$$g(X) = \sum_{i=0}^{t} g_i X^i \in \mathbb{F}_{2^m}[X]$$

be a monic polynomial of degree t called the Goppa polynomial. Let $\alpha = (\alpha_1, \dots, \alpha_{n-1}) \in \mathbb{F}_{2^m}^n$ a tuple of n distinct elements called the locators such that $g(\alpha_i) \neq 0$ for all $0 \leq i \leq n-1$.

Definition 3.16 (syndrome)**.** *For any $c = (c_1, \dots, c_n) \in \mathbb{F}_2^n$, the syndrome of c is $S_c(X) = \sum_{i=1}^{n} \frac{c_i}{X-\alpha_i}$.*

Definition 3.17 (binary Goppa Codes)**.** *The binary Goppa code* $\mathcal{G}(\alpha, g(X))$ *over* $\mathbb{F}_2$ *is the set of all* $c = (c_1, \ldots, c_n) \in \mathbb{F}_2^n$ *such that* $S_c(X) \equiv 0$ *mod* $g(X)$ *holds in* $\mathbb{F}_{2^m}[X]$

From the definition, it is clear that $\mathcal{G}(\alpha, g(X))$ is a linear code. If $g(X)$ is irreducible over $\mathbb{F}_{2^m}$, then $\mathcal{G}(\alpha, g(X))$ is called an irreducible binary Goppa code.

Lemma 3.18. *The minimum distance of an irreducible binary Goppa code is at least* $2t + 1$*.*

Proof. The proof can be found in [4] □

It follows that an irreducible Goppa code is a $\lfloor \frac{(2t+1)-1}{2} \rfloor$-error correcting code.

Theorem 3.19. *There exists a polynomial time error correction algorithm for an irreducible binary Goppa code that needs* $\mathcal{O}(n \cdot t \cdot m^2)$ *binary operations.*

Proof. The algorithm is presented in subsection 3.9 □

Remark 3.20. *This algorithm is known as the Patterson algorithm and uses the knowledge of the irreducible Goppa polynomial in the decoding process.*

As explained in [4], the check matrix of $\mathcal{G}(\alpha, g(X))$ can be written as :

$$H = \begin{bmatrix} g_t g(\alpha_1)^{-1} & \ldots & g_t g(\alpha_n)^{-1} \\ (g_{t-1} + g_t\alpha_1)g(\alpha_0)^{-1} & \ldots & (g_{t-1} + g_t\alpha_n)g(\alpha_{n-1})^{-1} \\ \vdots & & \vdots \\ (\sum_{j=1}^{t} g_j \alpha_1^{j-1})g(\alpha_1)^{-1} & \ldots & (\sum_{j=1}^{t} g_j \alpha_n^{j-1})g(\alpha_n)^{-1} \end{bmatrix} = XYZ$$

Where

$$X = \begin{bmatrix} g_t & \ldots & 0 \\ \vdots & \ddots & \vdots \\ g_1 & \ldots & g_t \end{bmatrix}; Y = \begin{bmatrix} 1 & \ldots & 1 \\ \alpha_1 & \ldots & \alpha_n \\ \vdots & & \vdots \\ \alpha_1^{t-1} & \ldots & \alpha_n^{t-1} \end{bmatrix}; Z = \begin{bmatrix} g(\alpha_1)^{-1} & & \\ & \ddots & \\ & & g(\alpha_n)^{-1} \end{bmatrix}$$

Since X is invertible, the following matrix is also a check matrix of $\mathcal{G}(\alpha, g(X))$:

$$X^{-1}H = \begin{bmatrix} 1 & \ldots & 1 \\ \alpha_1 & \ldots & \alpha_n \\ \vdots & & \vdots \\ \alpha_1^{t-1} & \ldots & \alpha_n^{t-1} \end{bmatrix} \begin{bmatrix} g(\alpha_1)^{-1} & & \\ & \ddots & \\ & & g(\alpha_n)^{-1} \end{bmatrix} \tag{3.2}$$

We see here that this check matrix has the same form as the check matrix of a $GRS_{n-t}(\alpha, v)$. Since the codewords are restricted to $\mathbb{F}_2^n$, it is tempting to conclude that $\mathcal{G}(\alpha, g(X))$ is none other than $\mathcal{A}_{n-t}(\alpha, v')$ for $v' = (g(\alpha_1)^{-1}, \ldots, g(\alpha_n)^{-1})$. The following theorem confirms this intuition and gives a complete description of the Goppa code as a subfield subcode of a GRS code.

Theorem 3.21. *The Goppa code $\mathcal{G}(\alpha, g(X))$ is $GRS_{n-t}(\alpha, v)|_{\mathbb{F}_2^n}$ where $v = (v_1, \ldots, v_n)$ with $v_i = g(\alpha_i)/(\prod_{j \neq i}(\alpha_i - \alpha_j))$ for $i \in \{1, \ldots, n\}$.*

Proof. From 3.2, we see that the check matrix of $\mathcal{G}(\alpha, g(X))$ is a generator matrix of $GRS_t(\alpha, v')$ defined in 3.9, hence we have that $\mathcal{G}(\alpha, g(X)) = GRS_t(\alpha, v')^{\perp}|_{\mathbb{F}_2^n}$ for $v' = (g(\alpha_1)^{-1}, \ldots, g(\alpha_n)^{-1})$. Moreover $GRS_t(\alpha, v')^{\perp} = GRS_{n-t}(\alpha, v'')|_{\mathbb{F}_2^n}$ for some $v'' \in \mathbb{F}_{2^m}^n$ by Theorem 3.10. So it remains to show that $v'' = v$ by showing that $GRS_t(\alpha, v')^{\perp} = GRS_{n-t}(\alpha, v)$, that is

$$v_1 g(\alpha_1)^{-1} f(\alpha_1) + \cdots + v_n g(\alpha_n)^{-1} f(\alpha_n) = 0$$

where $v = (v_1, \ldots, v_n)$ with $v_i = g(\alpha_i)/(\prod_{j \neq i}(\alpha_i - \alpha_j))$ for $i \in \{1, \ldots, n\}$ and for all polynomial $f(x) \in \mathbb{F}_{2^m}[x]$ with $deg(f) \leq n - 2$.
Since $deg(f) \leq n - 2$, it is determined by its values at $n - 1$ points and we can show that it satisfies the interpolation formula

$$f(x) = \sum_{i=1}^{n} f(\alpha_i) \prod_{j \neq i} \frac{x - \alpha_j}{\alpha_i - \alpha_j}$$

Since the coefficient of x^{n-1} is 0, we obtain

$$0 = \sum_{i=1}^{n} \frac{f(\alpha_i)}{\prod_{j \neq i}(\alpha_i - \alpha_j)} = v_1 g(\alpha_1)^{-1} f(\alpha_1) + \cdots + v_n g(\alpha_n)^{-1} f(\alpha_n)$$

□

Theorem 3.22. *A binary Goppa code $\mathcal{G}(\alpha, g(X))$ is a $[n, k, d]_2$ linear code with $k \geq n - mt$.*

Proof. The proof follows from theorem 3.15 and theorem 3.21. □

Goppa codes were the class of codes that McEliece chose for his cryptosystem because they are easy to generate but hard to find. Indeed, any irreducible polynomial over a finite field $\mathbb{F}_2^m$ can be used to create a Goppa code, but the generator matrix is nearly unstructured since a Goppa code is an alternant code. GRS codes are well structured by defining explicitly a generator matrix as a product of a Vandermonde matrix and a diagonal matrix. This makes them weaker again structural attacks as in 4.1. However, alternant codes are subcodes of GRS codes restricted to the subfield $\mathbb{F}_2$, a restriction that is a priori unrelated to the very definition of GRS codes. There is still some structure and Goppa codes are efficient error-correcting codes, but applying the matrices S and P makes the structure hidden in a more subtle way than it happens for GRS codes and this is what makes them relevant for McEliece cryptosystem. Over time, Goppa codes became one of the rare unbroken classes of codes for McEliece cryptosystem.

