GENERALIZED QUASI-BCH CODES AND APPLICATIONS IN CRYPTOGRAPHY

By Pauline Gonzalez

A Thesis

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

in Mathematics

Northern Arizona University

May 2020

ABSTRACT

GENERALIZED QUASI-BCH CODES AND APPLICATIONS IN CRYPTOGRAPHY

PAULINE GONZALEZ

Code based cryptography is a field of cryptography that has started to generate a considerable interest recently due to the emerging threat of quantum computation on the current cryptosystems. Even though the earliest example of a code-based cryptosystem was introduced at the same time as the RSA as the first examples of public key cryptosystems more than four decades ago, the McEliece cryptosystem was not chosen for practical applications due to its large key size. Recent variants of the McEliece cryptosystem have focused on a few improvements on the original cryptosystem, i.e., reducing the key size to make it computationally efficient and to increase the security by taking counter measures against certain attacks. A common theme in reducing the key size is using codes that have a compact description, such as cyclic codes, quasicyclic codes, etc. In this thesis, we consider a special family of codes that are named generalized quasi-BCH codes, which are codes that have a compact description, an efficient decoding algorithm and some additional security advantages that many of the current cryptosystems lack. The main advantage of our scheme is that we use blocks of different lengths, which makes it hard for the intruders to guess the structure of our code and also, it increases the error correction capability under certain circumstances. After an overview on codes and code based cryptography, we introduce generalized quasi-BCH codes and demonstrate how they can be used to design a new cryptosystem. We give several examples and make comparisons in terms of security and error-correction with the other schemes that use codes.

Acknowledgements

First of all, I would like to thank Dr. Bahattin Yildiz for his support through this master's. It was a pleasure working with him and learning so much about coding theory and research. I appreciate that he trusted me for almost two years with this project. Thank you to Dr. Sieben and Dr. Falk for agreeing being on my committee and giving me advice, making me a better student. I wanted to show my gratitude to the NAU math department for being so supportive during these two years and especially to Jeff Rushall for giving me my first research opportunity three years ago and bringing back the passion I had for mathematics.

Table of Contents

List of Tables

Chapter 1

Introduction

Coding theory emerged as a field to solve the engineering problem of transmitting digital information over a noisy channel so that the errors that occur during this transmission can be detected and corrected. The starting point of coding theory was the works of Shannon and Hamming in the late 40's, early 50's [13, 19, 11]. To communicate using codes, first we need to encode the information that will be transmitted. The second step is error correction, in other words, decoding. Lastly, we want to extract the original message. Therefore the main goal of this field of studies is to find ways to correct errors in transmissions. That is why coding theory has many applications in systems of communication such as data compression, cell phones, and several others. The main mathematical fields that we are using in coding theory are group theory, number theory, finite fields theory, and the algebra of polynomials.

Information theory has two distinct areas that we are going to look at, the theory of error correcting codes and cryptography. They both address different issues in communication. One application for the theory of error correcting codes is code based cryptography which gives us a link between these two distinct areas. We will describe the difference between traditional cryptography using asymmetric key encryption and code based cryptography.

In the traditional cryptography using asymmetric key encryption, the sender sends the plaintext to the receiver using the public key, or encryption key. Then the ciphertext can be decoded using the coupled private key, also called decryption key. To encode a codeword in Code Based Cryptography, the sender encodes the codeword and then modifies it before transmission, the receiver knows the secret to decode the message so decoding is easy.

The idea of public key cryptosystems was introduced in 1976. Two years later, two well-known cryptosystems were instituted, RSA and the McEliece cryptosystem (MECS), [16]. MECS is based on the algebraic theory of error correcting codes whereas RSA is based on asymmetric key encryption. When introduced, RSA became the favored cryptosystem between the two, mostly because of the large key size of MECS, even though the latter has faster encryption and decryption algorithms compared to more popular public key cryptosystems.

Recently, MECS has aroused interest since it is a candidate for post-quantum cryptography. Many of the public key cryptosystems that were more popular than MECS are based on the presumed difficulty of certain computational problems such as the discrete logarithm problem, the elliptic curve problem or the integer factorization problem in RSA. These problems, and others, can be solved efficiently using post-quantum computers. Therefore, these cryptosystems will not be secure anymore in the area of post-quantum computing while MECS is based on the fact that the decoding of a general linear code is computationally intractable, and so far, there has not been found an efficient quantum algorithm [23, 9, 6].

The McEliece cryptosystem is still a secure system if the parameters that we are using are large enough. It uses Goppa codes [12, 8], algebraically defined error correcting codes, which have an efficient decoding algorithm. We will now describe how the McEliece cryptosystem works. Let C be an $[n, k, 2t + 1]$ binary t-error correcting Goppa code of length n and dimension k with generator matrix $G_{k\times n}$. It is constructed via a randomly chosen irreducible polynomial of degree t over $GF(2^t)$. Let S be a random, dense $k \times k$ non-singular matrix, P be a random permutation matrix, and let $G' = SGP$. Note that G' generates a linear code that has the same parameters and error correction capability as C. This is how the McEliece cryptosystem works.

