An introduction to code-based cryptography

A report for the course MO421 at Unicamp

Fábio C. C. Meneghetti

IMECC — Unicamp

November 24, 2021

Contents

1 Introduction

Quantum computers can break the security systems we use on banks and the internet. While efficient quantum computers have not been built yet, the research has been advancing rapidly, and this might be a real worry somewhere in the future. This calls for the development of the so-called post-quantum cryptography, which is built to be resilient to those attacks. A strong candidate is code-based cryptography: the use of systems based on the theory of error-correcting codes — which were originally intended to provide reliability to noisy channels, not security. This text is a brief introduction to this new area.

In chapter 2, we present a short historical background of the development of information theory, coding theory and the use of these ideas in cryptographic systems. In part 3, we present the mathematical aspects of coding theory and some concepts which are important for code-based cryptography, such as Goppa codes. In part 4, we present two of the most important schemes of code-based cryptography — McEliece and Niederreiter — as well as discuss their relationship and possible attacks to their security.

The use of coding theory to develop cryptographic systems is a rather unexpected connection between two rich areas of research. The author hopes this text serves as a introduction, as well as a complement to the reader who is already familiar with either one of these areas, and has interest in expanding her/his knowledge about this connection.

2 Historical background

A revolution broke out in 1948: it was the year the electrical engineer and mathematician Claude Shannon published his breakthrough paper "A Mathematical Theory of Communication". So breakthrough, indeed, that one year later it was republished as a book, with the more incisive title "**The** Mathematical Theory of Communication". This publication gave birth to the field now known as information theory, together with a multitude of subfields with wide-ranging applications such as: economics, cryptography, mathematics, statistics, physics, biology, philosophy and so on.

The theory Shannon presented was a probabilistic one: he proposed a way to measure information using uncertainty, with the help of a concept he developed called entropy. Furthermore, he described with mathematical precision the notion of a communication channel, which could have some interference or noise, and explained how his probabilistic theory gives a natural way to design efficient codes for each channel. The main objective of these codes was to provide reliability, that is, to allow one to decode the original message with low probability of error, despite the noise of the channel.

Shannon hit the nail on the head. His theory was exactly what engineers needed to provide efficient and reliable transmission of data in all the systems that were emerging. To name a few, his ideas were used in the following communication channels: radio transmission, the telephone, TV, CD, DVD; and today they continue to be essential in cellular, WiFi, the internet of things, QR codes, and many others.

As time went on and the field expanded, people started to realize many applications didn't need codes with a probabilistic construction, but rather that an algebraic or combinatorial one would do. Most notably, in 1968 Richard Hamming proposed a very important family of codes called Hamming codes, together a set of tools such as the Hamming distance. These ideas materialized into a field today called linear coding, which studies error-correcting codes based on linear algebra.

Every area of mathematics seems to have their own space in coding theory. In the late 70's and early 80's, mathematician Valery Goppa produced a series of papers [5, 4] introducing the world to algebraic-geometric codes, or Goppa codes. The idea is that by using algebraic curves over a finite field, one can construct linear codes whose parameters are determined by the curve's parameters. The main importance of Goppa's construction is that these codes have very efficient decoding algorithms, making them especially useful for many applications.

Despite being conceived originally for reliability, codes have found a important use in security too. Robert J. McEliece was the first to introduce the use of binary Goppa codes to construct a public-key cryptographic scheme [11], where the objective is for an interceptor in the channel to be unable to decode the original message. What was found is that by adding noise and scramble, a system for error correction can be adapted into security system. Today, there are many modern

cryptoschemes based on the original ideas of McEliece and Niederreiter, and almost all of them use the powerful tool of Goppa codes. They are collectively called code-based cryptoschemes.

Much interest started to develop for code-based cryptography with the discovery that many of these algorithms are great candidates for post-quantum cryptosystems. What this means is that they are believed to be secure against quantum computers, whereas most of the cryptography used today is not. For example: RSA, the security used in most banks, can potentially be broken by Peter Shor's 1994 algorithm for integer factorization. This is not a short-term worry, since efficient quantum computers have not been invented yet. But mid and long-term, this may pose a threat to security worldwide, and by this reason the National Institute of Standards and Technology (NIST) of the United States created a challenge to develop a standard post-quantum cryptosystem (`https://csrc.nist.gov/projects/post-quantum-cryptography`). Until the writing of this text, it has not yet concluded. But one strong candidate is called Classic McEliece [7], and is a modern take on the original McEliece implementation.

The translation of ideas created for error-correction and reliability into that of cryptographic systems is an elegant achievement. By this and by the simplicity of their description, potential security against quantum computers, and relative easiness of implementation, code-based cryptoschemes may be increasingly present in the language and minds of cryptographers and cryptanalysts.

