OVERVIEW OF THE MCELIECE CRYPTOSYSTEM AND ITS SECURITY

NOLIVE GNAN

1. Introduction

Error correcting codes are used to control errors in data over unreliable or noisy communication channels. We consider some of them, particularly Goppa codes and present their decoding algorithm. The McEliece public-key cryptosystem, a candidate for post-quantum cryptography uses binary Goppa codes for their efficient decoding algorithm. This enables the cryptosystem to have some advantages because the encryption and decryption is faster, and it is also thought to be secure against a quantum adversary. We will explain that, the cryptosystem nevertheless does not achieve IND-CCA2 and briefly discuss the suggestions to make the system IND-CCA2 secure in its submission to NIST.

2. Error-correcting codes

An error-correcting code is a code that can detect that an error has entered a message during its transmission. It ensures good and reliable communication. It is an encoding technique based on data redundancy. The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without retransmission. The material in this section may be found in [4] and [11].

Example 2.1. If we repeat the message three times, and only one error occurs, we can detect it and then correct it. Suppose we receive c = 001. The possible code words are 000 and 111. If the probability of errors occurring during transmission is relatively small, then we may infer that the message was most likely 000.

2.1. **Linear codes** [4].

Definition 2.2. A linear code C over the finite field $\mathbb{F}$ is a subspace C of $\mathbb{F}^n$. Therefore C is a subset of $\mathbb{F}^n := \mathbb{F} \times \mathbb{F} \times \cdots \times \mathbb{F}$ that is closed under addition and multiplication by scalars in $\mathbb{F}$. An element $c \in C$ is called a codeword. If dim $(C) = k$ then $0 \leq k \leq n$ and we say that C is a (n, k)-code over $\mathbb{F}$.

Remark 2.3. When $\mathbb{F} = \mathbb{Z}/2\mathbb{Z}$ is a binary field, we usually write 0111 to mean a vector $(0, 1, 1, 1) \in \mathbb{F}_2^4$.

Example 2.4. The $(3, 1)$-repetition code $\Gamma = \{000, 111\}$ is a subspace of $\mathbb{F}_2^3$. It is a linear code.

Date: April 2021.

This research was supervised by Monica Nevins, University of Ottawa, and supported by the grant "Quantum Security via Algebras and Representation Theory (Quasar)" grant of the Tri-Council New Frontiers in Research Fund.

2.2. **Encoding.** We saw that in general, for any codeword $c = (c_1, ..., c_n)$, each $c_i \in \mathbb{F}$ are its components. The encoding is then done by choosing an isomorphism G such that $G : \mathbb{F}^k \to C$. To encode a message $m \in \mathbb{F}^k$ we calculate and transmit $c = G(m)$.

The choices for this linear map are the bases of C. Each choice of basis gives a different linear map G. Then the matrix G of this map is the generator matrix of C. We choose the convention common in the coding literature to write this as a $k \times n$ matrix, so that $G(m) = mG$ where m is a row vector. The result is the row vector $c \in \mathbb{F}^n$. We will use row and column vectors interchangeably; which one is meant will be clear from context.

Example 2.5. A generator matrix for the $(3, 1)$-repetition code is

$$G_1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

2.3. **Decoding.** Suppose that Alice sent $y = G(m) \in C$ but that during the transmission y was converted into a vector $x \in \mathbb{F}^n \setminus C$. We are going to choose the $y' \in C$ most probable to be y, that is to say, which would have produced x with the least number of errors.

Definition 2.6. Let C be a linear (n, k)-code over $\mathbb{F}$. The dual code is the $(n, n-k)$-code given by $C^\perp := \{v \in \mathbb{F}^n | v \cdot w = 0 \forall w \in C\}$ where "$\cdot$" is the dot product.

Definition 2.7. Let C be an (n, k)-linear code and let $G^\perp$ be a matrix generating $C^\perp$. We set $H = G^\perp$ that we call a parity check matrix of C. We have that $C = ker(H)$.

Example 2.8. Let C be the $(3, 1)$-binary repetition code, with dual $C^\perp =< 101, 011 >$. So we can take

$$G^\perp = H = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Lemma 2.9. *Let C be a code with a parity check matrix H. Then $w \in C$ if and only if $Hw = 0$.*

Proof. Here w is viewed as a column vector. Since each entry of Hw is the dot product of a row of H with w, and the rows of H are in $C^\perp$, we see that if $w \in C$ then $Hw = 0$. Conversely, if $Hw = 0$ then w is orthogonal to each row of H, so $w \in (C^\perp)^\perp = C$. □

We can introduce errors by adding an error vector to the code word. If the codeword y has been transmitted, but the vector x has been received, then we define $z = x - y$ to be the error vector.

2.4. **Hamming Distance** [4]**.**

Definition 2.10. Let $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n) \in \mathbb{F}^n$. The Hamming distance from x to y is defined as: $d(x, y) := |\{i | x_i \neq y_i\}|$. For example $d(101101, 011100) = 3$.