- Public Key: G' and t
- Private Key: G, S, P
- Plaintext Space: $\mathbf{m} \in \mathbb{F}_2^k$, the set of all k-bit vectors.
- Encryption: Encrypt the plain text $\mathbf{m}$ into $\mathbf{c} = \mathbf{m}G' + \mathbf{e}$, where $\mathbf{e} \in \mathbb{F}_2^n$ is a randomly chosen vector of weight at most t. i.e., $w_H(\mathbf{e}) \leq t$.

- Decryption:
  - Compute $\mathbf{c}' = \mathbf{c}P^{-1} = (\mathbf{m}S)G + \mathbf{e}P^{-1}$.
  - Decode $\mathbf{c}'$ with the fast decoding algorithm for C to obtain $\mathbf{m}'$.
  - Recover $\mathbf{m} = \mathbf{m}'S^{-1}$.

MECS is efficient mainly because of two reasons. First, there exists an efficient decoding algorithm which is not the case for arbitrary linear codes. The other thing is the structure of Goppa codes in MECS is hidden using random matrices to scramble the original generator matrix. If the structure is well hidden and the message sent is intercepted, then it won't be able to be decoded.

Recent research have been focusing on finding an alternative to MECS codes that would use a smaller key size without compromising the security of the code. Some of these variations have been proved to be vulnerable under certain types of attacks. The LDPC MECS was one of the first variants introduced, using LDPC codes to have an efficient decoding algorithm [17]. Later on, another variant was proposed using Quasi-Cyclic LDPC codes [4], this approach reduced the key size but was vulnerable to attacks that were focusing on finding low weight codewords.

In order to have an ideal code we would need an efficient decoding algorithm, a reduced key size and a resistance to the attacks that are known. In our research, we want to reach this goal by using a well-known family of codes, BCH codes, and generalize them in order to be able to hide the structure of the code having a preexisting efficient decoding algorithm.

Our goal is to find a type of codes that would still be relevant in a post-quantum computing context, which means the decoding algorithm would be secure against attacks via quantum computers. We have talked earlier how most of the cryptosystems known are using problems that are proven to be broken using post-quantum computers. We did not use any of the latter problems which makes us think that our cryptosystem could still be useful in a time where post-quantum computers will be in effect. To be able to construct such a code, we are going to start, in Chapter 2, with some background information on Coding theory and Code-Based Cryptography. We will give some necessary definitions and properties of codes and talk about some examples of codes with a well-known decoding algorithm. Then, in Chapter 3, we will look at a generalization of Quasi BCH codes and their decoding algorithm. In Chapter 5, we will design a code-based cryptosystem based on the Generalized Quasi-BCH codes described in Chapter 4. Finally we will conclude with some open questions and future research.

Chapter 2

Background

2.1 Coding Theory

First and foremost, we need to give introductory definitions.

Definition 2.1.1. Let $q = p^m$, where p is a prime number and $m \in \mathbb{N}$, and $GF(q) = \mathbb{F}_q$ the finite field of size q. A subset of $\mathbb{F}_q^n$ is called a *code* of length n over $\mathbb{F}_q$.

2.1.1 Linear Codes

When looking at codes, we are going to focus our attention on linear codes since their decoding algorithms are usually easier to find and to compute. Since we are exclusively going to work with linear codes, let us give a formal definition.

Definition 2.1.2. Let $\mathbb{F}_q$ be a finite field of order q. A *linear code* C over $\mathbb{F}_q$ of length n is a vector subspace of $\mathbb{F}_q^n$. An element of a code C is a *codeword*.

Another representation that we will use for a codeword c is its associated polynomial $c(x) = c_0 + c_1x + \ldots + c_{n-1}x^{n-1}$. This defines a vector space isomorphism which establishes a key link between coding theory and algebra. This notation is useful when studying cyclic codes and their multiple generalizations, which is a big part of algebraic coding theory. We will give some definitions to characterize codes.

Definition 2.1.3. When $q = 2$, we call the codes over $\mathbb{F}_q$ *binary codes*. When $q = 3$, they are called *ternary codes*. More generally, codes over $\mathbb{F}_q$ are called *q-ary codes*.

Definition 2.1.4. If C is a k-dimensional subspace of $\mathbb{F}_q^n$, then C is a $[n,k]_q$-*code*, where n is the length of C.

Definition 2.1.5. The *Hamming distance* of two codeword of C is the function

$$\begin{aligned} d : C \times C &\to \mathbb{N} \cup \{0\} \\ (u,v) &\mapsto |\{i : u_i \neq v_i\}|. \end{aligned}$$

We need the Hamming distance to define the minimum distance of a code C which is the minimum distance between each distinct pairs of codewords of C, we usually denote it $d(C)$ or just d.