3 Mathematical background

3.1 Finite fields

Definition 3.1.1. A **commutative group** is a set G together with an operation $*\colon G \times G \to G$ such that the following properties hold, for every $a, b, c \in G$:

1. $(a * b) * c = a * (b * c)$ (associativity)
2. there is an element $e \in G$ such that $a * e = e * a = a$ (identity)
3. for every $a \in G$ there is an element $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$ (inverse)
4. $a * b = b * a$ (commutativity)

Definition 3.1.2. A **field** is a set K together with two operations $+$ and $\cdot$, such that:

1. K with $+$ is a commutative group (the identity is called 0 and the inverse $-a$),
2. $K \setminus \{0\}$ with $\cdot$ is a commutative group (the identity is called 1 and the inverse a^{-1}),
3. $a \cdot (b + c) = a \cdot b + a \cdot c$ for every $a, b, c \in K$ (distributivity).

A **finite** field is simply a field with a finite number of elements (also called a **Galois field**). We begin with a theorem of finite fields.

Theorem 3.1.3. Let $q \in \mathbb{N}$. There is a field with q elements if, and only if, $q = p^n$ for some prime p and some $n \in \mathbb{N}$. If that is the case, then the field is unique up to isomorphism.

The "up to isomorphism" can be understood as: if there are two fields with q elements, then it is possible to go from one to the other by just renaming the elements accordingly. They're essentially the same field. From now on, we use $\mathbb{F}_q$ to denote the finite field with q elements.

We won't show the general construction of the Galois field, which requires quotients by ideals in the ring of polynomials. If, however, $q = p$ is a prime, the Galois field can be easily constructed over the set $\mathbb{F}_p = \{0, 1, \ldots, (p-1)\}$, with the operations of sum and product modulo p:[1]

$$a + b \coloneqq a +_{\mathbb{Z}} b \pmod{p}, \text{ and} \qquad a \cdot b \coloneqq a \cdot_{\mathbb{Z}} b \pmod{p}.$$

For example, in the binary field $\mathbb{F}_2 = \{0, 1\}$, sum and product are given by the following tables.

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

In the field $\mathbb{F}_3 = \{0, 1, 2\}$, they are given by:

[1] Here, $+_{\mathbb{Z}}$ and $\cdot_{\mathbb{Z}}$ denote the standard sum and multiplication of integers.

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

·	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

3.2 Coding theory

Since $\mathbb{F}_q$ is a field, $\mathbb{F}_q^n$ is a n-dimensional vector space. A **linear code** C is defined as a subspace of $\mathbb{F}_q^n$. We call n the *length* of the code, $k = \dim(C)$ the *dimension*, and q the *alphabet size*. In this case, we say it is a **linear $[n, k]_q$-code**.

Example 3.2.1. The set $C = \{(0, 0), (1, 2), (2, 1)\}$ is a linear $[2, 1]_3$-code.

Definition 3.2.2. A **generator matrix** for a linear $[n, k]_q$-code C is a $k \times n$ matrix G with entries in $\mathbb{F}_q$, that generates C, in the sense that[2]

$$C = \left\{ xG \;\middle|\; x \in \mathbb{F}_q^k \right\}.$$

In the standard terminology, an element $c \in C$ is called a **codeword**. It is also important to have notion of distance between codewords. For codes, the most important one is the Hamming distance.

Definition 3.2.3. The **Hamming distance** in a code $C \subset \mathbb{F}_q^n$ is the function $d\colon C \times C \to \mathbb{R}^+$ given by the number of coordinates in which two codewords differ. In mathematical terms, if $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are codewords, then

$$d(x, y) = \left|\{i : x_i \neq y_i\}\right|.$$

For example, the Hamming distance between $(1, 0, 2, 1)$ and $(0, 0, 0, 1)$ as codewords in $\mathbb{F}_3^4$ is 2. The Hamming distance satisfies the axioms of a metric distance, which means that for every $x, y, z \in C$, we have

(i) $d(x, y) \geq 0$ with equality iff $x = y$ (non-negativity),

(ii) $d(x, y) = d(y, x)$ (symmetry), and

(iii) $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

Definition 3.2.4. The **minimum distance** of a code C is

$$d(C) \coloneqq \min\left\{d(x, y) : x, y \in C, x \neq y\right\}.$$

We then say that C is a linear $[n, k, d]_q$-code.

A similar notion to that of Hamming distance is the **Hamming weight**, defined as $w(x) \coloneqq d(x, 0)$. By linearity, finding the minimum distance of a code is equivalent to finding the minimum weight: $d(C) = \min_{x \in C} w(x)$.

[2] In coding theory it is the standard to denote matrix multiplication with a vector from the left, and not from the right as in most linear algebra courses.

We define the *product*[3] between codewords $x, y \in \mathbb{F}_q^n$ as $\langle x, y\rangle \coloneqq \sum_{i=1}^n x_i y_i$. If two codewords x, y satisfy $\langle x, y\rangle = 0$ we say they are *orthogonal*.