Definition 2.11. The Hamming weight of a vector $x \in \mathbb{F}^n$ is $wt(x) = d(x, 0)$.

Definition 2.12. The minimum distance d of a linear code C is the minimum of the distances between distinct codewords. Formally, this can be described as $d = min\{d(x, y) | x \neq y \in C\}$.

Lemma 2.13. *If C is a linear code, then this is the same as the smallest Hamming weight of any nonzero codeword.*

Proof. If C is a linear code, then 0 is in C, and for any $x \in C, d(x, 0) = wt(x)$, and so the minimum weight is less or equal to the minimum distance. Conversely, if the minimum distance is attained for some $x, y \in C$, then $x - y \in C$ since C is linear so $wt(x - y) = d(x, y)$ and thus it is also the minimum weight. □

Theorem 2.14. *A code can correct r errors if and only if the minimum distance of the code is at least $2r + 1$.*

Proof. Suppose the minimum distance is at least $2r + 1$. Then if c is a codeword and e is an error of weight r, then $c + e$ is a distance of r from c and it must be distance at least $r + 1$ from any other codeword by the triangle inequality; therefore c is the unique closest codeword, so the error can be corrected.

Conversely, suppose the minimum distance is $\leq 2r$. Then take two codewords c_1 and c_2 that are this minimum distance apart. Let $x = c_1 - c_2$; then we can write x as a sum of two words of weight less than or equal to r, say $x = e + f$. Then $y := c_1 + f = c_2 - e$ is obtained from two different codewords by an error of weight $\leq 2r$, so the code cannot correct r errors. □

3. Syndrome decoding [4]

3.1. Definition syndrome decoding.

Syndrome decoding is a method of decoding a linear code over a noisy channel. It is minimum distance decoding using a table.

Definition 3.1. Let C be a (n, k) linear code with parity check matrix H. The *syndrome* of $x \in \mathbb{F}^n$ is the vector $Hx \in \mathbb{F}^{n-k}$.

Lemma 3.2. *If $x = c + e$ where $c \in C$ and e is an error vector, then*

$$Hx = He$$

Proof. For all codewords $c \in C, Hc = 0$. Thus, $Hx = H(c + e) = Hc + He = He$. □

If C can correct r errors, then the syndromes of all error vectors of weight less than or equal to r will be distinct. So we create a table: the first column is a list of all the error vectors of weight less than or equal to r; the second column is their syndromes. To decode x, we find the syndrome Hx in the second column and then $x - e = c$ is our corrected codeword.

Note that a (n, k) linear code over a finite field $\mathbb{F}_q$ has q^k code words. Each row corresponds to a syndrome which also corresponds to a specific coset.

3.2. Example.

Let C=⟨ 100011, 010101, 001110 ⟩ ; a generator matrix is

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

and the parity check matrix

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

We have $d(C)$=3, so we can correct any error with weight at most 1. This gives the table of syndromes:

e	He
000000	000
100000	011
010000	101
001000	110
000100	100
000010	010
000001	001

Say we receive the vector v= (111101), then we calculate Hv, after that we look in the table and try to find e such that e is the most likely error vector and thus $c = v - e$. Here, the table tells us that with e=(010000), $He = Hv$.

Therefore, $c = v - e$= 111101 - 010000=101101.

3.3. **Complexity.** If a codeword $x \in \mathbb{F}_q^n$ is sent and an error e occurs, $z = x + e$ is received. Therefore to perform the decoding, we have to look-up a precomputed table of size q^{n-k}, mapping Hz to z.

Under the assumption that no more than t errors were made during transmission, the receiver can look up the value He in a further reduced table of size $\sum_{i=0}^{t} \binom{n}{i}(q-1)^i \leq q^{n-k}$ which is the Hamming bound see [4, Theorem 1.7.8].

4. Goppa codes

All in this section is summarized from [7],[4] and [3].

4.1. **Polynomial rings over finite fields** [7],[4]**.**

Definition 4.1. Let $\mathbb{F}$ be a field. The set $\mathbb{F}[x] := \{\sum_{i=1}^{n} a_i x^i : a_i \in \mathbb{F}, 0 \leq i \leq n, \text{ and } n \geq 0\}$ with polynomial addition and multiplication forms a ring. It is called the polynomial ring over $\mathbb{F}$. The elements of $\mathbb{F}[x]$ are called polynomials over $\mathbb{F}$. The degree of a polynomial is the highest power of variable in it.

Definition 4.2. A polynomial $f(x)$ of positive degree is reducible over $\mathbb{F}$ if there exist two polynomials $g(x)$ and $h(x)$ over $\mathbb{F}$ such that $1 \leq \deg g(x), \deg h(x) \leq \deg f(x)$ and $f(x) = g(x)h(x)$. If there does not exist such polynomials in $\mathbb{F}[x]$, then $f(x)$ is irreducible over $\mathbb{F}$.