Definition 2.1.6. Let C be a linear code of length n over $\mathbb{F}_q^n$, of dimension k and minimum distance d. Then we say that C is a $[n,k,d]_q$-*code*.

Now let us give a way to find the minimum distance of a code C.

Definition 2.1.7. The *Hamming weight* of a codeword u of C is defined as

$$w(u) :=\mid \{i \mid u_i \neq 0\} \mid.$$

It is the number of nonzero coordinates in u.

Definition 2.1.8. The *minimum Hamming weight* of C, denoted $w(C)$, is the minimum weight of all codewords of C, i.e. $w(C) = \min\{w(u), \mid u \in C, u \neq 0\}$.

Theorem 2.1.9. *([3]) Let C be a linear code over $\mathbb{F}_q$. Then*

$$d(C) = w(C)$$

We can now construct the Hamming weight enumerator.

Definition 2.1.10. The *Hamming weight enumerator* of a code C is defined by the following polynomial

$$\begin{aligned} W_C(y) &= \sum_{u \in C} y^{w(u)} \\ &= \sum_{i} A_i y^i \end{aligned}$$

where $A_i = \mid \{u \in C, w(u) = i\} \mid$.

Definition 2.1.11. The *minimum distance* of a code C is

$$d(C) := \min\{d(u,v) \mid u, v \in C\}.$$

It is usually denoted d or $d(C)$.

We note that the smallest nonzero power of y appearing in the Hamming weight enumerator is the minimum distance of the code C.

Definition 2.1.12. An $k \times n$ matrix whose rows form a basis for an $[n,k]$-linear code C is called a *generator matrix* for C, usually we denote it by G.

Definition 2.1.13. A *systematic matrix* is a generator matrix of the form $[I_k \mid A]$ where A is a $k \times (n-k)$ matrix over $\mathbb{F}_q$.

Then every linear code has a systematic matrix since we can find a generator matrix and take its row echelon form.

Definition 2.1.14. Let C be a linear code of length n over $\mathbb{F}_q$. The *dual code* $C \perp$ of the code C is the orthogonal complement of the code C as a vector subspace of $\mathbb{F}_q^n$.

Definition 2.1.15. A *Parity-Check matrix* for an $[n,k]$-linear code C is an $n-k \times n$ generator matrix for the dual code $C^\perp$ of the code C. We usually denote this matrix H.

If the generator matrix G of a code C can be written of the form $[I_k \mid A]$, where A is a $k \times (n-k)$ matrix, then the parity check matrix for the code C that corresponds to G can be written in the form $H = [-A^T \mid I_{n-k}]$.

Another way to determine the minimum distance, or a bound for the minimum distance of a code, is by using its parity-check matrix.

Theorem 2.1.16. *([3]) Let C be a linear code over $\mathbb{F}_q$ and let H be a parity-check matrix for C. Then*

- *C has minimum distance greater than or equal to d if and only if any $d-1$ columns of the matrix H are linearly independent,*
- *C has minimum distance less than or equal to d if and only if the matrix H has d columns that are linearly dependent.*

Now, the following corollary of this theorem gives us the distance of a linear code by combining both characterizations.

Corollary 2.1.17. *([3]) Let C be a linear code over $\mathbb{F}_q$ and H be a parity-check matrix for C. Then C has minimum distance d if and only if any $d-1$ columns of H are linearly independent and H has d columns that are linearly dependent.*

Some well-known families of codes have efficient decoding algorithm due to their particular algebraic structures. They include Hamming Codes, BCH codes, Constacyclic codes, Quasi-cyclic (QC) codes, Quasi-twisted (QT) codes, Reed-Solomon codes, Goppa codes. For example, an efficient decoding algorithm for BCH codes is the Berlekamp algorithm that uses the polynomial version of the extended Euclidean algorithm, and the algorithm is of polynomial complexity, $O(qn^3)$, where n is the degree of the polynomial in $\mathbb{F}_q$, [7]. This decoding algorithm for BCH codes was the most efficient for almost fifteen years. Based on this, Patterson constructed an efficient decoding algorithm for Goppa codes by building a variation of Berlekamp's algorithm [18]. We will talk more in details about some of these codes and their decoding algorithm in later sections.

2.1.2 Syndrome Decoding

When decoding a codeword received, a method to save time would be to calculate the syndromes of this codeword, which is finding in what coset it belongs.

Definition 2.1.18. Let C be an $[n,k]_q$-linear code over $\mathbb{F}_q$ and let H be a parity-check matrix for C. Then the *syndrome* of a codeword $w \in \mathbb{F}_q^n$ is defined as follows :

$$S_H(w) = wH^T \in \mathbb{F}_q^{n-k}.$$

Note that since the parity-check matrix of C is not unique, the syndrome depends on the one that we choose.