3.8 Algebraic Geometry codes

As we saw, Goppa codes are closely related to GRS codes as being their subfield subcodes over $\mathbb{F}_2$. In fact, we can better understand how this relation is made if we take a broader look using algebraic geometry to find a possible generalization. This is motivated by the following observation:

Let $n = q-1$ and $\alpha = (\beta, \beta^2, \ldots, \beta^n)$ where $\beta \in \mathbb{F}_q^\times$ is a primitive element. We notice that if we denote $\mathcal{L}_k = \{f \in \mathbb{F}_q[X] | \deg f \leq k-1\}$ and the evaluation map given by :

$$ev(f) = (f(\beta), f(\beta^2), \ldots, f(\beta^n)) \in \mathbb{F}_q^n$$

Then it directly follows that the image of this evaluation map is $GRS(\alpha, (1, \ldots, 1))$. In fact, V.D.Goppa observed that $\mathbb{F}_q$ can be considered as an affine line, with corresponding projective line $\mathbb{P}^1_{\mathbb{F}_q}$. Then, the polynomials in $\mathbb{F}_q[x]_{<k}$ can be thought as polynomials with a pole of multiplicity no more than k at the point at infinity in $\mathbb{P}^1_{\mathbb{F}_q}$. With this in mind, Goppa looked toward the Riemann-Roch space defined below. The elements of a Riemann-Roch space are exactly those functions with pole multiplicities restricted below a given threshold, with the restriction being prescribed in the coefficients of a corresponding divisor. Evaluating those functions at the rational points on an algebraic curve $\mathcal{X}$ over $\mathbb{F}_q$ gives a code in the same sense as the Reed-Solomon construction for the affine projective line.

In the following, we give a brief overview of the general concepts behind algebraic geometry codes based to illustrate this different and more general understanding of Goppa codes related to algebraic geometry. We do not provide any proof but the reader can refer to the book "Algebraic Function Fields and Codes" by H.Stichtenoth [12].

Definition 3.23 (Projective plane curve Γ)**.** *Let $f(X,Y,Z) \in \mathbb{C}[X,Y,Z]$ be a homogenous irreducible polynomial. A projective plane curve Γ is defined by*

$$\Gamma = \{(X,Y,Z) \in \mathbb{P}^2(\mathbb{C}) | f(X,Y,Z) = 0\}$$

Definition 3.24 (Function field $K(\Gamma)$ of a projective plane curve)**.** *Let $f(X,Y,Z) \in \mathbb{C}[X,Y,Z]$ be a homogenous irreducible polynomial. The function field $K(\Gamma)$ of a projective plane curve is defined by*

$$K(\Gamma) = Quot f(R(\Gamma)) = \left\{ \frac{g(X,Y,Z)}{h(X,Y,Z)} \text{ such that } g, h \text{ have same homogenous degree } d \right\}$$

Definition 3.25 (Valuation ring $\mathcal{O}_P(\Gamma)$ of a point $P \in \Gamma$)**.** *A valuation ring of a point $P \in \Gamma$ is defined by*

$$\mathcal{O}_P(\Gamma) = \{\phi \in K(\Gamma) | \phi \text{ is defined at } P\}$$

It is shown in [12] that there is a correspondence between the points of Γ and the valuation rings of its function field $K(\Gamma)$. Now let us define a more general notion of function field from which $K(\Gamma)$ falls into

Definition 3.26 (Algebraic function field F/K of one variable)**.** *An algebraic function field of one variable F/K is a field extension F of a field K such that F is an algebraic extension of $K(x)$ where $x \in F$ is transcendental over K.*

If F/K is called rational if $F = K(x)$ for some $x \in F$ which is transcendental over K.

Definition 3.27 (Valuation ring of F/K)**.** *A Valuation ring of F/K is a ring $\mathcal{O}$ with the following properties*

- $K \subsetneq \mathcal{O} \subsetneq F$
- *for any $f \in F$ we have that $f \in \mathcal{O}$ or $f^{-1} \in \mathcal{O}$*

Definition 3.28 (Place P of a function field F/K)**.** *A place P of a function field F/K is the unique maximal ideal of a valuation ring $\mathcal{O}$ of F/K. We define*

$$\mathbb{P}_F = \{P|\ P \text{ is a place of } F/K\}$$

Definition 3.29 (Valuation v_P)**.** *Let P be a place in $\mathbb{P}_F$ and let $t \in \mathcal{O}$ such that $t \cdot \mathcal{O} = P$. We define $v_P : F \longrightarrow \mathbb{Z} \cup \{\infty\}$ by*

$$v_P(f) = \begin{cases} n & \text{if } f \geq 0, f = t^n u,\ u \text{ a unit} \\ \infty & \text{if } f = 0 \end{cases}$$

Definition 3.30 (Divisor D)**.** *A divisor D is a formal sum $\sum_{P \in \mathbb{P}_F} n_P P$ with $n_P \in \mathcal{Z}$ where almost all $n_P = 0$.*

We can add divisors as $D + D' = \sum_{P \in \mathbb{P}_F} (n_P + n'_P)P$ and define a partial order $D \leq D'$ if and only if $v_P(D) \leq v_P(D')$ for all $P \in \mathbb{P}_F$. Furthermore, we call the set $\{P \in \mathbb{P}_F : n_P \neq 0\}$ the support $supp(D)$ of D.

Definition 3.31 (Principal divisor)**.** *Let $f \in F \setminus \{0\}$, the principal divisor (f) of f is*

$$(f) = \sum_{P \in \mathbb{P}_F} v_P(f)P$$

Definition 3.32 (Riemann-Roch space $\mathcal{L}(D)$)**.** *Let D be a divisor, the Riemann-Roch space of D is*

$$\mathcal{L}(D) = \{f \in F | (f) \geq -D\} \cup \{0\}$$

Lemma 3.33. *$\mathcal{L}(D)$ is a vector space over K*

Proof. The proof can be found in [12]. □

Now let us introduce the algebraic geometry codes (AG codes). Let $P_1, \ldots, P_n$ be pairwise distinct places of a function field $\mathrm{F}/\mathbb{F}_q$ and $D = P_1 + \cdots + P_n$. Furthermore, let G be a divisor of $F/\mathbb{F}_q$ such that $suppG \cap suppD = \emptyset$.