Definition 3.2.5. The **dual code** (or *orthogonal code*) $C^\perp$ of a linear code $C \subset \mathbb{F}_q^n$ is given by the set of all codewords $y \in \mathbb{F}_q^n$ such that

$$\langle x, y\rangle \coloneqq \sum_{i=1}^n x_i y_i = 0, \qquad \forall x \in C. \tag{3.1}$$

It is easy to see that $C^\perp$ is in fact a linear code, using the fact that the product $\langle \cdot, \cdot \rangle$ is bilinear: $\langle x, y + \alpha\tilde{y}\rangle = \langle x, y\rangle + \alpha\langle x, \tilde{y}\rangle$.

The dimensions of dual codes add up to the dimension of the space: $\dim C + \dim C^\perp = n$, and therefore $\dim C^\perp = (n - k)$. Another property is that the dual is involutive: $(C^\perp)^\perp = C$.

Definition 3.2.6. A **parity-check matrix** of a $[n, k]$-code C is a generator matrix H for the dual code $C^\perp$, with dimensions $(n - k) \times n$.

We note the important fact that $GH^\top = 0$ is a matrix of k by $n - k$ zeros, a fact that follows from orthogonality.

Example 3.2.7. Consider the linear $[3, 2]_3$-code

$$C = \{(0,0,0), (1,2,1), (2,1,2), (1,1,1), (2,2,2), (2,0,2), (1,0,1), (0,1,0), (0,2,0)\}.$$

The minimum distance is $d(C) = 1$. A generator matrix for this code is

$$G = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

The dual code is $C^\perp = \{(0,0,0), (1,0,2), (2,0,1)\}$, and a parity-check matrix is $H = \begin{bmatrix} 1 & 0 & 2 \end{bmatrix}$.

The purpose of error-correcting codes is to allow error correction in the transmission of a message. The way to do this is by encoding messages as codewords in some code C. This can be done by the natural one-to-one mapping $x \mapsto xG$ induced by the generator matrix.

Suppose, for instance, Alice encodes a message $x \in \mathbb{F}_q^k$ as $y = xG$, and sends it to Bob. However, the channel of transmission is noisy and errors can occur; so Bob may, instead, receive the codeword $\tilde{y} = y + e$, where $e \in \mathbb{F}_q^n$ is the error. If this error is sufficiently small, $\tilde{y}$ will not be in C and Bob will know there is an error.

Bob will then decode $\tilde{y}$ using a method called *nearest neighbor decoding*: he will find a codeword z in the code C that minimizes the Hamming distance from $\tilde{y}$.

Definition 3.2.8. A **nearest neighbor decoder** for a $[n, k]_q$-code C is a function $\mathrm{Dec}\colon \mathbb{F}_q^n \to C$ such that[4]

$$\mathrm{Dec}(y) \in \operatorname*{argmin}_{z \in C} d(y, z), \quad \forall y \in \mathbb{F}_q^n.$$

[3] In general this product is not a inner product; for instance, if $n = q$, the one-vector $\mathbf{1} = (1, \ldots, 1) \in \mathbb{F}_q^n$ satisfies $\langle \mathbf{1}, \mathbf{1}\rangle = 0$.

[4] Note that two decoders will only differ for $y \in \mathbb{F}_q^n$ that are "in the middle" of two or more legitimate codewords.

Bob will then be able to decode the message from the code by doing $\tilde{x} = G^{-1}(\mathrm{Dec}(\tilde{y}))$. If at most $\lfloor (d-1)/2 \rfloor$ occurred, Bob will be able to decode the original message. In other words, if $w(e) \leq \lfloor (d-1)/2 \rfloor$, then $y = \mathrm{Dec}(y+e)$ for all $y \in C$. This amount $t = \lfloor (d-1)/2 \rfloor$ is called the **error-correcting capacity** of the code C.

However, if Bob was to compare $\tilde{y}$ to each codeword in C in order to find the closest one, the algorithm would not be polynomial-time. But he can take advantage of the linear subspace structure of C, using a method called *syndrome decoding*. Fix a parity-check matrix H for the code C. Given $y \in \mathbb{F}_q^n$, the *syndrome* of y is defined as the vector

$$\mathrm{Syn}_H(y) := Hy^\top.$$

From now on we suppose there's a fixed parity-check matrix and denote the syndrome as just Syn. The following lemma shows that syndromes can identify codewords.

Lemma 3.2.9. Let C be a linear $[n,k]_q$-code, and $y \in \mathbb{F}_q^n$. Then $y \in C$ if and only if $\mathrm{Syn}(y) = 0$.