Let us give some properties of syndromes.

Theorem 2.1.19. *[3] Let C be an $[n,k]_q$-linear code over $\mathbb{F}_q$ and H be a parity-check matrix for C. Suppose $u, v \in \mathbb{F}_q^n$, then*

(a) $S_H(u+v) = S_H(u) + S_H(v)$,

(b) $S_H(u) = 0$ *if and only if u is a codeword in C,*

(c) $S_H(u) = S_H(v)$ *if and only if u and v are in the same coset of C.*

With the last property of the theorem, we notice that a coset can be identified using its syndromes, the syndrome of a coset is the syndrome of any of the words in the coset since they all have the same syndrome. Hence there is a one to one correspondence between cosets and syndromes.

2.1.3 Cyclic Codes

Looking at linear codes, we have seen that they can be described using their generator or parity-check matrix, we needed the Hamming weight to determine the minimum distance of the code. Researchers in Code-Based Cryptography started to investigate special types of linear codes in order to describe codes and encode and decode them in an easier way. In cyclic codes, the cyclic shift of a codeword is still in the code, this property leads to simpler encoding and decoding, we hence need an algebraic structure to create codes that have this property. That is why algebra has taken such a big part of Coding theory.

Definition 2.1.20. A code C of length n over $\mathbb{F}_q$ is *cyclic* if for all codeword $c = (c_0, c_1, \ldots, c_{n-1})$ in C, the left shift $(c_1, \ldots, c_{n-1}, c_0)$ is an element of C.

We want to have an algebraic structure on the code to be able to write the codewords of a code in the form of a polynomial. Let p be the function that sends $(c_0, c_1, \ldots, c_{n-1})$ to $c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$.
We want to connect ideals of $\mathbb{F}_q[x] / (x^n - 1)$ to cyclic codes.

Theorem 2.1.21. *([2]) Let C be a nonempty subset of $\mathbb{F}_q^n$. Then C is a cyclic code if and only if $p(C)$ is an ideal of ${}^{\mathbb{F}_q[x]}/_{(x^n-1)}$.*

Theorem 2.1.22. *([2]) Let I be a non zero ideal of ${}^{\mathbb{F}_q[x]}/_{(x^n-1)}$ and $g(x)$ be a nonzero monic polynomial in I of least degree in I. Then $g(x)$ is a generator for I, that is $(g(x)) = I$ and $g(x) \mid (x^n - 1)$.*

Now that we have these characterization of generator polynomials for ideals of ${}^{\mathbb{F}_q[x]}/_{(x^n-1)}$, we want to give a definition for generator polynomials of a cyclic code.

Definition 2.1.23. Let C be a cyclic code. Then the unique polynomial that generates the ideal $p(C)$ is called *generator polynomial* of C.

The following theorem gives us a one to one correspondence between cyclic codes in $\mathbb{F}_q^n$ and monic divisors of $x^n - 1$.

Theorem 2.1.24. *([2]) Each monic polynomial that divides $x^n - 1$ generates a cyclic code in $\mathbb{F}_q^n$.*

Theorem 2.1.25. *([2]) Let the factorization of $x^n - 1$ be $\displaystyle\prod_{i=1}^{r} p_i^{e_i}(x)$ where each p_i are distinct irreducible monic polynomials and $e_i \geq 1$. Then there are $\displaystyle\prod_{i=1}^{r}(e_i + 1)$ cyclic codes of length n over $\mathbb{F}_q$.*

Therefore if we know the factorization of $x^n - 1$, we know how many cyclic codes over $\mathbb{F}_q^n$ there are.

Theorem 2.1.26. *([2]) Let C be a cyclic code of length n over $\mathbb{F}_q$ and $g(x)$ be its generator polynomial. Then the dimension of the code C is k where $\deg(g(x)) = n - k$.*

Hence, to determine all the parameters of a cyclic code, we only need the generator polynomial of this code.

2.2 Different types of codes

2.2.1 BCH codes and their decoding algorithm

Binary BCH, which is an acronym for Bose-Chaudhuri-Hocquenghem, codes were invented in 1959 by Alexis Hocquenghem in [14] and generalized to BCH codes in 1960 by Raj Chandra Bose and Dwijendra Kumar Ray-Chaudhuri in [10].
This category of codes is a generalization of Hamming codes that can detect and decode multiple errors in a transmission whereas Hamming codes can only decode single errors. BCH codes are a class of cyclic codes, it is one of the best family of cyclic codes in terms of error-decoding. We will see in this section two different types of BCH codes, namely narrow sense, primitive BCH codes. We will also talk about one of the efficient decoding algorithms for BCH codes and the BCH bound.