Definition 3.34. *The algebraic geometry code $\mathcal{C}_{\mathcal{L}}(D,G)$ is defined as*

$$\mathcal{C}_{\mathcal{L}}(D,G) = \{(f(P_1),\ldots,f(P_n))|f \in \mathcal{L}(G) \subseteq \mathbb{F}_q^n\}.$$

With those definitions at hand, it is possible to show that every generalized Reed-Solomon code $GRS(\alpha, v)$ over $\mathbb{F}_q$ can be represented as an AG code over a rational function field. We do not prove the results here but we refer the reader to proposition 2.3.5 in [12]. More precisely if we let:

- $F = \mathbb{F}_q[x]$
- P_i the zero of $x - \alpha_i$ $(i = 1,\ldots,n)$ and P_∞ the pole of x
- $u \in F$ such that $u(P_i) = v_i, (i = 1,\ldots,n)$
- $D = P_1 + \cdots + P_n$ and $G = (k-1)P_\infty - (u)$

Then

$$GRS(\alpha, v) = \mathcal{C}_{\mathcal{L}}(D,G)$$

Now let $m \in \mathbb{N}^*$. Similarly, every Goppa code $\mathcal{G}(\alpha, g(X))$ with $\alpha \in \mathbb{F}_{q^m}^n$ and $g(x) \in \mathbb{F}_{q^m}[x]$ can be interpreted as subfield subcode of an algebraic geometry code the following way, with now an additional parameter $m \in \mathbb{N}$. If we let

- $F = \mathbb{F}_{q^m}[x]$
- P_i the zero of $x - \alpha_i$ $(i = 1,\ldots,n)$ and P_∞ the pole of x
- Let G_0 be the divisor of $g(x)$ in

Then

$$\mathcal{G}(\alpha, g(X)) = \mathcal{C}_{\mathcal{L}}(D, G_0 - P_\infty)^{\perp}|\mathbb{F}_q$$

3.9 Patterson's algorithm

The minimum distance of a binary Goppa code $\mathcal{G}(\alpha, g(X))$ which is generated by an irreducible polynomial of degree t is at least $2t+1$ by Lemma 3.18. Therefore, it is always possible to correct up to t errors, which can be done in polynomial time by the so-called Patterson's algorithm which heavily exploits the fact that the code is binary. Let assume the codeword is $y = mG \in \mathcal{G}(\alpha, g(X))$ and the error vector is $e \in \mathbb{F}_2^n$ with weigth $wt(e) \leq t$, then the received cyphertext is

$$c = y \oplus e$$

Since y is a codeword, we have $S_y(X) = \sum_{i=1}^{n} \frac{y_i}{X-\alpha_i} \equiv 0 \bmod g(X)$ so

$$S_c(X) \equiv S_e(X) \bmod g(X).$$

We can compute the syndrome $S_c(X)$ in at most $(n-k)n$ binary operations by employing the check matrix H.
Let define the *error locator polynomial*. For $I_e = \{i : e_i = 1\}$, we set

$$\sigma_e(X) = \prod_{j \in I_e} (X - \alpha_j) \in \mathbb{F}_{2^m}[X]$$

We notice that $S_e(X) \equiv \sum_{i=1}^{n} \frac{e_i}{X-\alpha_i} \equiv \sum_{i \in \mathcal{I}_e} \frac{1}{X-\alpha_i} \mod g(X)$, hence we can show by induction on $|\mathcal{I}_e|$ that

$$\sigma_e(X) S_e(X) \equiv \sigma_e'(X) \mod g(X) \tag{3.3}$$

where for a given polynomial $\sigma \in \mathbb{F}_{2^m}[X]$, $\sigma'(X) = \sum_{i=1}^{n} i\sigma_i X^{i-1}$ is its derivative.

Let us state two lemmas :

Lemma 3.35. *The map $\mathbb{F}_{2^m} \to \mathbb{F}_{2^m}, x \mapsto x^2$ is the Frobenius automorphism on $\mathbb{F}_{2^m}$, therefore every $y \in \mathbb{F}_{2^m}$ has a unique square root for any $m \geq 1$.*

Proof. The map is a ring homomorphism for any field of characteristic p prime since $p | \binom{p}{k}$ for any $1 \leq k \leq p-1$ with p prime. Moreover, it is injective since, as a ring homomorphism, its kernel is a proper ideal of $\mathbb{F}_{p^m}$ and a field only admits the $\{0\}$ proper ideal. Finally, since $\mathbb{F}_{2^m}$ is finite the map is also surjective. □

Lemma 3.36. *The Frobenius map :*

$$\mathbb{F}_{2^m}[X] \to \mathbb{F}_{2^m}[X], f(X) = \sum_{i=0}^{n} f_i X^i \mapsto (f(X))^2 = \sum_{i=0}^{n} f_i^2 X^{2i}$$

is an injective but not surjective ring homomorphism. Its image is $\mathbb{F}_{2^m}[X^2]$, a set of polynomials which are perfect squares on the ring $\mathbb{F}_{2^m}[X]$.

Proof. Follows from previous lemma 3.35 □

Now, because the polynomial $\sigma_e(X)$ is defined over a field of characteristic 2, lemma 3.36 guarantees that we can split it in squares and non-squares

$$\sigma_e(X) = \alpha^2(X) + X\beta^2(X)$$

Also, since the characteristic is 2 we have $\sigma_e' = \beta^2(X)$. Thus equation 3.3 can be rewritten as

$$\beta^2(X)(XS_e(X) + 1) \equiv \alpha^2(X) S_e(X) \mod g(X). \tag{3.4}$$

Let recall that the minimum distance d satisfies $d > 2t + 1$ from Lemma 3.18 Hence $0 < wt(e) = t < \lfloor \frac{d-1}{2} \rfloor$ and so we can assume that e is not a codeword, thus $S_e(X) \not\equiv 0 \mod g(X)$. Therefore, we can compute the inverse $T(X) = S_e^{-1}(X)$ by making use of the extended Euclidean algorithm. This takes $O(t^2 m^2)$ binary operations, as the computations are modulo $g(X)$, a polynomial of degree t and coefficients in $\mathbb{F}_2^m$. Multiplying equation 3.4 by $T(X)$ we obtain

$$\beta^2(X)(X + T(X)) \equiv \alpha^2(X) \mod g(X). \tag{3.5}$$

By 3.35, each element of $\mathbb{F}_{2^{mt}} \cong \frac{\mathbb{F}_{2^m}[X]}{g(X)}$ has a unique square root, let $\tau(X) \in \mathbb{F}_{2^m}[X]$ be the unique square root of the polynomial $T(X)+X$. We can compute

the square root of $T(X) + X$ in $O(t^2m^2)$ operations since it is a linear map on $\frac{F_{2^m}[X]}{g(X)}$. Taking the square root of equation 3.5 we obtain

$$\beta(X)\tau(X) \equiv \alpha(X) \bmod g(X) \tag{3.6}$$

To solve this equation for known $\tau(X)$ and $g(X)$, we have to determine $\alpha(X)$ and $\beta(X)$ of least degree. In fact, by assumption, we have $\deg(\sigma_e(X)) \leq t$ so $\deg(\alpha(X)) \leq \lfloor t/2 \rfloor$ and $\deg(\beta(X)) \leq \lfloor \frac{t-1}{2} \rfloor$ so there is a unique solution of equation 3.6 which can be found by applying the extended Euclidean algorithm. More explanations about the uniqueness of the solution are discussed in [4]. The algorithm proceeds as follows: at each step k, it computes the polynomials $\alpha_k(X) + \beta_k(X)\tau_k(X) \equiv 0 \bmod g(X)$ with $\deg(\alpha_k(X)) = \deg(g(X)) - \deg(\beta_{k-1}(X))$ and $\alpha_0(X) = g(X)$ and $\beta(X) = 0$. After each step, $deg(\beta)$ increases, and the degree of α decreases. We can see that there is a unique step in the computation of the Euclidean algorithm where the degree of both polynomials is below the respective bound.