Proof. Suppose y is a codeword, that is, $y = xG$ for some $x \in \mathbb{F}_q^k$. So $\mathrm{Syn}(y) = HG^\top x^\top$. But $HG^\top = 0$ by orthogonality of the dual code; hence $\mathrm{Syn}(y) = 0$. If, on the other hand, $\mathrm{Syn}(y) = 0$ then $Hy^\top = 0$. This means y is orthogonal to each codeword of $C^\perp$, so $y \in (C^\perp)^\perp = C$. □

A consequence of this lemma is that if $y \in C$, then $\mathrm{Syn}(y+e) = \mathrm{Syn}(e)$ for all $e \in \mathbb{F}_q^n$. Therefore, the syndrome of the received vector $\tilde{y}$ will be the syndrome of the error, and will be zero if there is no error. This suggests the syndrome as a measure of error in the received vector, and the following algorithm for syndrome decoding:

Step 1: if $\mathrm{Syn}(\tilde{y}) = 0$, we suppose there is no error and decode $\tilde{y}$ as $\tilde{y}$.
Step 2: if, on the other hand, $\mathrm{Syn}(\tilde{y}) \neq 0$, start a **for** loop on the weights $w = 1, \ldots, \left\lfloor \frac{d-1}{2} \right\rfloor$. For each codeword e of weight w, compare the syndromes $\mathrm{Syn}(e)$ with $\mathrm{Syn}(\tilde{y})$. If they are equal, decode $\tilde{y}$ as $\tilde{y} - e$ and return. If not, continue the loop.
Final step: If loop ends without return, we conclude that more than $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors occurred.

And the algorithm works:

Proposition 3.2.10. If $w(e) \leq \lfloor (d-1)/2 \rfloor$, then the algorithm shown above recovers the original vector.

Proof. Suppose Alice sends y and Bob receives $\tilde{y} = y + e$, with $w(e) \leq \lfloor (d-1)/2 \rfloor$. Bob then runs the above algorithm and obtains $\tilde{y} - r$, with $w(r) \leq \lfloor (d-1)/2 \rfloor$ and $\mathrm{Syn}(\tilde{y}) = \mathrm{Syn}(r)$. So we have that $\mathrm{Syn}(\tilde{y} - r) = 0$ and therefore $\tilde{y} - r \in C$. We have to show that $\tilde{y} - r = y$.

But $\tilde{y} - r = y + (e - r)$ and $d(y, y + (e-r)) = w(e-r) \leq w(e) + w(r) \leq (d-1)$. Since there are no codewords with distance from y less than d, we conclude $(e-r) = 0$ and $\tilde{y} - r = y$. □

The syndrome decoding algorithm is far more efficient than checking all possible codewords. In most applications binary codes are used, and in this case the number of syndromes one needs to check is at most

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{\lfloor (d-1)/2 \rfloor}.$$

For non-binary codes, each term $\binom{n}{k}$ of the sum has to be multiplied by $(q-1)^k$.

3.3 Goppa codes

A Goppa code, also called an *algebraic geometric code*, in general, is a linear code constructed using an algebraic curve over a finite field $\mathbb{F}_q$. In post-quantum cryptosystems such as McEliece and Niederreiter, a particular subclass called *binary Goppa codes* is frequently used, by setting the field size as $q = 2$. We define Goppa codes in a simpler way which does not assume knowledge of algebraic geometry.

Let $\mathbb{F}_q$ be a finite field, which will be called the *symbol field*. By fixing a integer $m > 0$, we can regard $\mathbb{F}_q$ as a subfield of $\mathbb{F}_{q^m}$, which we call the *location field*. Let $g(x)$ be a polynomial with coefficients in the field $\mathbb{F}_{q^m}$, and let $L = \{L_1, \ldots, L_n\}$ be a subset of elements of $\mathbb{F}_{q^m}$ which are not roots of g. The size $n = |L|$ will be called the *length* of the code.

Definition 3.3.1. The **Goppa code** $\Gamma(g, L)$ is the set of all codewords $c = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$ that satisfy

$$\sum_{i=1}^{n} \frac{c_i}{x - L_i} \equiv 0 \pmod{g(x)}$$

Goppa codes are linear codes, and their redundancy $(n - k)$ is at most $\deg g$. Despite being defined by a rational polynomial, if we had to calculate the inverse $p^{-1}(x)$ of a polynomial modulo $g(x)$, we would need to solve the equation

$$h(x) \cdot g(x) + p^{-1}(x) \cdot p(x) = 1$$

for $h(x)$ and $p^{-1}(x)$. This can be done, e.g. using the extended Euclidean algorithm, which gives us a polynomial expression for the inverse, with degree less than $\deg g$.

A important fact of Goppa codes is that they have a lower bound on the minimum distance:

$$d \geq n - \deg g.$$

3.3.1 Decoding Goppa codes

One of the main advantages of Goppa codes is that they have efficient polynomial-time decoding algorithms. The standard textbook algorithm for this is the so-called *Patterson decoding* introduced in 1977 [9], which uses a syndrome polynomial. For simplicity, we suppose the code is binary, i.e. $q = 2$; a generalization for prime fields can be found in [1].

Definition 3.3.2. The **syndrome polynomial** of a Goppa code is the polynomial $s(x)$ with degree less than $\deg(g)$ such that

$$s(x) \equiv \sum_{i=1}^{n} \frac{c_i}{x - L_i} \mod g(x).$$

By design, $s(c) = 0$ for every codeword $c \in \Gamma(g, L)$. A natural consequence of this is that if the received vector is $\tilde{c} = c + e$, than $s(\tilde{c}) = s(e)$, so s act just like the syndrome in general linear codes.