Definitions

Definition 2.2.1. Let $\alpha \in \mathbb{F}_{q^m}$ be of order n, let be b and d two integers such that $0 \leq b < n$ and $2 \leq d \leq n$. A *BCH-code* is a cyclic code over $\mathbb{F}_q$ defined by the following $(d-1) \times n$ parity check matrix over $\mathbb{F}_{q^m}$:

$$H = \begin{bmatrix} 1 & \alpha^b & \alpha^{2b} & \cdots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{b+d-2} & \alpha^{2(b+d-2)} & \cdots & \alpha^{(n-1)(b+d-2)} \end{bmatrix}.$$

The first row of H consists of the first n consecutive powers of α^b.

Let us define specific types of BCH codes.

Definition 2.2.2. Let α be a primitive element of $\mathbb{F}_{q^m}$. Then the BCH code defined by α is called a *primitive BCH code*.

We can also define primitive BCH codes using a generator polynomial. Let α be a primitive root of $\mathbb{F}_{q^m}$. Let $m_i(x) \in \mathbb{F}_q[x]$ be the minimum polynomial of α^i. Then a primitive BCH code over $\mathbb{F}_q$ of length $n = q^m - 1$ is a code generated by the polynomial $g(x) := \operatorname{lcm}(m_b(x), m_{b+1}(x), \ldots, m_{b+\delta-2}(x))$, where $\delta \in \mathbb{N}$ is called the designed distance of the code and b is an integer.

If we have a primitive BCH code over $\mathbb{F}_{q^m}$, then the length of this BCH code is $n = q^m - 1$, which is, for $\mathbb{F}_{q^m}$, the maximum length possible.

Definition 2.2.3. A BCH code with $b = 1$ is called a *narrow-sense BCH code.*

Definition 2.2.4. If the minimum distance of the BCH code is $d = 2t + 1$ or $d = 2t + 2$, then we say that the code is *t-error-correcting.*

Let C be a t-error correcting primitive narrow-sense BCH code over $\mathbb{F}_q$ with decoder alphabet $\mathbb{F}_{q^m}$. Then a parity-check matrix H for C is

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{(n-1)} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{2t} & \alpha^{4t} & \cdots & \alpha^{2t(n-1)} \end{bmatrix}$$

where $n = q^m - 1$ is the blocklength of C.

BCH Bound

Let a be a nonzero element of $\mathbb{F}_q$ such that a does not have an n-th root in $\mathbb{F}_q$ and $\gcd(n, q) = 1$. Because q and n are relatively prime, $x^n - a$ does not have multiple roots. The roots of the polynomial $x^n - a$ are $\delta, \delta\zeta, \delta\zeta^2, \ldots, \delta\zeta^{n-1}$ where $\delta^n = a$ and ζ is a primitive n-root of unity. Let $m = \operatorname{ord}_q(n)$ be the multiplicative order of q modulo n. Then $\zeta \in \mathbb{F}_{q^m}$. Let r be the multiplicative order of a in $\mathbb{F}_q^*$, we can write $a = \alpha^i$ where α is a primitive element of $\mathbb{F}_q$ and i a positive integer. Then $r = \frac{q-1}{\gcd(i,q-1)}$ and $\delta^{n^r} = a^r = 1$. Therefore δ is an n^r-th primitive root of one, hence $\delta \in F_{q^s}$ where $s = \operatorname{ord}_q(nr)$. We have $q^s - 1 \equiv 0 \mod nr$, which implies that $q^s - 1 \equiv 0 \mod n$ and since $m = \operatorname{ord}_q(n)$, this implies that $m \mid s$, therefore $\mathbb{F}_{q^m} \subseteq \mathbb{F}_{q^s}$. Hence $\zeta, \delta \in \mathbb{F}_{q^s}$. Let ω be a primitive element of $\mathbb{F}_{q^s}$. Then there exists some integer t such that $\delta = \omega^t$ and $\zeta = \omega^{rt} = \delta^r$. We also have $q^s - 1 = ntr$.

Then we can write $x^n - a$ as

$$\begin{aligned} x^n - a &= \prod_{i=0}^{n-1}(x - \delta\zeta^i) \\ &= \prod_{i=0}^{n-1}(x - \delta^{1+ir}). \end{aligned}$$

For each i, $x - \delta^{1+ir}$ is an irreducible factor of $x^n - a$ that corresponds to a cyclotomic coset modulo nr. Since we know that $\delta, \delta\zeta, \delta\zeta^2, \ldots, \delta\zeta^{n-1}$ are all nr-th primitive root of unity, the polynomial $x^n - 1$ divides $x^{nr} - 1$ and $(x^{nr} - 1) \mid (x^{n(q-1)} - 1) \mid (x^{q^s-1} - 1)$. Hence $x^n - a$ divides $x^{q^s-1} - 1$.