Example 4.3. The polynomial $f(x) = x^3 + x^5 \in \mathbb{Z}_3[x]$ is of degree 5 and reducible as $f(x) = x^3(1+x^2)$. The polynomial $g(x) = 1 + x + x^3 \in \mathbb{Z}_2[x]$ is of degree 3 and is an irreducible polynomial.

Theorem 4.4. *Let $f(x)$ be a polynomial over a field $\mathbb{F}$ with degree ≥ 1. Then the ring $\mathbb{F}[x]/\langle f(x)\rangle$ is a field if and only if $f(x)$ is an irreducible polynomial over $\mathbb{F}$.*

Example 4.5. The ring $\mathbb{Z}_2[x]/\langle 1 + x + x^2\rangle = \{0, 1, x, 1 + x\}$ is a field of order $2^2 = 4$. An element α in a finite field $\mathbb{F}_q$ is called a generator of $\mathbb{F}_q$ if $\mathbb{F}_q^\times = \{1, \alpha, \alpha^2, \cdots, \alpha^{q-2}\}$. Note that $\alpha^{q-1} = 1$.

Consider the field $\mathbb{F}_4 = \mathbb{F}_2[\alpha]$, where α is a root of the irreducible polynomial $1 + x + x^2 \in \mathbb{F}_2[x]$. Then we have $\alpha^2 = -(1+\alpha) = 1+\alpha; \alpha^3 = \alpha(\alpha^2) = \alpha(1+\alpha) = \alpha+\alpha^2 = \alpha+1+\alpha = 1$. From this, we see $\mathbb{F}_4 = \{0, \alpha, 1 + \alpha, 1\} = \{0, \alpha, \alpha^2, \alpha^3 = 1\}$, so α is a generator of $\mathbb{F}_4$.

For each power m of a prime q, there exists a unique finite field of size q^m, that we denote $\mathbb{F}_{q^m}$. It contains $\mathbb{F}_q$ as a subfield, so we have $\mathbb{F}_q \subseteq \mathbb{F}_{q^m}$ for all m.

4.2. **Goppa codes.** Let $g(z) \in \mathbb{F}[z]$ and $\alpha \in \mathbb{F}$ be such that $g(\alpha_i) \neq 0$. Then $(z - \alpha_i)$ is invertible mod $g(z)$ and so $\frac{1}{z-\alpha_i}$ can be viewed as a polynomial $p_i(z)$ modulo $g(z)$. It is defined as the unique polynomial $p_i(z)$ satisfying

$$p_i(z)(z - \alpha_i) \equiv 1 \mod g(z)$$

and which has degree less than t. Here we have used that since g has degree t, each element of $\mathbb{F}_{q^m}[z]/\langle g(z)\rangle$ can be represented by a unique polynomial of degree at most $t - 1$. Thus

$$\frac{1}{z - \alpha_i} \equiv p_i(z) = p_{i1} + p_{i2}z + \cdots + p_{it}z^{t-1} \mod g(z).$$

Example 4.6. Let $\mathbb{F} = \mathbb{F}_2$. Suppose $g(z) = z^3 + z^2 + 1$. We have $z^3 + z^2 + 1 = z^2(z - 1) + 1$ so

$$\frac{1}{z-1} \equiv z^2 \mod g(z)$$

also $z^3 + z^2 + 1 = z(z^2 + z) + 1$ so

$$\frac{1}{z} = z^2 + z \mod g(z).$$

Definition 4.7. Let $g(z)$ be a polynomial of degree $t \in \mathbb{F}_{q^m}[z]$, and let $T = \{\alpha_1, \alpha_2, \cdots, \alpha_n\} \subseteq \mathbb{F}_{q^m}$ such that, $g(\alpha_i) \neq 0$ for all $\alpha_i \in T$. Then the code denoted by $\Theta(T, g(z))$ and defined by

$$\{c = (c_1, c_2, \cdots, c_n) \in \mathbb{F}_q^n \, : \, \sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} \equiv 0 \mod g(z)\}$$

is called Goppa code.

4.3. **Parameters** [3]**.** The Goppa code $\Theta(T, g(z))$ of size n is a linear code over $\mathbb{F}_q$ with the properties:

- the dimension of the code $k \geq n - mt$.
- the minimal distance satisfies $d \geq t + 1$.

We will use the notation $[\,n, k, d]$ Goppa code for a Goppa code with parameters n, k and d with n the length of the codewords c.

4.4. **Syndrome decoding of Goppa codes.** Goppa codes are interesting because their decoding algorithm is more efficient than the one we presented in an earlier section. In the section below, we will show how this algorithm works.

4.4.1. *Parameters.* Let $y \in \mathbb{F}_q^n$ be a received word, containing r errors, with $2r + 1 \leq d$. Then

$$(\, y_1, \cdots, y_n) \, = (\, c_1, \cdots, c_n) + (\, e_1, \cdots, e_n)$$

where $c = (c_1, \ldots, c_n) \in C$ and $e_i \neq 0$ in exactly r places. We call $e = (e_1, \cdots, e_n)$ the error vector.