To find the location of errors, we define the **error locator polynomial** as

$$\sigma(x) = \prod_{\substack{i=1 \\ e_i=1}}^{n} (x - L_i).$$

The name of this polynomial is justified, since the roots L_i indicate when errors occurred: $\sigma(L_i) = 0 \iff e_i = 1$. There's also a important variant of this polynomial, defined as

$$\eta(x) = \sum_{e_i=1} e_i \prod_{\substack{j \neq i \\ e_j=1}} (x - L_j).$$

Given the polynomials $\eta(x)$ and $\sigma(x)$, it is easy to find the error vector $e = (e_1, \ldots, e_n)$:

$$e_i = \begin{cases} 0 & \text{if } \sigma(L_i) \neq 0, \\ \frac{\eta(L_i)}{\sigma(L_i)} & \text{if } \sigma(L_i) = 0. \end{cases} \tag{3.2}$$

By simple algebra of polynomials, it can also be shown that

$$\eta(x) \equiv \sigma(x)s(x) \mod g(x). \tag{3.3}$$

Equation 3.3 is called **key equation** for decoding Goppa codes. This is because the task of the decoder is: given the polynomial $g(z)$ and the syndrome $s(x)$, find low-degree polynomials $\eta(x), \sigma(x)$ that solve the key equation. By doing this he can then find the error via 3.2 and decode the message.

Each polynomial in 3.3 can be reduced modulo $g(x)$, and by doing this we see that the key equation is nothing more than a linear system of equations with variables $1, x, \ldots, x^{\deg g - 1}$. It is not true that it always has a unique solution, but when it has, then we have a efficient algorithm for decoding the Goppa code. Berlekamp [2] proved non-singularity of the system in following two cases:

- when $\deg g = 2t$ and there are no more than t errors; and
- when the code is binary, g has no repeated irreducible factors, $\deg g = t$, and there are no more than t errors.

He also mentioned that when it is the case that $g(x) = x^n$, the code reduces to a primitive BCH code, for which there was already an efficient algorithm for solving the linear system. In 1975, Patterson [9] generalized this for arbitrary $g(x)$ by modifying Berlekamp's algorithm.

The algorithms are well described in the original paper from Patterson [9], but we note the important fact that they are polynomial-time, and thus very efficient.

4 Code-based cryptography

The use code-based schemes for cryptography is based on either the knowledge or strong belief that a underlying coding theory problem is hard. In general, this underlying problem is among the four shown below [11].

Problem 1 (General Decoding). *Given a code $C \subset \mathbb{F}_q^n$, an integer $t_0 > 0$ and a vector $v \in \mathbb{F}_q^n$, find a vector $x \in \mathbb{C}$ such that $d(x, v) \leq k$.*

Problem 2 (Syndrome Decoding). *Given a parity-check matrix H, a syndrome $s \in \mathbb{F}_q^k$, and an integer $t_0 > 0$, find a vector $x \in \mathbb{F}_q^n$ such that $w(x) = t_0$ and $\mathrm{Syn}_H(x) = s^\top$.*

Problem 3 (Goppa Parametrized Syndrome Decoding — GPSD). *Given a binary $2^m \times r$ parity-check matrix H and a syndrome s for a code C, decide if there exists a codeword $x \in C$ of weight r/m with $\mathrm{Syn}_H(x) = s$.*

Problems **1** and **2** were proven to be NP-complete for binary codes by Berlekamp in 1978 [3], and for q-ary codes by Alexander Barg in 1997. Problem **3** was proven to be NP-complete in 2005.

Problem 4 (Goppa Code Distinguishing — GD). *Given an $r \times n$ matrix H, decide if H is a parity-check matrix of a Goppa code.*

Problem **4** does not have a full NP-completeness proof, but there are restricted results for some particular cases.

The key in using Goppa codes is that they have efficient decoding algorithms, such as Patterson's algorithm. The idea is for Bob to keep one such algorithm as part of his private key, so only he can efficiently decode Alice's message.

4.1 McEliece

Let G be a generator matrix for a linear $[n, k]_q$-code (e.g. a Goppa code), and $m \in \mathbb{F}_q^k$ be a message Alice wants to transmit. Alice then encodes m as the vector $y \in \mathbb{F}_q^n$ given by

$$y = mG + e$$

for some small error vector e, and sends it to Bob. As we've seen, if $w(e) \leq \lfloor (d-1)/2 \rfloor$, then Bob could use syndrome decoding to recover the message.

But then the interceptor Eve could use the same algorithm to obtain m. She would only need to calculate the syndrome matrix H given G, which is not that hard. This poses a problem to our scheme. The McEliece cryptosystem tries to solve that by adding two layers of matrices to the encoding process: a **random non-singular matrix** S and a **permutation matrix** P.

The random non-singular matrix S is any $k \times k$ matrix with entries in $\mathbb{F}_q$ and which is non-singular ($\det S \neq 0$). This represents a random linear shuffling of the message space. Non-singularity is required for this shuffling to be a bijection.