Theorem 2.2.5. *[1] If the parity-check matrix of a BCH code has $\delta - 1$ rows, then we have $d_{min} \geq \delta$ where δ is the designed distance of the code.*

Theorem 2.2.6. *([15])*

(1) Let C be a BCH code over $\mathbb{F}_q$ of length $q^m - 1$ and generated by

$$g(x) := lcm(m_b(x), m_{b+1}(x), \ldots, m_{b+\delta-2}(x))$$

. Then the dimension of C is independent of the choice of the primitive element α.

(2) A BCH code over $\mathbb{F}_q$ of length $q^m - 1$ and designed distance δ has dimension at least $q^m - 1 - m(\delta - 1)$.

Proposition 2.2.7. *([15]) Let C be a narrow-sense q-ary BCH code of length $q^m - 1$ and designed distance δ. If $q \neq 2$ and $\gcd(q^m - 1, n) = 1$ for all $1 \leq n \leq \delta - 1$, then $\dim(C) = q^m - 1 - m(\delta - 1)$.*

Proposition 2.2.8. *([15]) Let C be a narrow-sense binary BCH code of length $n = 2^m - 1$ and designed distance $\delta = 2t + 1$. Then $\dim(C) \geq n - \frac{m(\delta-1)}{2}$.*

The designed distance of a BCH code and its minimum distance can be related by the next theorem.

Theorem 2.2.9. *([15]) If the designed distance of a code is δ, then its minimum distance is at least δ.*

Decoding BCH codes

Let C be a BCH code over $\mathbb{F}_q$ of length $n = q^m - 1$ Suppose that we have ν errors, where $\nu \leq t$ and that these ν errors are in locations $i_1, \ldots, i_\nu$. Let $Y_1, \ldots, Y_\nu$ be the error magnitudes.

Definition 2.2.10. Let α be a n-th root of $\mathbb{F}_{q^m}$. The *partial syndromes* S_j of a word y are defined as :

$$S_j := Y_1(\alpha^{i_1})^j + \ldots + Y_\nu(\alpha^{i_\nu})^j$$

for each $1 \leq j \leq 2t$.

Now let $X_j := \alpha^{i_j}$, we can write $S_i = \sum_{j=1}^{\nu} Y_j X_j^i$.

Example 2.2.11. Since

$$x^{15} - 1 = (1+x)(1+x+x^4)(1+x+x^2+x^3+x^4)(1+x^3+x^4)(1+x+x^2)$$

is the prime factorization over $\mathbb{F}_{16}$, we can construct $\mathbb{F}_{16}$ using a polynomial of degree 4. We are going to construct it using the polynomials $1+x+x^4$ and $1+x^3+x^4$.

Let C be the BCH code of length 15 generated by the polynomial

$$g(x) = 1+x+x^3+x^4+x^5+x^7+x^8 = (1+x+x^4)(1+x^3+x^4).$$

First of all, we want to know how many error C can decode.

If α be a root of $g(x)$ then we have two cases, α is a root of $1+x+x^4$ or α is a root of $1+x^3+x^4$.

Let α be such that $1+\alpha+\alpha^4 = 0$. Then

$$\mathbb{F}_{16} \cong \mathbb{F}_2[x]/\langle 1+x+x^4 \rangle = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{14}\}$$

i	α^i
0	1
1	α
2	α^2
3	α^3
4	$1+\alpha$
5	$\alpha+\alpha^2$
6	$\alpha^2+\alpha^3$
7	$1+\alpha+\alpha^3$
8	$1+\alpha^2$
9	$\alpha+\alpha^3$
10	$1+\alpha+\alpha^2$
11	$\alpha+\alpha^2+\alpha^3$
12	$1+\alpha+\alpha^2+\alpha^3$
13	$1+\alpha^2+\alpha^3$
14	$1+\alpha^3$

Table 2.1: Powers of the roots of $1+x+x^4$.

. We have that $\alpha^3, \alpha^6, \alpha^9$ and α^{12} are roots of $1+x+x^2+x^3+x^4$ and $\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$ are roots of the polynomial $1+x^3+x^4$. Then we have 4 consecutive roots, hence the designed distance is five which means that we can decode at most two error. Let us construct F_{16} using α in the table below, table 2.1 of the powers of α.