To correct the word, and find the right codeword c, we have to find the error vector e and discover :

- the set of error locations $B = \{i \mid e_i \neq 0\}$,
- the corresponding error values e_i for $i \in B$.

For that we have to compute the syndrome of our codeword and define two polynomials which would be essential in the error correction process.

4.4.2. *Polynomial description* [7]. Define:

- $\sigma(z) = \prod_{i \in B} (z - \alpha_i)$: the error locator polynomial. This polynomial has a degree r.
- $w(z) = \sum_{i \in B} e_i \prod_{j \in B, j \neq i} (z - \alpha_j)$: the error evaluator polynomial which is a degree-$r - 1$ polynomial.

Remark 4.8. $\prod_{j \in B, j \neq i}(z - \alpha_j)$ is the product of all $(z - \alpha_j)$ with $j \in \{1, ..., n\} \setminus \{i\}$. So if we evaluate this product at $z = \alpha_k$ with $k \neq i$,we will obtain 0 as the result. Let us name this product $f_i(z) = \prod_{j \in B, j \neq i}(z - \alpha_j)$. Then

$$f_i(\alpha_k) = \begin{cases} 0 & \text{if } k \neq i \\ \prod_{j \in B, j \neq i}(\alpha_i - \alpha_j) & \text{if } k = i. \end{cases}$$

Lemma 4.9. *If we obtain $\sigma(z)$ and $w(z)$ then we can deduce the positions of the errors and their values.*

Proof. 1⋄ From this definition, it is clear that the error locations follow directly from the roots of $\sigma(z)$ since $B = \{i \mid \alpha_i \text{ is a root of } \sigma(z)\}$.

2⋄ To find the error made we first have to calculate $\sigma'(z)$. Notice that

$$\sigma'(z) = \sum_{i\in B} \prod_{j\in B, j\neq i} (z-\alpha_j) = \sum_{i\in B} f_i(z).$$

Therefore it follows that for $i \in B$:

$w(\alpha_i) = \sum_{k\in B} e_k f_k(\alpha_i) = e_i \prod_{j\in B, j\neq i}(\alpha_k - \alpha_j)$ if k=i (see remark 4.8.)

$\sigma'(\alpha_i) = \sum_{i\in B} f_i(z) = \prod_{j\in B, j\neq i}(\alpha_i - \alpha_j)$.

This implies

$$\frac{w(\alpha_i)}{\sigma'(\alpha_i)} = \frac{\displaystyle\sum_{k\in B} e_k f_k(\alpha_i)}{\displaystyle\sum_{k\in B} f_k(\alpha_i)} = \frac{e_i \displaystyle\prod_{j\in B, j\neq i} (\alpha_i - \alpha_j)}{\displaystyle\prod_{j\in B, j\neq i} (\alpha_i - \alpha_j)} = e_i.$$

□

Example 4.10. Let's imagine that we already found our polynomials and now we want to find the error and its position. Suppose $e = (0, 0, 2, 0, 1)$ is the error vector; we should be finding at the end of this process and $B = \{3, 5\}$ then:

$$\begin{aligned} \sigma(z) &= (z-\alpha_3)(z-\alpha_5) \text{ and} \\ w(z) &= 2(z-\alpha_5) + (z-\alpha_3). \end{aligned}$$

The roots of $\sigma(z)$ are α_3 and α_5, so the errors are at position 3 and 5. Now we have to find the error made at each position.

Here, $\sigma'(z) = (z-\alpha_5) + (z-\alpha_3)$. This implies that at position 3, the error made was

$$\begin{aligned} \frac{w(\alpha_3)}{\sigma'(\alpha_3)} &= \frac{2(\alpha_3-\alpha_5) + (\alpha_3-\alpha_3)}{(\alpha_3-\alpha_5) + (\alpha_3-\alpha_3)} \\ &= \frac{2(\alpha_3-\alpha_5)}{(\alpha_3-\alpha_5)} \\ &= 2 \end{aligned}$$

We do the same calculation for α_5 and the result is $\frac{w(\alpha_5)}{\sigma'(\alpha_5)} = 1$. And we obtain $(0, 0, 2, 0, 1)$ which is the error vector that we were supposed to find.

4.4.3. *Syndrome of $y = (y_1, \dots, y_n)$.* Suppose that y is the received word, e the error vector and c the word sent. Then

$$\begin{aligned} s_y(z) &\equiv \sum_{i=1}^{n} \frac{y_i}{z-\alpha_i} \mod g(z) \\ &\equiv \sum_{i=1}^{n} \frac{c_i+e_i}{z-\alpha_i} \mod g(z) \\ &\equiv \sum_{i=1}^{n} \frac{c_i}{z-\alpha_i} + \sum_{i\in B} \frac{e_i}{z-\alpha_i} \mod g(z) \\ &\equiv \sum_{i\in B} \frac{e_i}{z-\alpha_i} \mod g(z);\ \text{since} \sum_{i=1}^{n} \frac{c_i}{z-\alpha_i} \equiv 0 \mod g(z) \end{aligned}$$

This polynomial is called the syndrome of y. We know that $w \in \Theta$ if and only if $s_y(w) = 0$. It is analogous to the syndrome we studied earlier, see Lemma 2.9.