The permutation matrix P, on the other hand, changes the order of entries in the codeword. It is a $n \times n$ matrix given by permutation of rows of the identity matrix. For example, a 4×4 permutation matrix could be

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

We can then construct the **encoding matrix** $\hat{G}$ for the McEliece cryptosystem by simple composition of these matrices: $\hat{G} \coloneqq S \cdot G \cdot P$. To encode a message m with the generator matrix $\hat{G}$ translates as: first, shuffle using S; then encode using G, and finally permute using P.

The McEliece cryptosystem is a asymmetric scheme. This means there are two keys: a *public key*, held by everyone (including the attacker), and used to encode a message; and a *private key*, held only by the receiver of the message. For this particular cryptosystem, the keys are given below.

The **public key** $(\hat{G}, t)$ is composed of:

- $\hat{G}$, the $k \times n$ generator matrix and
- $t \in \mathbb{Z}$, the error-correcting capacity (given by $t = \lfloor (d-1)/2 \rfloor$).

The **private key** (S, P, Dec^G) is composed of:

- S, the $k \times k$ random non-singular matrix,
- P, the $n \times n$ permutation matrix, and
- Dec^G, an efficient decoding algorithm for the code C with generator matrix G.

4.1.1 Encryption

The encryption algorithm defines a function $\mathrm{Enc}\colon \mathbb{F}_q^k \to \mathbb{F}_q^n$. Suppose Alice wants to encrypt a message $m \in \mathbb{F}_q^k$ to send to Bob. She has to do the following:

1. choose an error $e \in \mathbb{F}_q^n$ with $w(e) \leq t$,
2. calculate $\mathrm{Enc}(m) = m \cdot \hat{G} + e$.

This vector $y = \mathrm{Enc}(m) \in \mathbb{F}_q^n$ is then sent to Bob.

4.1.2 Decryption

Bob receives a vector $y \in \mathbb{F}_q^n$. He has the private key (S, P, Dec^G). He proceeds to multiply y by P^{-1} to the right, obtaining:

$$yP^{-1} = (mSGP + e)P^{-1} = mSG + eP^{-1}.$$

Since P is just a permutation, this new vector $\tilde{e} = eP^{-1}$ has the same weight $w(\tilde{e}) = w(e)$, and therefore can be removed by error correction without any worries. This is exactly what he

does next. He could use syndrome decoding for the code G, which would work; but since G is a Goppa code with an efficient decoding algorithm Dec^G, he uses this algorithm to obtain the vector $mS \in \mathbb{F}_q^k$:

$$\mathrm{Dec}^G(yP^{-1}) = \mathrm{Dec}^G\left((mS)G + eP^{-1}\right) = mS.$$

Since Bob knows the random matrix S, he finishes by applying S^{-1} to obtain the message m.

4.2 Niederreiter

The Niederreiter cryptoscheme is a variant of McEliece, with the significant tweak that the message is encoded as a syndrome, instead of as a codeword. We start with the parity-check matrix H of a linear $[n,k]_q$-code C, and, similarly to McEliece, we define an associated $(n-k)\times n$ parity-check matrix

$$\hat{H} = S \cdot H \cdot P,$$

where S is a $(n-k)\times(n-k)$ random non-singular matrix and P is a $n \times n$ permutation.

The **public key** $(\hat{H}, t)$ is composed of

- $\hat{H}$, the $(n-k)\times n$ parity-check matrix,
- t the error-correcting capacity,

and the **private key** (S, P, Dec^C) of

- S, the $(n-k)\times(n-k)$ random non-singular matrix,
- P, the $n \times n$ permutation matrix, and
- Dec^C, an efficient decoding algorithm for the code C with parity-check matrix H.

4.2.1 Encryption

The encryption function is $\mathrm{Enc}\colon B_t(\mathbb{F}_q^n) \to \mathbb{F}_q^{n-k}$, where the message domain $B_t(\mathbb{F}_q^n)$ is the set of all codewords in $\mathbb{F}_q^n$ with weight at most t. This is necessary to guarantee unique decodability.

Given a message $m \in \mathbb{F}_q^n$, Alice encrypts it by doing

$$y = \mathrm{Enc}(m) = \hat{H}m^\top = SHPm^\top.$$

Note that, differently from McEliece, no error vector has to be added to encrypt the message. Instead, the message is encoded in the place of error vector.

4.2.2 Decryption

Bob receives a vector $y \in \mathbb{F}_q^{n-k}$ and wants to decrypt it. First, he removes the scrambling matrix S:

$$S^{-1}y^\top = S^{-1}\left(SHPm^\top\right) = HPm^\top.$$

What he has now is a syndrome $S^{-1}y^\top = \mathrm{Syn}_H(Px^\top)$. By solving a linear system, he finds a vector $z \in \mathbb{F}_q^n$ such that $Hz^\top = S^{-1}y^\top$. This vector z is in the same space as the code C with parity-check matrix H. Therefore Bob can use the efficient algorithm Dec^C to find the codeword $c = z - P\tilde{m}^\top \in C$ closest to z, and the error $e = P\tilde{m}^\top$.