With the help of table 2.1, we can now determine the partial syndromes for a received codeword, $y(x) = 1+x^5+x^7+x^8+x^9$. We have

$$
\begin{aligned}
S_1(y) &= y(\alpha) = 1 + \alpha^5 + \alpha^7 + \alpha^8 + \alpha^9 \\
&= 1 + \alpha + \alpha^2 + 1 + \alpha + \alpha^3 + 1 + \alpha^2 + \alpha + \alpha^3 \\
&= 1 + \alpha, \\
S_2(y) &= y(\alpha^2) = 1 + \alpha^{10} + \alpha^{14} + \alpha^{16} + \alpha^{18} \\
&= 1 + 1 + \alpha + \alpha^2 + 1 + \alpha^3 + \alpha + \alpha^3 \\
&= 1 + \alpha^2, \\
S_3(y) &= 1 + \alpha^{15} + \alpha^{21} + \alpha^{24} + \alpha^{27} \\
&= 1 + 1 + \alpha^6 + \alpha^9 + \alpha^{12} \\
&= \alpha^2 + \alpha^3 + \alpha + \alpha^3 + 1 + \alpha + \alpha^2 + \alpha^3 \\
&= 1 + \alpha^3, \\
S_4(y) &= 1 + \alpha^{20} + \alpha^{28} + \alpha^{32} + \alpha^{36} \\
&= 1 + \alpha^5 + \alpha^{13} + \alpha^2 + \alpha^6 \\
&= 1 + \alpha + \alpha^2 + 1 + \alpha^2 + \alpha^3 + \alpha^2 + \alpha^2 + \alpha^3 \\
&= \alpha.
\end{aligned}
$$

Now let us construct the field using a root of the polynomial $1+x^3+x^4$, call it γ. We have that $\mathbb{F}_{16} = \mathbb{F}_2[x] / \langle 1 + x^3 + x^4 \rangle = \{0, 1, \gamma, \gamma^2, \ldots, \gamma^{14}\}$. We find the zeroes of the polynomials $1 + x + x^2 + x^3 + x^4$ and $1 + x + x^4$. The roots of $1 + x + x^4$ are $\gamma^7, \gamma^{11}, \gamma^{13}, \gamma^{14}$, the ones of $1 + x + x^2 + x^3 + x^4$ are γ^3, γ^{12}. We have 5 consecutive roots again, so $\delta = 5$ and $t = 2$.

And we construct $\mathbb{F}_{16}$ in the table of powers of γ below and calculate the syndromes.

i	γ^i
0	1
1	γ
2	γ^2
3	γ^3
4	$1+\gamma^3$
5	$1+\gamma+\gamma^3$
6	$1+\gamma+\gamma^2+\gamma^3$
7	$1+\gamma+\gamma^2$
8	$\gamma+\gamma^2+\gamma^3$
9	$1+\gamma^2$
10	$\gamma+\gamma^3$
11	$1+\gamma+\gamma^3$
12	$1+\gamma$
13	$\gamma+\gamma^2$
14	$\gamma^2+\gamma^3$

Table 2.2: Powers of the root of $1+x^3+x^4$.

Now we calculate the syndromes of y using table 2.2

$$\begin{aligned}
S_1(y) &= 1+\gamma^5+\gamma^7+\gamma^8+\gamma^9 \\
&= 1+1+\gamma+\gamma^3+1+\gamma+\gamma^2+\gamma+\gamma^2+\gamma^3+1+\gamma^2, \\
&= \gamma+\gamma^2 \\
S_2(y) &= 1+\gamma^{10}+\gamma^{14}+\gamma^{16}+\gamma^{18} \\
&= 1+\gamma^2+\gamma^3, \\
S_3(y) &= 1+1+\gamma^6+\gamma^9+\gamma^{12} \\
&= 1+\gamma^3, \\
S_4(y) &= 1+\gamma^5+\gamma^{13}+\gamma^2+\gamma^6 \\
&= 1+\gamma+\gamma^2.
\end{aligned}$$

Definition 2.2.12. The *error-locator polynomial* $\Lambda(x)$ is

$$\begin{aligned}\Lambda(x) &= \prod_{i=1}^{\nu}(1 - xX_i)\\ &= 1 + \Lambda_1 x + \ldots + \Lambda_\nu x^\nu.\end{aligned}$$

We notice that the roots of $\Lambda(x)$ are the inverses of the error locations. Define the following matrix, where ν is the number of errors in a received word y,

$$M_\nu = \begin{bmatrix} S_1 & S_2 & \cdots & S_\nu \\ S_2 & S_3 & \cdots & S_{\nu+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-1} \end{bmatrix}.$$

We will use this matrix to decode BCH codes. The following lemma gives us a characterization of M_ν.

Lemma 2.2.13. *[3] Let C be a t-error decoding BCH code and suppose that we have ν errors in a received word y.*

- *If $\nu \leq t$, then* $\det(M_\nu) \neq 0$*;*
- *If $\nu > t$, then* $\det(M_\nu) = 0$*.*

Decoding BCH codes is done in four steps. First of all we want to calculate the partial syndromes $S_i = y(\alpha^i)$, $i = b, b+1, \ldots, b+d-2$, of the word we receive.

The second step is finding the coefficients of $\Lambda(x)$ by solving the following linear system

$$M_\nu[\Lambda_\nu, \Lambda_{\nu-1}, \ldots, \Lambda_1]^T = [-S_{\nu+1}, -S_{\nu+2}, \ldots, -S_{2\nu}]^T$$

where ν is the largest integer less than or equal to t such that $\det(M_\nu) \neq 0$.