Lemma 4.11. $\sigma(z)s_y(z) \equiv w(z) \mod g(z)$

Proof. So we already know that the error locator polynomial $\sigma(z) = \prod_{i\in B}(z-\alpha_i)$, the set of error locations is $B = \{i \mid e_i \neq 0\}$ and the evaluator polynomial $w(z) = \sum_{i\in B} e_i \prod_{j\in B, j\neq i}(z-\alpha_j)$.

Therefore, it follows that, modulo $g(z)$,

$$\begin{aligned} \sigma(z)s_y(z) &= \prod_{i\in B}(z-\alpha_i) \sum_{i\in B} \frac{e_i}{z-\alpha_i} \\ &= \frac{e_1 \prod_{i\in B}(z-\alpha_i)}{z-\alpha_1} + \dots + \frac{e_r \prod_{i\in B}(z-\alpha_i)}{z-\alpha_r}; B = \{1 \dots r\} \\ &= e_1 \prod_{i\in B, i\neq 1}(z-\alpha_i) + \dots + e_r \prod_{i\in B, i\neq r}(z-\alpha_i) \\ &= \sum_{i\in B} e_i \prod_{j\in B, j\neq i}(z-\alpha_j) \\ &= w(z) \end{aligned}$$

□

After computing the syndrome $s_y(z)$, and knowing that $\sigma(z)s_y(z) \equiv w(z) \mod g(z)$ we can find the values of $\sigma(z)$ and $w(z)$. Since we already know what the degree of our two polynomials are, we can solve the equation $\sigma(z)s_y(z) \equiv w(z) \mod g(z)$ by writing :

$$\sigma(z) = \sigma_0 + \sigma_1 z + \sigma_2 z^2 + \dots + z^r, \qquad w(z) = w_0 + w_1 z + w_2 z^2 + \dots + w_{r-1} z^{r-1}$$

and solve for the $2r$ unknowns with t equations, since $2r \leq d-1 \leq t$.

Example 4.12. The following example is from [3, Example 3]. Let $\mathbb{F}_{3^2}$ be the field corresponding to the primitive polynomial $x^2 - x - 1$ over the base field $\mathbb{F}_3$, where $\mathbb{F}_9 = \{a + b\alpha\} | a, b \in \{0, 1, -1\}$

and let α be one of its roots. We set $x = \alpha$ and we get $\alpha^2 = 1 + \alpha$. Then we have :

$$\begin{aligned} 0 &= 0 \\ 1 &= 1 \\ \alpha &= \alpha \\ \alpha^2 &= 1+\alpha \\ \alpha^3 &= 1-\alpha \\ \alpha^4 &= -1 \\ \alpha^5 &= -\alpha \\ \alpha^6 &= -1-\alpha \\ \alpha^7 &= -1+\alpha \end{aligned}$$

Consider $\Theta(T, g(z))$ such that $g(z) = z(z - \alpha^7) = z^2 + \alpha^3 z$ and $T = \{\alpha^i | 0 \le i \le 6\}$.

Suppose that the vector $y = (0,0,0,0,-1,-1,0) = (y_0, y_1, \ldots, y_6)$ is received having one error, so the degree of $\sigma(z)$ is 1 and the degree of $w(z)$ is 0. Our aim is to find the error vector.

(i) Syndrome. We compute

$$s_y(z) = \sum_{i=0}^{6} \frac{y_i}{z-\alpha^i} = -\frac{1}{z-\alpha^4} - \frac{1}{z-\alpha^5}$$

We know that if $ab \equiv 1 \mod g(z)$ then $b \equiv \frac{1}{a} \mod g(z)$, this is our we proceed to find our inverse. So here, our goal is find the two polynomials b_1 and b_2 such that: $-\frac{1}{z-\alpha^4} \equiv b_1 \mod g(z)$, $-\frac{1}{z-\alpha^5} \equiv b_2 \mod g(z)$. Using long division we find

$$z^2 + \alpha^3 z = (z-\alpha^4)(z-\alpha) - \alpha^5.$$

Multiplying both sides by $(-\alpha^5)^{-1} = \alpha^7$ gives

$$\alpha^7 g(z) = (z-\alpha^4)(\alpha^7 z - 1) + 1.$$

Therefore $-\frac{1}{z-\alpha^4} \equiv \alpha^7 z - 1 \mod g(z)$. Similarly, we compute

$$\begin{aligned} z^2 + \alpha^3 z &= (z-\alpha^5)(z-\alpha^2) + \alpha^7 \\ \alpha g(z) &= (z-\alpha^4)(\alpha z + \alpha^3) + 1 \end{aligned}$$

so $-\frac{1}{z-\alpha^5} \equiv \alpha z + \alpha^3 \mod g(z)$. This implies :

$$\begin{aligned} s_y(z) &= -\frac{1}{z-\alpha^4} - \frac{1}{z-\alpha^5} \\ &\equiv (\alpha z + \alpha^3) + (\alpha^7 z - 1) \mod g(z) \\ &\equiv z(\alpha + \alpha^7) + \alpha^3 - 1 \mod g(z) \\ &\equiv z(\alpha - 1 + \alpha) + 1 - \alpha - 1 \mod g(z) \\ &\equiv z(2\alpha - 1) - \alpha \mod g(z) \\ &\equiv \alpha^6 z - \alpha \mod g(z) \end{aligned}$$