He finishes off by removing P and decoding the ciphertext:

$$P^{-1}\operatorname{Dec}^H(S^{-1}y) = \tilde{m}^\top.$$

Since we supposed m had weight at most t, if we take $t = \lfloor (d-1)/2 \rfloor$ then syndrome decoding works. By using the more efficient Patterson decoding shown in section 3.3.1, we listed some conditions over t such that decryption is correct (e.g. $t = \deg g$ if g has no repeated irreducible factors).

4.3 Equivalence

The previously shown cryptoschemes McEliece and Niederreiter are equivalent, in the sense that one can be converted into the other. Also, if one is vulnerable to some attack, the other is too (a fact that will simplify our description of attacks on section 4.4).

4.3.1 From McEliece to Niederreiter

On McEliece, the message m is encrypted as $y = m\hat{G} + e$ for some error e and $\hat{G} = SGP$. Since $\hat{G}$ is the generator matrix of some code $\hat{C}$, it has an associated parity-check matrix $\hat{H}$. This matrix $\hat{H}$ will taken as the parity-check matrix of the Niederreiter scheme, and it in fact has form $\hat{H} = S'HP$ for some S' (the reader will hopefully be convinced by the fact that $\hat{G}\hat{H}^\top = SGPP^\top H^\top S'^\top = SGH^\top S'^\top = S0S'^\top = 0$).

We then have that

$$y\hat{H}^\top = (m\hat{G} + e)\hat{H}^\top = e\hat{H}^\top.$$

On the associated Niederreiter scheme, the ciphertext is $y' = y\hat{H}^\top = e\hat{H}^\top$. Since e has weight $\leq t$, on this Niederreiter scheme it is possible to find the vector e such that $y' = e\hat{H}^\top$ (by decryption), and then $c - e = m\hat{G}$, which is easily decodable since it has no error.

4.3.2 From Niederreiter to McEliece

To go the other way around we start with a parity-check matrix $\hat{H} = SHP$ and define an associated generator matrix $\hat{G}$ for the same code. In Niederreiter, the message m is encrypted as $y = m\hat{H}^\top$. This y will be taken as the error of the associated McEliece Scheme

By solving a linear system, one can find a vector y' with weight $\geq t$ such that $y = y'H^\top$ and $y' = mG' + y$. This vector y' is the associated ciphertext. The message m is easily obtainable by using McEliece decryption.

This ends the proof of equivalence. Usually, Niederreiter is preferred over McEliece for two reasons:

1. the ciphertext is smaller ($n - k$ bits instead of k),
2. the ciphertext does not have a direct copy of the message, which could happen in McEliece when $\hat{G}$ is taken in the systematic form $\hat{G} = [I_{k\times k}|A]$.

4.4 Attacks

We describe here some of the most well-known attacks to both McEliece and Niederreiter.

4.4.1 Information set attack

For the next definition we introduce this notation: if M is an $k \times n$ matrix and $I \subset \{1, \ldots, n\}$ is a collection of indices, then M_I denotes the submatrix of M formed by taking only columns with indices in I.

Definition 4.4.1. Let C be an $[n, k]_q$-linear code with generator matrix G and parity-check matrix H. An **information set** I is a subset of $\{1, \ldots, n\}$ with $|I| = k$ such that one of the equivalent conditions hold:

1. the $k \times k$ matrix G_I is invertible;
2. the $(n-k) \times (n-k)$ matrix $H_{\{1,\ldots,n\}\setminus I}$ is invertible.

In 1962, Eugene Prange [10] introduced a technique for decoding linear codes called *information set decoding* — ISD. In the context of code-based cryptography, this technique is an attack to both McEliece and Niederreiter.

The idea is that if Eve, an attacker, knows (or guesses correctly) a information set, she can use the public encoding matrix ($\hat{G}$ in McEliece and $\hat{H}$ in Niederreiter) to decode the ciphertext. She does that by inverting $\hat{G}_I$ or $\hat{H}_I$ for a information set I.

In McEliece, a vector

$$y = m\hat{G} + e = (\langle m, \hat{G}_1 \rangle + e_1, \ldots, \langle m, \hat{G}_n \rangle + e_n)$$

is sent over the channel, where $\hat{G}_i$ are the columns of $\hat{G}$ and $e = (e_1, \ldots, e_n)$. In this scheme, e has no more than t errors (i.e., $\left|\{i : e_i \neq 0\}\right| \leq k$). So Eve could try to guess coordinates i for which $e_i = 0$, and they would satisfy $y_i = m\hat{G}_i$.

If she finds a collection $I = \{i_1, \ldots, i_k\}$ of k coordinates for which $e_{i_j} = 0 \ \forall j$, and if furthermore, this collection forms a information set for $\hat{G}$, then by inverting $\hat{G}$ Eve would be able to decrypt the message:

$$y = m\hat{G}_I \implies m = y\hat{G}_I^{-1}.$$

It would require at most $\binom{n}{k}/\binom{n-t}{k}$ guesses to find this information set, so adding the cost of inverting $\hat{G}$, which is k^3, it would lead to a work factor of

$$k^3 \left(1 - \frac{t}{n}\right)^{-k}$$

operations [11]. Since McEliece and Niederreiter are equivalent (as show in section 4.3.1), it is easy to translate this attack to Niederreiter.