Since the number of steps is bounded by t, the extended Euclidean algorithm takes $O(t^2m^2)$ binary operations. Finally, at the suitable step k we obtain $\sigma_e(X) = \alpha_k^2(X) + X\beta_k^2(X)$. Computing the zeroes of $\sigma_e(X)$ leads by definition to the error vector e, hence retrieving y and finally m by inverting the given generator matrix G.

In fact, the hardest part of the complete algorithm presented here is to find all roots of the error locator polynomial. This last step can be performed in $n(tm^2+tm)$ binary operations. Thus the whole error correction algorithm needs $O(n \cdot t \cdot m^2)$ binary operations.

4 Some attacks on the McEliece cryptosystem

An attacker who intercepted an encrypted message y has two possibilities to retrieve the original message m:

- find the private key or an equivalent one which allows to efficiently retrieve the message m as a trapdoor (structural attacks)
- decode y by brute force without knowing the private key (decoding attacks)

4.1 A classical structural attack

Not all classes of codes are secure in the McEliece cryptosystem. We study here a possible structural attack to break the McEliece cryptosystem based on GRS code. The idea of using GRS code came along with the wish to reduce the key size. It is Niederreiter who proposed this in his code-based cryptosystem which was later shown to be equivalent to McEliece's. For example for 80-bit security, a RSA public key of size 1024 bits is comparable to a 552000 bits key in the McEliece scheme which is essentially the size of G' and a little less for Niederreiter cryptosystem using Goppa codes. Meanwhile, using GRS code with $n = 663$, $k = 561$, and $m = 8$, the Niederreiter cryptosystem achieves 80-bits security with a key of size 8980 bits. However, it turns out that using a GRS code on the McEliece and Niederreiter cryptosystems is insecure. The McEliece cryptosystem based on GRS codes was broken by Sidelnikov and Shestakov in 1992.
Let us recall that a GRS code is characterized by two parameters α and v which determine the generator matrix and the check matrix.

Definition 4.1. *Two $[n,k]_q$ codes $\mathcal{C}$ and $\mathcal{C}'$ over a finite field $\mathbb{F}_q$ are called permutation equivalent if there exists a permutation π of the permutation group $\mathcal{S}_n$ such that*

$$\mathcal{C}' = \pi(\mathcal{C}) = \{(x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(n)} | x \in \mathcal{C}\}$$

Since multiplication with a permutation matrix to the right permutes the columns of a matrix, it directly follows that two linear codes $\mathcal{C}$ and $\mathcal{C}'$ are permutation equivalent if and only if any generator matrix of $\mathcal{C}$ is the product of any generator matrix of $\mathcal{C}'$ with a permutation matrix P to the right.

We recall that the public key of the McEliece cryptosystem is $G' = SGP$ with $G = G_{\alpha,v}$ for the parameters $\alpha, v \in \mathbb{F}_{2^m}^n$. Let denote $\tilde{G} = GP$. Then using the general form of $G_{\alpha,v}$ in definition 3.9, we conclude that $\tilde{G}$ is another generator matrix of an alternative GRS code of parameter $\tilde{\alpha} = \alpha$ and $\tilde{v} = \pi(v)$ with π being the permutation in the permutation group S_n associated with the permutation matrix P.

Lemma 4.2. *Let us define a $GRS_k(\alpha, v)$ over a finite field $\mathbb{F}_q$ with $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_q^n$ and $v = (v_1, \ldots, v_n) \in \mathbb{F}_q^n$, then*

$$GRS_k(\alpha, v) = GRS_k((a\alpha_1 + b, \ldots, a\alpha_n + b), (cv_1, \ldots, cv_n))$$

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

Proof. The proof can be found in [7] □

In particular, it follows that α_1 and α_2 can be fixed to arbitrary distinct values in $\mathbb{F}_{2^m}$.
The Sidelnikov-Shestakov attack shows that with the knowledge of the public key ($G' = SGP, t$), we can retrieve the alternative parameters $\tilde{\alpha}$ and $\tilde{v}$ as an alternative private key in polynomial time by solving some linear equations on a classical computer. We explain the attack based on the explanation of the attack from [13], with a bit more precision. The three steps of the attack are the following :

- Compute the reduced row echelon form $E(G')$ of G' :

$$E(G') = \begin{bmatrix} 1 & \ldots & 0 & b_{1,k+1} & \ldots & b_{1,n} \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \ldots & 1 & b_{k,k+1} & \ldots & b_{k,n} \end{bmatrix}$$

  This takes polynomial time in n.

- To calculate each $\tilde{\alpha}_j$, we first calculate $\tilde{\alpha}_j$ for $j = k+1, \ldots, n$ by picking two arbitrary rows such that the second one does not have a null j-th element. If the j-th column is the null column for j, since $E(G')$ is the reduced row echelon form of G' then j-th column of G' is also the null column, and since $\tilde{v_i} \neq 0 \ \forall i \in \{1, \ldots, n\}$, looking at the form of the generator matrix in 3.5 we conclude that $\tilde{\alpha}_j = 0$. Moreover, in practice having a null column in the generator matrix is irrelevant since every codeword would have a 0 at the corresponding position and so this would add memory for no gain of information.
  Let us say rows b_1 and b_2 satisfy this condition, i.e. $b_{2,j} \neq 0$. Then we set the polynomials $f_{b_i}(y)$, depending on $\tilde{\alpha}$ and $\tilde{v}$, to be polynomials of degree at most $k-1$ satisfying

$$(\tilde{v_1} f_{b_i}(\tilde{\alpha}_1), \ldots, \tilde{v_n} f_{b_i}(\tilde{\alpha}_n)) = (0, \ldots, 1, \ldots, 0, b_{i,k+1}, \ldots, b_{i,n}) = b_i \quad (4.1)$$

  for $i = 1, 2$. Since $\tilde{v}_i \neq 0$, f_{b_i} should have the following form

$$f_{b_i}(y) = c_{b_i} \cdot \prod_{j=1, j\neq i}^{k} (y - \tilde{\alpha}_i) \quad (4.2)$$

  and is said to be associated to row b_i of $E(G')$. The goal is to use 4.2 to obtain the following equalities

$$\frac{b_{1,j}}{b_{2,j}} = \frac{\tilde{v}_j \cdot f_{b_1}(\tilde{\alpha}_j)}{\tilde{v}_j \cdot f_{b_2}(\tilde{\alpha}_j)} = \frac{c_{b_1}(\tilde{\alpha}_j - \tilde{\alpha}_1)}{c_{b_2}(\tilde{\alpha}_j - \tilde{\alpha}_2)} \quad (4.3)$$

for $j = k+1, \ldots, n$, recalling that α_j, α_2 are distinct by definition. By lemma 4.2, we can assume that $\tilde{\alpha}_1 = 0$ and $\tilde{\alpha}_2 = 1$.