(ii) Substituting $\sigma(z) = \sigma_0 + z$, we get:

$$\begin{aligned} \sigma(z) s_y(z) &= (\sigma_0 + z)(\alpha^6 z - \alpha) \\ &= -\sigma_0 \alpha + (\sigma_0 \alpha^6 - \alpha) z + \alpha^6 z^2 \\ &\equiv -\sigma_0 \alpha + (\sigma_0 \alpha^6 - 2\alpha) z \mod g(z) \\ &\equiv \sigma_0 \alpha + (\sigma_0 \alpha^6 + \alpha) z \mod g(z) \end{aligned}$$

Thus for $w(z) = w_0$, we get the system of equations by comparing coefficients of $\sigma(z)s_y(z) \equiv w(z) \mod g(z)$. Here, we compare the coefficients of $\sigma_0\alpha + (\sigma_0\alpha^6 + \alpha)z \equiv w_0 \mod g(z)$ and the system is: $\begin{cases} w_0 = -\sigma_0\alpha & (1) \\ 0 = \sigma_0\alpha^6 + \alpha & (2). \end{cases}$

From (2), we derive the value of $\sigma_0 = -\alpha^3$. Then, we substitute this value in equation (1) and we obtain $w_0 = -\alpha \times -\alpha^3 = \alpha^4$. Therefore, the unique solution to this system is $\sigma_0 = -\alpha^3$, $w_0 = \alpha^4$ hence $\sigma(z) = z - \alpha^3$ and $w(z) = \alpha^4$.

(iii) The root of $\sigma(z)$ is α^3 . The set of error locations is $B = \{i \mid \sigma(\alpha^i) = 0\} = \{3\}$.

(iv) The error value e_3 is

$$e_3 = \frac{w(\alpha^3)}{\sigma'(\alpha^3)} = \frac{\alpha^4}{1} = -1$$

(v) The codeword sent must have been

$$c = y - e = (0,0,0,0,-1,-1,0) - (0,0,0,-1,0,0,0) = (0,0,0,1,-1,-1,0).$$

5. McEleice

NIST in [1] has initiated a process to standardize quantum-resistant public-key cryptographic algorithms. The first code-based public-key cryptosystem was introduced in 1978 by Robert McEliece. The McEliece cryptosystem is based on binary Goppa codes. Code-based public-key cryptosystems like McEliece are built on codes for which there is an efficient decoding algorithm but such that an attacker will be forced to use an inefficient decoding algorithm such as syndrome decoding.

In the next sections taken from [7], we fix the parameters of Goppa code as n : length of the code, k : the dimension of code over the field $\mathbb{F}_q$ and t : the degree of the Goppa polynomial. In the classic McEliece, we work with binary Goppa codes and the values of n, k and t are public parameters. But the Goppa polynomial and the set T remain secret, they are private. Then, the public-key $\hat{\mathrm{G}}$ is computed as follows:

Firstly Alice has to generate a public and private key pair depending on the public parameters. (i) Alice selects a binary $[n, k]$-Goppa code, with its generator matrix of size $k \times n$ capable to correct up to t errors[1];

(ii) She then selects a random $k \times k$ non-singular binary matrix S (S is called the scrambler matrix) and a permutation matrix P of size $n \times n$;

(iii) She computes the $k \times n$ matrix $\hat{G} = S \times G \times P$ which is the public key[2] and keeps her private key (S, G, P).

So whenever Bob wants to send Alice a message, he has to encrypt it as follows:

(i) Bob has a binary message m of length k;

(iii) He generates a random n-bit error vector e with Hamming weight t;

(iv) Bob computes the ciphertext $c = m \times \hat{G} + e$ and sends it to Alice.

After recieving the ciphertext from Bob, Alice decrypts it this way:

(i) Alice computes P^{-1} using her private key;

(ii) She computes $c \times P^{-1} = m \times S \times G + e \times P^{-1}$;

[1]For binary Goppa codes, $w(z) = \sigma'(z)$ so the decoding algorithm can correct t errors, not just $t/2$ errors.

[2]In another formulation, instead of choosing S and P independently, she chooses the for $\hat{G}$ the systematic form of G, which is a particular example of SGP.

(iii) Finally, she uses the Patterson's decoding algorithm of Goppa codes in [7] to determine the value of m. This algorithm is a more efficient version of syndrome decoding algorithm presented earlier, but is specific to the binary case.