4.4.2 Brute force attack

Syndrome decoding could be theoretically used to decrypt both McEliece and Niederreiter without having the private key. But since syndrome decoding relies on brute-force finding which

codeword generated such a syndrome, we'll see this becomes intractable on high dimensions.

Let's show this in Niederreiter. Eve is given a syndrome s and a parity-check matrix $\hat{H}$, and has the task to find a vector e with weight exactly t (usually the case for greater security). Let $\hat{H}_1, \ldots, \hat{H}_n$ be the columns of $\hat{H}$, and $e = (e_1, \ldots, e_n) \in \mathbb{F}_2^n$. Then

$$\mathrm{Syn}_H(e) = \hat{H}e^\top = e_1\hat{H}_1 + \cdots + e_h\hat{H}_n.$$

Since each $e_i \in \{0, 1\}$, finding a syndrome corresponds to summing t columns of $\hat{H}$. This means Eve has to calculate at most $\binom{n}{t}$ sums of t vectors. By appropriately changing the choice of columns so that each iteration changes one column, the number of operations can be reduced to $O\binom{n}{t}$ additions of one column.

For McEliece the attack works similarly. By putting $\hat{G}$ in the systematic form, it is easy to calculate a parity-check matrix $\hat{H}$, and the problem reduces to finding the error e that generated the syndrome $\hat{H}y^\top = \hat{H}e^\top$.

5 Closing comments

Code-based cryptography makes heavy use of concepts which are very natural to coding theory, such as the Hamming distance and syndrome decoding. For this reason, a brief study of this area is necessary to understand the applications to cryptography. For those interested in a more complete description of coding theory we refer to the classical text of Huffman and Pless [6].

The two cryptosystems we showed here are the 'textbook' versions, and for real life applications some tweaks and cautions are needed. In fact, we can talk about families of cryptosystems which are based on McEliece and Niederreiter. For descriptions of cryptoschemes which are more suitable to implementation, as far as the time this text is being written, we refer to the McEliece NIST submission [7] and the detailed text [11]. Nevertheless, the core ideas remain those described here.

Code-based cryptography is an area of great intersection: information theory, linear algebra, cryptography, complexity and quantum computing all have great importance in the development of the core ideas. With interdisciplinarity comes great richness of content, collaborations and rapid advances. The author hopes this text acts as a propellor of interests, so even more people can see the beauty and hopefully bring new ideas to the table.

Bibliography

[1] P. S. L. M. Barreto, R. Misoczki, and R. Lindner. "Decoding Square-Free Goppa Codes Over $\mathbb{F}_p$". In: *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6851–6858. DOI: 10.1109/TIT.2013.2270272.

[2] E. Berlekamp. "Goppa codes". In: *IEEE Transactions on Information Theory* 19.5 (1973), pp. 590–592. DOI: 10.1109/TIT.1973.1055088.

[3] E. Berlekamp, R. McEliece, and H. van Tilborg. "On the inherent intractability of certain coding problems (Corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386. DOI: 10.1109/TIT.1978.1055873.

[4] V. D. Goppa. "Algebraico-Geometric Codes". In: *Mathematics of the USSR-Izvestiya* 21 (1983), pp. 75–91. DOI: 10.1070/IM1983v021n01ABEH001641.

[5] V. D. Goppa. "Codes Associated with Divisors". In: *Problemy Peredachi Informatsii* 13 (1977). (in Russian), pp. 33–39. URL: http://mi.mathnet.ru/ppi1065.

[6] W. C. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003. DOI: 10.1017/CBO9780511807077.

[7] D. J. Bernstein et al. M. R. Albrecht. "Classic McEliece: conservative code-based cryptography". In: (2020). URL: https://classic.mceliece.org/nist/mceliece-20201010.pdf.

[8] M. Marcus. "White Paper on McEliece with Binary Goppa Codes". MA thesis. Technische Universiteit Eindhoven, 2019. URL: https://www.hyperelliptic.org/tanja/students/m_marcus/whitepaper.pdf.

[9] N. Patterson. "The algebraic decoding of Goppa codes". In: *IEEE Transactions on Information Theory* 21.2 (1975), pp. 203–207. DOI: 10.1109/TIT.1975.1055350.

[10] E. Prange. "The use of information sets in decoding cyclic codes". In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9. DOI: 10.1109/TIT.1962.1057777.

[11] H. Singh. *Code based Cryptography: Classic McEliece*. 2020. arXiv: 1907.12754 [cs.CR].

[12] D. R. Stinson and M. B. Paterson. *Cryptography: theory and practice*. 4th ed. Textbooks in Mathematics. CRC Press, 2018. ISBN: 978-1-1381-9701-5.