Then, $\tilde{\alpha}_j$ can be retrieved by computing

$$\tilde{\alpha}_j = \frac{\tilde{\alpha}_1 \frac{b_{1,j}}{b_{2,j}} - \tilde{\alpha}_2 \frac{c_{b_1}}{c_{b_2}}}{\frac{b_{1,j}}{b_{2,j}} - \frac{c_{b_1}}{c_{b_2}}}$$

for $j = k+1, \ldots, n$. Here all the variables are known except the ratio $\frac{c_{b_1}}{c_{b_2}} \in \mathbb{F}_{2^m}$. However, since the size of $\mathbb{F}_{2^m}$ is 2^m, we can just repeat the whole attack by guessing for the 2^m possibilities and stop when the deciphered text makes sense and can be considered to be the original message. Since m is typically of order 10 in practice, 2^m is a constant and does not change the complexity of the attack. Also, even if the message is not supposed to make sense, 2^m is low enough to keep in an acceptable reduced storage all potential plaintexts so that an attempt on this finite set is successful with too high probability to consider the cryptosystem as secure.

A posteriori, we see that since equation 4.3 is a ratio, we are not interested in the truly associated polynomials f_{b_1} and f_{b_2} but rather their spans

$$span(f_{b_i}) = \{f : \exists \lambda \in \mathbb{F}_{2^m} : f(y) = \lambda f_{b_i}\}, i = 1, 2$$

and finding the good guess reduces to finding a good fit or ratio in the set

$$\{\frac{f}{g} = \frac{\lambda_1}{\lambda_2} \frac{f_{b_1}}{f_{b_2}} : \lambda_1, \in \mathbb{F}_{2^m}, \lambda_2 \in \mathbb{F}_{2^m}^*\}$$

It remains to find $\alpha_3, \ldots, \alpha_k$. Therefore we replace the row b_2 by b_i for $(i = 3, \ldots, k)$ in the above equation 4.3, making sure that $b_{i,j} \neq 0$ and get

$$\frac{b_{i,j}}{b_{i,j}}(\tilde{\alpha}_j - \tilde{\alpha}_1) = \frac{c_{b_1}}{c_{b_i}}(\tilde{\alpha}_j - \tilde{\alpha}_i) \tag{4.4}$$

$\tilde{\alpha}_i$ and $\frac{c_{b_1}}{c_{b_i}}$ are unknown but choosing two values of j in $\{k+1, \ldots, n\}$ such that $b_{i,j} \neq 0$, we obtain a system of two linear equations and retrieve $\tilde{\alpha}_i$

In total, $\tilde{\alpha}$ can be calculated using $O(k^2 n)$ arithmetic operations in $\mathbb{F}_{2^m}$, hence $O(k^2 nm)$ bit operations since $\mathbb{F}_{2^m}$ can be seen as a $\mathbb{F}_2$-vector space of dimension m.

- To calculate $\tilde{v}$, we first find a non-trivial solution c of the linear system

$$G'_{\cdot,\{1,\ldots,k+1\}} \cdot c = 0 \tag{4.5}$$

where $\tilde{G}_{\cdot,\{1,\ldots,k+1\}}$ denotes the matrix $\tilde{G}$ restrained to the columns indexed by the set $\{1, \ldots, k+1\}$. Because $G'_{\cdot,\{1,\ldots,k+1\}} = S\tilde{G}_{\cdot,\{1,\ldots,k+1\}}$, c also solves

$$\tilde{G}_{\cdot,\{1,\ldots,k+1\}} \cdot c = 0$$

Noticing that $\tilde{G}$ is determined by $\tilde{\alpha}_i$ and $\tilde{v}$, knowing c from (4.5) and $\tilde{\alpha}_i$ from step 2 allows to retrieve $\tilde{v}_1, \ldots, \tilde{v}_{k+1}$. Hence we obtain $\tilde{G}_{\cdot,\{1,\ldots,k+1\}}$ Then we obtain an alternative generator matrix $\tilde{G}$ by computing

$$S = G'_{\cdot,\{1,\ldots,k\}} \tilde{G}^{-1}_{\cdot,\{1,\ldots,k\}} \text{ and } \tilde{G} = S^{-1} G'$$

and finally we can deduce $\tilde{v}_{k+2}, \ldots, \tilde{v}_n$ using $\tilde{G}$. The last step can be completed with $O(k^3 + k^2 n)$ bit operations.

Let us now explain how to use the alternative private key to decipher the message m from the cyphertext $c = mG' \oplus e = mS\tilde{G} \oplus e$. Knowing the alternative parameters $\tilde{\alpha}$ and $\tilde{v}$, we can apply the efficient $\lfloor \frac{n-k}{2} \rfloor$-errors decoding Berlekamp-Welch algorithm $\mathcal{D}_{\tilde{\mathcal{G}}}$ associated to the alternative GRS code generator to $c = mS\tilde{G} \oplus e$ and recover $mS\tilde{G}$ since $wt(e) = t \leq \lfloor \frac{n-k}{2} \rfloor$. Then we find a set $J \subseteq \{1, \ldots, n\}$ of size k such that $\tilde{G}_{\cdot J}$ is invertible. Such a set always exists since $\tilde{G}$ is of rank k and it can be found in polynomial time by solving a set of k linear equations. Finally, knowing S we can compute the original message $m = (mS\tilde{G})_J (\tilde{G}_{\cdot J})^{-1} S^{-1}$.

However, the Sidelnikov-Shestakov attack presented here which focused on the generator matrix perspective does not seem to apply to the McEliece cryptosystem based on Goppa codes. Multiplying the generator matrix with a random permutation matrix P makes the product $\tilde{G}$ a generator matrix of a Goppa code again since it essentially permutes the columns of a previous Goppa code, hence permutes the elements of α into $\tilde{\alpha}$ and keeps the Goppa polynomial identical. However, since a Goppa code is a subfield subcode of a GRS code, if we construct explicitly the generator matrix $\tilde{G} \in \mathbb{F}_2^{k \times n}$ with $k \geq n - mt$ then the matrix in the binary field has no apparent structure contrarily to the GRS case so the parameter $\tilde{\alpha}$ is hidden in this explicit construction and the situation is one degree more complex than the GRS's case. In fact, if we want to propose an attack inspired by the Sidelnikov-Sheskatov attack, we should add a step allowing us to cleverly convert elements in $\mathbb{F}_2$ to elements in $\mathbb{F}_{2^m}$, to eventually recover $\tilde{\alpha}$ and $g(x)$. To better understand what is happening, let us give an example:

Let $g(x) = \gamma^3 + x + x^2 \in \mathbb{F}_8[x]$ where γ is a primitive element of $\mathbb{F}_8$ that satisfies $\gamma^3 + \alpha + 1 = 0$. Let $\alpha = (0, 1, \gamma, \gamma + 1, \gamma^2, \gamma^2 + 1, \gamma^2 + \gamma, \gamma^2 + \gamma + 1)$, that is the vector containing all elements of $\mathbb{F}_8$. Then $n = 8, m = 3$ and $\mathcal{G}(\alpha, g(X)) = \{c \in \mathbb{F}_2^8 : cH^T = 0\}$ with

$$H = \begin{bmatrix} \gamma^4 & \gamma^4 & \gamma & 1 & \gamma & \gamma^2 & \gamma^2 & 1 \\ 0 & \gamma^4 & \gamma^2 & \gamma^2 & \gamma^4 & \gamma^6 & 1 & \gamma^6 \end{bmatrix}$$

which is a check matrix of a $GRS_6(\alpha, v)$ code for some $v \in \mathbb{F}_8^8$, where the associated vector is $v' = (\gamma^4, \gamma^4, \gamma, 1, \gamma, \gamma^2, \gamma^2, 1)$. Replacing each entry in H by a column vector in $\mathbb{F}_8$, we obtain a matrix $\bar{H}$ over the binary field which has

the following reduced row echelon form:

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

so that $\mathcal{G}(\alpha, g(X))$ admits a generator matrix

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

This generator matrix cannot be directly deduced from H without losing information on its predefined original structure from the GRS definition as we have explained in 3.7. In fact, we can see a Goppa code as the intersection between a GRS code and the binary field, intersecting those two unrelated mathematical objects seems to make the structure of the resulting code quite more subtle.

This is the same phenomenon that occurs for Elliptic curves and index calculus attacks. The index calculus is a method for efficiently solving the discrete logarithm problem in a finite field $\mathbb{F}_p$. However, no such method is known for elliptic curve discrete logarithms. Elliptic curve discrete logarithms are similar to discrete logarithms but we replace the field $\mathbb{F}_q^*$ by the group of rational points $E(\mathbb{F}_q)$ on an elliptic curve. More precisely, the Elliptic Curve Discrete Logarithm Problem" (ECDLP) is to find the discrete logarithm of a random elliptic curve element with respect to a publicly known base point on the elliptic curve. The security of elliptic curve cryptography depends on the ability to compute a point multiplication and the inability to compute the multiplicand given the original point and product point. Similar heuristic arguments than for Goppa codes suggest that the natural generalization of the index calculus to the elliptic curve discrete logarithm problem yields an algorithm that is super-polynomial and less efficient than a brute-force search algorithm (the reader can refer to [11] for more details).

Another essential fact that makes the attack works on GRS codes is Lemma 4.2, which says that GRS codes are over-determined by their parameters and this can be exploited to recover $\tilde{\alpha}$ in the attack by arbitrarily setting $\tilde{\alpha}_1, \tilde{\alpha}_2 = 0, 1$. We can wonder whether this is the case for Goppa codes. Let $\tilde{\alpha}, \tilde{g}(x)$ be Goppa codes parameters such that

$$\mathcal{G}(\tilde{\alpha}, \tilde{g}(x)) = \mathcal{G}(\alpha, g(x)) \tag{4.6}$$

with $\deg(\tilde{g}) = t$. Using the complete description from 3.21, we must have

$$\mathcal{G}(\tilde{\alpha}, \tilde{g}(x)) = GRS_{n-t}(\tilde{\alpha}, \tilde{v})|_{\mathbb{F}_2^n}$$

where $\tilde{v} = (\tilde{v}_1, \ldots, \tilde{v}_n)$ with $\tilde{v}_i = g(\tilde{\alpha}_i)/(\prod_{j \neq i}(\tilde{\alpha}_i - \tilde{\alpha}_j))$ for $i \in \{1, \ldots, n\}$ Noticing that we have a similar equality $\mathcal{G}(\alpha, v) = GRS_{n-t}(\alpha, v)|_{\mathbb{F}_2^n}$, using Lemma 4.2 we conclude that equation 4.6 is satisfied if

$$\tilde{\alpha} = a\alpha + b \text{ and } \tilde{g}(x) = cg(x)$$

for all $a, b, c \in \mathbb{F}_q$ and $a, c \neq 0$, Here $a\alpha + b = (a\alpha_1 + b, \ldots, a\alpha_n + b)$ and we also used the fact that $\deg(\tilde{g}) = t$ so it is determined by its values at n points $\alpha_1, \ldots, \alpha_n$. Hence, Goppa codes are at least as determined as GRS codes, and a similar lemma as Lemma 4.2 might be exploited for a structural attack using alternative parameters.

We presented the attack using the generator matrix perspective, this gives us an interesting and intuitive point of view but it makes a transposition attempt to Goppa code difficult since as we explained, constructing explicitly the generator matrix from a GRS code takes several steps. However, it is also possible to formulate the attack via check matrices which was the perspective Sidelnikov and Shestakov originally took. This could potentially make a transposition attempt more accessible.

To summarise, Sidelnokov and Shestakov's attack can efficiently compute the matrices S and GP from the public matrix $G' = SMP$ if the underlying code is a generalized Reed-Solomon code. This attack can reveal the structure of an alternative code but does not reveal the secret permutation P. There exist plenty of other interesting attempts using various techniques which show that one should be careful concerning the family of codes used in the McEliece cryptosystem. For instance, in 2001, P. Loidreau and N. Sendrier proposed an efficient way to know whether the Goppa polynomial used for the Goppa code is a binary generator polynomial $g \in \mathbb{F}_2[X]$. If this is the case, then this reduces the search space of a brute force attack on the private key to polynomial time using the so-called *support splitting algorithm* and finally allows us to find the permutation P.

4.2 Decoding attacks

Decoding attacks consist in using decoding techniques of coding theory to retrieve the message without having any trapdoor information. These decoding techniques all consist in searching for a good candidate among a set of potential candidates that could lead to the original message. For this reason, it is also called a brute force attack.

4.3 Information set decoding

We saw in the introduction some basic decoding techniques. A first technique was based on the minimal distance of a code which lead to the nearest neighbor decoding principle. A second technique took advantage of the fact that the codes are linear and we can define syndromes from cyphertexts applied to the check matrix, a procedure that searches for the position of the errors of the vector e. Here we present an even more clever way to decipher any linear codes based on the observation that it is easier to guess enough error-free positions than it is to guess all the error positions.
Indeed, to give a concrete example, the security parameters suggested by McEliece in its original paper were $n = 1024, k = 524, t = 50$. Recalling that a Goppa code can correct up to $t = wt(e)$ errors, we will see that the error positions are in proportion far less than the proportion of error-free positions needed to efficiently recover the message m, which turns out to be $\frac{n-k}{n}$ as we will show through the approach introduced by Prange and McEliece, a first information set decoding algorithm (ISD).
Let us notice that the public key $G' = SGP$ from McEliece cryptosystem is again a generator matrix of a linear code since it is a product of invertible binary matrices. Hence, let us assume we are given a generator matrix G of a $[n, k]$ linear code, the attack solves the general decoding problem by using the notion of *information set* and proceeds as follows. Let choose randomly $\mathcal{I} \subset \{0, \dots, n-1\}$ with $|\mathcal{I}| = k$. If we denote c and e the cyphertext and error vector, we have :