5.1. **Properties or efficiency.** The authors of [8] present several strong security arguments, among them, the fact that knowing only $\hat{\mathrm{G}}$, the attacker cannot distinguish this code from a random linear code, and since decoding random linear codes is inefficient (see section 3.3), decrypting the message will be computationally infeasible for large enough parameters. Also, they evoked the fact that, instances of McEliece's encryption scheme can be reduced to instances of the Syndrome Decoding problem, which is known to be NP-complete.

6. Attacks on McEliece

For the binary Goppa codes, no efficient structural attacks have been found and the existing attacks depend on an exhaustive search [8]. Among them, we can enumerate the ISD (Information Set Decoding Attack) in [7], the Gibson Attack [8] , and the Sidelnikov-Shestakov attack [9].

6.1. **IND-CCA2.** Ciphertext indistinguishability is a property of many encryption schemes [13]. When a cryptosystem is indistinguishable an adversary will be unable to distinguish pairs of ciphertexts based on the message they encrypt, that is, no adversary can distinguish the ciphertexts with probability greater than $\frac{1}{2} + \epsilon$ where ϵ is a negligible function in the security parameter k.

Definition 6.1. [10]

IND-CPA (indistinguishability under chosen plaintext attack)

In this game we suppose that the adversary who is Eve generates two messages of equal length. Alice, the challenger, decides randomly to encrypt one of them. Eve after recieving the ciphertext tries to guess which of the messages was encrypted.

IND-CCA1 (indistinguishability under non-adaptive chosen ciphertext attacks)

The target of the game is the same as in IND-CPA and it is basically the same process. Except that here, Eve has an additional capability: to call an encryption or decryption oracle. In other words Eve can encrypt or decrypt arbitrary messages before obtaining the challenge ciphertext from Alice. She will have a temporary access to the algorithms of Bob.

IND-CCA2 (indistinguishability under adaptive chosen ciphertext attacks)

In addition to its capabilities under IND-CCA1, Eve is now given access to the oracles after receiving the ciphertext from Alice, but cannot send the ciphertext to the decryption oracle. This means that she is able to encrypt and decrypt any ciphertext in order to help her find out which of the two messages Alice has chosen.

6.2. **McEliece vs Chosen-ciphertexts Attacks** [2]**.** In this section, we show that McEliece's system does not resist chosen-ciphertext attacks, it does not achieve "IND-CCA2 security." For instance, encryption of the same message twice produces two different ciphertexts which can be compared to find out the original message despite the fact that it is highly unlikely that errors were added in the same positions.

To see how the classic McEliece works under IND-CCA2 security, we have implemented a routine in a Sage code followed by the original implementation of McEliece cryptosystem found in [6]. Our implementation is based on the assumption that m_0 and m_1 are the chosen messages (by Eve), their encryptions are $enc_0 = m_0G + e_0$ and $enc_1 = m_1G + e_1$. Also, the ciphertext c that Alice sends Eve is $c = mG + e$.

So Eve can just compare the Hamming distance of c and her own encryptions of m_0 and m_1. We know that if Eve and Alice choose exactly the same message m, the Hamming distance of the ciphertext

c and the encryption of $m = mG + e'$ will be $d(c, enc) = d((mG + e), (mG + e')) = d(e, e')$. This distance is always between 0 and $2 \times wt(e)$:

- 0 when the encryptions of c and m are exactly the same, which means that the error vectors e, e' are the same,
- $2 \times wt(e)$ when the encryptions of c and m are completely different, which means that the error vectors e, e' are different,
- the distance is in between otherwise.

For example, if enc_0 is chosen by Alice, $mG = m_0G$: when we compute $d(c, enc_0) = d((mG + e), (m_0G+e_0)) = d(e, e_0)$. This implies that when Alice and Eve choose the same message, $d(e, e_0) \leq 2wt(e)$ (e of weight t, the maximum number of errors that the code can correct.