$$c_{\mathcal{I}} = mG_{\cdot,\mathcal{I}} \oplus e_{\mathcal{I}}$$

where $c_{\mathcal{I}}$ and $e_{\mathcal{I}}$ denote the restriction of the vectors to the elements indexed by $\mathcal{I}$. If $G_{\cdot,\mathcal{I}}$ is non-singular and $e_{\mathcal{I}} = 0$, then

$$m = c_{\mathcal{I}} G_{\cdot,\mathcal{I}}^{-1}$$

So repeating this procedure until we get a suitable set $\mathcal{I}$ allows us to retrieve the message m. In fact, we only know that we indeed found an information set when the computation $c_{\mathcal{I}} G_{\cdot,\mathcal{I}}^{-1}$ produces a plaintext that does make sense. The set $\mathcal{I}$ is called an information set for two reasons. First, it only contains indexes i for which $e_i = 0$, the so-called error-free positions. Secondly, the number k of such positions is the quantity needed to potentially invert $G_{\cdot,\mathcal{I}}$ and if it is the case, $\mathcal{I}$ indicates where to find the relevant piece of information in the public key G and the cyphertext c to recover m.

However, finding an information set $\mathcal{I}$ takes super-polynomial time because the set of all the information sets is a priori unstructured among all the potential subsets of size k. Since we randomly choose $\mathcal{I} \subset \{1, \ldots, n\}$ of size k, there are $\binom{n}{k}$ choices. Also, since we want $\mathcal{I}$ to be error-free, that is $e_{\mathcal{I}} = 0$, the good subsets have their k elements among the remaining $n - t$ error-free positions among $\{1, \ldots, n\}$. Finally, the procedure succeeds only if $G_{.,\mathcal{I}}$ is indeed invertible. It is known that approximately 29% of all $k \times k$ matrices over $\mathbb{F}_2$ are invertible. Hence, since $\mathcal{I}$ is randomly chosen and G is a matrix pre-scrambled by the random scrambler matrix S of size $n \times n$ from the McEliece cryptosystem, it is reasonable to think that $G_{.,\mathcal{I}}$ in average over $\mathcal{I}$ behaves as an arbitrary matrix of size $k \times k$ and is invertible with probability 0.29, independently of the $e_{\mathcal{I}}$, although this should be proved rigorously. This last point is discussed in [2]. Hence, the probability of $\mathcal{I}$ being an information set is :

$$Pr[\mathcal{I} \text{ is an information set}] = 0.29 \cdot \frac{\binom{n-t}{k}}{\binom{n}{k}}$$

Since inverting a $k \times k$ matrix takes $O(k^3)$ bit operations, using the asymptotic equivalence $\binom{n}{k} \sim \frac{n^k}{k!}$ derived from Stirling's formula we conclude that Prange's based ISD algorithm takes an expected number of bit operations :

$$T \approx \frac{k^3}{Pr[Pgood]} \approx 3.45 \cdot k^3 \left(1 - \frac{t}{n}\right)^{-k} \tag{4.7}$$

The original parameters proposed in McEliece cryptosystem are

$$n = 1024, m = 10, t = 50, k = n - mt = 524$$

The work factor to discover the message using those parameters is approximately

$$1.9 \cdot 10^{24} \simeq 2^{79.7}$$

which is too high to conduct any practical attacks.

This computation shows the work factor needed on a given set of parameters for a fixed Goppa code. It is interesting to study how it scales asymptotically. To this end, we define the rate $R = \frac{k}{n}$ which is constant in practice for a given rate of information transmission. Also, the typical value used for Goppa codes is $t \approx (1 - R)n/log(n)$ (see [2]). Then we can make the following approximations:

$$\frac{\binom{n-t}{k}}{\binom{n}{k}} = \frac{n \ldots (n-t+1)}{(n-k) \ldots n-k-t+1} \approx \left(\frac{n}{n-k}\right)^t = \left(\frac{1}{1-R}\right)^t = c^{(1+o(1))\frac{n}{log(n)}}$$

for $c = \left(\frac{1}{1-R}\right)^{1-R}$. Hence we see that the running time is super-polynomial in the codelength n.

This first approach of ISD was introduced by Prange and McEliece, called the information-set decoding algorithm. We can generalize this approach by allowing some error positions in $\mathcal{I}$. Intuitively, it works as a trade-off between increasing the probability to find a good set $\mathcal{I}$ and the cost of doing an enumeration of the possible $e_{\mathcal{I}}$ giving the good weight t. Indeed, If $G_{\cdot,\mathcal{I}}$ is non-singular and $e_{\mathcal{I}} = t$, then m can be recovered by doing a reasonably short enumeration of the candidates $e_{\mathcal{I}}$ and check if

$$wt((c_{\mathcal{I}} \oplus e_{\mathcal{I}})G_{\cdot,\mathcal{I}}^{-1}G \oplus c) = t$$

It turns out that this trade-off is advantageous. Indeed, the number of sets $\mathcal{I}$ such that there are exactly i errors in vector $c_{\mathcal{I}}$ is $\binom{t}{i}\binom{n-t}{k-i}$. Therefore, the expected number of attempts before finding a set $\mathcal{I}$ such that there are at most j errors in vector $c_{\mathcal{I}}$ is

$$T_j = 3.45 \cdot \frac{\binom{n}{k}}{\sum_{i=0}^{j}\binom{t}{i}\binom{n-t}{k-i}}$$

which is less than T from (4.7) However, one also has to count the possibilities of choosing the vector $e_{\mathcal{I}}$ with $wt(e_{\mathcal{I}}) \leq j$ which is

$$N_j = \sum_{i=0}^{j}\binom{k}{i}$$

and then to go through the elements of $e_{\mathcal{I}}$ to check if $wt((c_{\mathcal{I}} \oplus e_{\mathcal{I}})G_{\cdot,\mathcal{I}}^{-1}G \oplus c) = t$. Overall, the expected work factor for the attack is

$$W_j = \alpha T_j(k^3 + N_j k)$$

for some constant α. Although the N_j quantity add some computation, W_j is eventually less than T and is minimized for $j = 2$ (see [5]), but is still super-polynomial in n.

ISD techniques have been studied a lot since Prange's version, and a lot of other improvements can be done to reduce even more the costs of computation. However, the final cost is still asymptotically $c^{(1+o(1))\frac{n}{log(n)}}$ which is super-polynomial in n.

4.4 Making use of Grover's quantum algorithm

Grover's algorithm is a quantum algorithm that aims to find the roots of an arbitrary function of finite input space. For that reason, Grover's algorithm can be seen as a way to search target elements in an unstructured data basis

if we can implement an oracle that computes a function returning 0 on target elements. This oracle is implemented via a logical circuit in practice. Taking advantage of the laws of quantum physic, Grover's algorithm finds the roots with a quadratic speed up compared to checking each argument one by one.
In 2010, D. Bernstein showed that Grover's algorithm could be used for a quadratic speed-up of Prange's algorithm by constructing a suitable oracle. We describe the quantization of Prange's algorithm that he proposed in [2].
Let consider the following function of parameters $y \in \mathbb{F}_2^n$ and $t \in \mathbb{N}^*$ given the subset argument $\mathcal{I} \subset [n]$ does :

- invert $G_{\cdot,\mathcal{I}}$ if possible, otherwise return 1
- compute $v = y_{\mathcal{I}} G_{\cdot,\mathcal{I}}^{-1} \in \mathbb{F}_2^k$
- return 0 if $wt(vG - y) = t$
- return 1 otherwise

Clearly, the roots of this function are the information sets and v calculated at the second step is the original message if $wt(vG - y) = t$. Bernstein argues that this function can be computed by a combinatorial circuit and computes outputs in $O(n^3)$ bit operations. Combinatorial circuits use NAND logical gates which can easily be translated into quantum reversible gates ready to be used inside Grover's algorithm. Hence, since Grover's algorithm offers a quadratic speed-up for finding roots in unstructured functions, then implementing this ISD function oracle on Grover's algorithm would lead to a quadratic speed-up for finding an information set and so decoding the cyphertext. As a consequence, since Prange's algorithm has to perform $c^{(1+o(1))\frac{n}{log(n)}}$ iterations of the above function before finding an information set, this means that Grover's algorithm only uses $\sqrt{c^{(1+o(1))\frac{n}{log(n)}}} = c^{(1/2+o(1))\frac{n}{log(n)}}$ iterations. Each iteration has computation dominated by the quantum function evaluation of the quantum combinatorial circuit which uses $O(n^3)$ qubit operations so it takes time $n^{O(1)}$. Overall, the work factor needed to find $\mathcal{I}$ remains $c^{(1/2+o(1))\frac{n}{log(n)}}$, which is superpolynomial in n.

Although the improvement of Grover's quantum algorithm lets the work factor superpolynomial in n, the key size is quadrupling which is not negligible. One has to replace n by $(2 + o(1))n$ to keep the same security level, that is keeping the same work factor for quantum computers compared to the case of classical computers.

An interesting feature to have in mind is that Grover's algorithm is asymptotically optimal among all possible quantum search algorithms for unstructured data (see [9]). This means that for large n, no better improvement can be done than a quadratic speed-up. This perspective gives some more guarantees for the security of the McEliece cryptosystem subject to brute-force attacks, that is attacks without taking advantage of existing structures in the cryptosystem.

4.5 Quantum Fourier Sampling attack

Finally, we mention another result suggesting the security of McEliece cryptosystem in post-quantum cryptography. The majority of quantum algorithms that offer an exponential speed-up solve an instance of the *Hidden Subgroup Problem* using the quantum Fourier sampling technique. Here we won't explain this technique but it can be found in section 3 of [3]. Let us define the Hidden Subgroup Problem.

Definition 4.3 (Hidden Subgroup Problem (HSP))**.** *Given a function f on a group G and a subgroup $H \leq G$ such that f is constant on each left coset of H and distinct on different left cosets, find a set of generators for H.*

For instance, the problem of factoring numbers reduces to an order-finding problem, a specific instance of the Hidden Subgroup Problem. Shor's algorithm uses a quantum Fourier Sampling attack to solve it in polynomial time and break the RSA cryptosystem. More specifically, it uses quantum Fourier Sampling over the cyclic group $\mathbb{Z}_N$. Similarly, Simon's quantum algorithm which inspired Shor's algorithm uses quantum Fourier Sampling over $\mathbb{Z}_2^n$. The question that arises is whether this technique can work for the McEliece problem 3.8.
Let us mention the results presented in [3]. Let us assume we are given the hint of the original matrix G from the private key, then the McEliece problem reduces to the following problem:

Definition 4.4 (Scrambler-Permutation Problem (SPP))**.** *Given two $k \times n$ matrices G and G' with entries in a finite field $\mathbb{F}_q$ such that $G' = SGP$ for some $S \in GL_k(\mathbb{F}_q)$ and some $n \times n$ permutation matrix P, find such a pair $(S; P)$.*

However, in [3] it is shown that SPP reduces to a specific case of HSP on a non-abelian group G such that the quantum Fourier Sampling attack fails to solve. In fact, it is known that the quantum Fourier Sampling attack works when G is abelian, but it is not known to work in general for non-abelian groups.

5 Conclusion

The goal of this semester project was to study McEliece cryptosystem against classical and quantum computers. We saw that the robustness of the McEliece cryptosystem, like any cryptosystem, relies on subtle mathematical objects that need to be used carefully to prevent structural attacks. For the moment, neither classical nor quantum attacks succeeded to break the McEliece cryptosystem based on Goppa codes. We do not know if a classical method could break the cryptosystem, but the way Goppa codes are defined in terms of subfield subcodes requests at least a deeper thinking in understanding the hidden structure behind the McEliece problem for Goppa codes. To this end, it could be fruitful to vary the point of view, for example by investigating attacks focused on the check matrix rather than the generator matrix in future works. Besides, we saw that quantum attacks fail because the current known methods to solve specific mathematical problems by harnessing quantum laws are not suitable for the McEliece problem, even though this is not specific to the use of Goppa codes. However, improvements and a better understanding of quantum computation in the coming years could give birth to innovative methods.

6 Acknowledgements

We want to thank Vlad Serban for his helpful suggestions and his careful review of the report.

References

[1] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. "On the inherent intractability of certain coding problems (corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386.

[2] Daniel J Bernstein. "Grover vs. mceliece". In: *Post-Quantum Cryptography: Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings 3*. Springer. 2010, pp. 73–80.

[3] Hang Dinh, Cristopher Moore, and Alexander Russell. "McEliece and Niederreiter cryptosystems that resist quantum Fourier sampling attacks". In: *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*. Springer. 2011, pp. 761–779.

[4] Daniela Engelbert, Raphael Overbeck, and Arthur Schmidt. "A summary of McEliece-type cryptosystems and their security". In: *Journal of Mathematical Cryptology* 1.2 (2007), pp. 151–199.

[5] Pil Joong Lee and Ernest F Brickell. "An observation on the security of McEliece's public-key cryptosystem". In: *Advances in Cryptology—EUROCRYPT'88: Workshop on the Theory and Application of Cryptographic Techniques Davos, Switzerland, May 25–27, 1988 Proceedings 7*. Springer. 1988, pp. 275–280.

[6] San Ling and Chaoping Xing. *Coding theory: a first course*. Cambridge University Press, 2004.

[7] Florence J Macwilliams, NJA Sloane, and Neil J Sloane. *The Theory of Error-Correcting Codes (North-Holland Mathematical Library)*. 1983.

[8] Robert J McEliece. "A public-key cryptosystem based on algebraic coding thoery". In: *The Deep Space Network Progress Report* (1978), pp. 114–116.

[9] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[10] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332.

[11] Joseph H Silverman and Joe Suzuki. "Elliptic curve discrete logarithms and the index calculus". In: *Advances in Cryptology—ASIACRYPT'98: International Conference on the Theory and Application of Cryptology and Information Security Beijing, China, October 18–22, 1998 Proceedings*. Springer. 1998, pp. 110–125.

[12] Henning Stichtenoth. *Algebraic function fields and codes*. Vol. 254. Springer Science & Business Media, 2009.

[13] Christian Wieschebrink. "Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes". In: *Cryptology ePrint Archive* (2009).