Here is our routine:

```
def main():
# We generate a field 3 times to see if Eve still has a chance to win when the field is big
for k in range(3):
    m = k+6;
    n = 2**m;
    t = floor((2+(2**m-1)/m)/2);
    F_2m = GF(n,'Z');
    PR_F_2m = PolynomialRing(F_2m,'X');


    #The algorithm prposed only used irreducible goppa codes because of their utility, see[5]
    while 1:
        irr_poly = PR_F_2m.random_element(t);
        if irr_poly.is_irreducible():
            break;

    crypto = McElieceCryptosystem(n,m,irr_poly);

    #Eve get two random messages and sends them to Alice encrypt it.
    m0=GetRandomMessage(crypto.goppa_code().generator_matrix().nrows());
    m1=GetRandomMessage(crypto.goppa_code().generator_matrix().nrows());
    print("m0",m0,"--")
    print("m1",m1,"\n")

    #Alice chooses randomly one of them to encrypt and sends it back to Eve
    r=randint(0,1)
    if(r==0):
        cypher=crypto.Encrypt(m0);
    else:
        cypher=crypto.Encrypt(m1);

    print("number of errors:",crypto.max_num_errors())

    #Eve now tries to guess which one of the original messages has been chosen
    #We have a for loop in which we generate some encryptions of our two messages and
    #we compare the Hamming distance of the ciphertext received from Alice and the two
    #we actually have
    #There is also an iteration to see how many times the Hamming distance is less or equal
```

```
        #than 2*wt(e), it should bigger for the right message since m*G is the same for the
        #ciphertext and the right message
        for i in range(20):
            enc=crypto.Encrypt(m0)
            enc1=crypto.Encrypt(m1)
            t0=0
            t1=0
            #print(dist(enc, cypher),"m0")
            #print(dist(enc1, cypher),"m1")
            if dist(enc, cypher)<=2*crypto.max_num_errors():
                t0+=1
            if dist(enc1, cypher)<=2*crypto.max_num_errors():
                t1+=1

        #We compare the results to help us choose
        if t0>t1:
            print("Eve finds m0",m0,"--","Alice has chosen",crypto.Decrypt(cypher),"\n")
        if t1>t0:
            print("Eve finds m1",m1,"--","Alice has chosen",crypto.Decrypt(cypher),"\n")

#function to calculate the Hamming distance of two vectors
def dist(enc1,enc2):
    i=0
    ham=0
    while(i<len(enc1[0])):
        if (enc1[0][i]!=enc2[0][i]):
            ham+=1
        i+=1
    return ham

#function to calculate the Hamming weight of a vector
def d(enc):
    i=0
    ham=0
    while(i<len(enc[0])):
        if (enc[0][i]!=0):
            ham+=1
        i+=1
    return ham
```

After running our code, it appeared that every time Eve wins even when the fiels gets bigger.

6.3. **Suggestions to make The McEliece cryptosystem CCA2-secure.** It is a difficult problem to provide IND-CCA2 security but there are many articles that evoke suggestions to make it secure. In some of them, they claimed that, it is possible to obtain CCA2-Security for the McEliece cryptosystem in the random Oracle model [1], [5]. Concerning, the submission to NIST [1], the goal was to create a KEM [3] with IND-CCA2 security in other words to send a random session key to use as the cipher key for a subsequent symmetric cryptosystem.

[3](Key Encapsulation mechanism is a protocol to transmit securely a session key to a party over an unsecured channel.)

7. Conclusion

Despite its vulnerability under some attacks, the McEliece Cryptosystem remains a good candidate to NIST's Post-Quantum Cryptography Standardization Project. This is because of all its benefits and advantages cited earlier. It is considered to be more efficient than some existing cryptographic systems due to its fast encryption and decryption scheme. It would interesting for us to do a follow up of this submission to see if the system is as strong as it is said to be.

References

[1] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, Wen Wang. NIST PQC Competition, 2020.
https://classic.mceliece.org/nist.html.

[2] Daniel J. Bernstein , Tanja Lange, and Christiane Peters. "Attacking and defending the McEliece cryptosystem", Department of Mathematics, Statistics, and Computer Science (M/C 249) University of Illinois at Chicago, Chicago, IL 60607–7045, USA, Department of Mathematics and Computer Science Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands,2008.
https://eprint.iacr.org/2008/318.pdf.

[3] E. Jochemsz. "Goppa Codes the McEliece Cryptosystem," Ph.D Thesis, Vrije Universiteit Amsterdam, 2002.

[4] Monica Nevins, Notes de cours, MAT3743: Algèbre linéaire appliquée. Avril 2020.

[5] Edoardo Persichetti. "On the CCA2 Security of McEliece in the Standard Model", Florida Atlantic University, 2012.
https://eprint.iacr.org/2012/268.pdf.

[6] Christopher Roering, " Coding Theory-Based Cryptopraphy: McEliece Cryptosystems in Sage", Honor Theses, College of Saint Benedict/Saint John's University, 2013.
https://digitalcommons.csbsju.edu/cgi/viewcontent.cgi?article=1019&context=honors_theses

[7] Harshdeep Singh, Scientific Analysis Group, Defence RD Organisation, Delhi – 110 054. "Code based Cryptography: Classic McEliece", 2019.
https://arxiv.org/pdf/1907.12754.pdf.

[8] Nicolas Sendrier. "On the Security of the McEliece Public-key Cryptosystem", Information, coding and mathematics, pages 141-164, Springer 2002.

[9] Filip Stojanovic, "A consideration of attacks and theory in code-based cryptography",Department of Mathematics and Statistics, University Of Ottawa, 2020.

[10] Crypto.stackexchange.com,Ciphertext indistinguishability notions. Last visit:April 17th.
https://crypto.stackexchange.com/questions/26689/easy-explanation-of-ind-security-notions.

[11] Wikipedia link. https://en.wikipedia.org/wiki/Error_correction_code, last visit: April 15th.

[12] Wikipedia link. https://en.wikipedia.org/wiki/Binary_Goppa_code, last visit: April 15th.

[13] Wikipedia link. https://en.wikipedia.org/wiki/Ciphertext_indistinguishability, last visit: April 15th.