Next we find $X_1^{-1}, X_2^{-1}, \ldots, X_\nu^{-1}$, the zeroes of $\Lambda(x)$, which are also the inverses of the error locators $X_1 = \alpha^{i_1}, X_2 = \alpha^{i_2}, \ldots, X_\nu = \alpha^{i_\nu}$.

Our last step is to find the error magnitudes $Y_1, Y_2, \dots, Y_\nu$ by solving the following system of linear equations

$$\begin{aligned} Y_1X_1 + \cdots + Y_\nu X_\nu &= S_1 \\ Y_1X_1^2 + \cdots + Y_\nu X_\nu^2 &= S_2 \\ &\cdots \\ Y_1X_1^{2t} + \cdots + Y_\nu X_\nu^{2t} &= S_{2t}. \end{aligned}$$

In our research, We will work exclusively with $q = 2$, that is BCH binary codes. Therefore each Y_i will be equal to one, which means that the last step of the decoding algorithm is not necessary.

Example 2.2.14. Let us decode the BCH code $\langle 1 + x + x^2 \rangle$. The generator polynomial is $g(x) = 1+x+x^2$. We know that for this code we can decode up to one error. Let w be a root of $g(x)$ with $w \in \mathrm{GF}(2^3)$. Suppose that the received codeword is $y = (1, 0, 1)$ then $y(x) = 1+x^2$. We need to calculate the partial syndromes for y. We have $S_1 = y(w) = 1 + w^2 = w$ and $S_2 = y(w^2) = 1 + w$. Let X be the error and Y be the error magnitude. Then since the code is binary, $Y = 1$. Hence $S_1 = X$, $S_2 = X^2$. Now let us find the coefficient of the error locator polynomial $\Lambda(x) = 1+\Lambda_1 x$. Since X^{-1} is a zero of $\Lambda(x)$, we have $1+\Lambda_1 X^{-1} = 0$. By multiplying on both sides by X^2, we get $X^2 + \Lambda_1 X = 0$ which is $S_2 + \Lambda_1 S_1 = 0$. Therefore $\Lambda_1 = \frac{S_2}{S_1} = \frac{1+w}{w}$ and $\Lambda(x) = 1 + \frac{1+w}{w}x$. We note that $\Lambda(1 + w) = 0$, therefore the zero X^{-1} of $\Lambda(x)$. The error location is i where $X = w^i$, here $X^{-1} = w + 1$ so $X = w$. Hence the codeword sent is $(1, 1, 1)$.

2.2.2 Reed-Solomon Codes

Reed-Solomon codes are a subclass of BCH codes where the decoder alphabet is the channel alphabet. They were developed by Irvin S. Reed and Gustave Solomon in 1960 in [22] independently of the research of Bose Chaudhuri and Hocquenghem on BCH codes.

Definition 2.2.15. A *$q-$ ary Reed-Solomon code* is a BCH code of length $q - 1$ that is generated by the polynomial

$$g(x) = (x - \alpha^b)(x - \alpha^{b-1}) \cdots (x - \alpha^{b+\delta-2})$$

where α is a primitive element of $\mathbb{F}_q$ and $b \geq 0$, $2 \leq \delta \leq q-1$.

The minimum distance of the code is $\delta = 2t+1$ which is independent of the choice of α and b. Let us describe briefly a decoding algorithm, the Berlekamp-Massey algorithm, for Reed-Solomon codes. First, we are going to define $X_i = \alpha^{j_i}$ and Y_i to be respectively the error location and error magnitude, as we have already seen in the decoding of BCH codes in Section 2.2.1. The first step is to calculate the partial syndromes. We define two polynomials. The first one is the error locator polynomial, $\Lambda(x)$, that we defined earlier. The second polynomial is the error evaluator polynomial defined by

$$w(x) = \Lambda(x) + \sum_{i=1}^{s} x X_i Y_i \prod_{j=1, j \neq i}^{s} (1 - X_j x)$$

. Then we can find the roots of both polynomials.

2.2.3 Quasi-Cyclic Codes

Quasi-Cyclic codes are a generalization of cyclic codes. They were mostly studied to be applied in the variations of cryptosystems like the McEliece cryptosystem or Niederreiter's cryptosystem. The main aspect that was interesting when looking at Quasi-Cyclic codes was that it reduces considerably the key size which is an issue in both cryptosystems. But this idea was replaced by alternant codes since the decoding of random Quasi-Cyclic codes can be difficult.

In this section, let $n \in \mathbb{N}$ and let C be a code of length n over $\mathbb{F}_q$. Let $T : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be the map that transforms a codeword to its left cyclic shift, i.e, $T(c_0, c_1, \ldots, c_{n-1}) = (c_1, c_2, \ldots, c_{n-1}, c_0)$. We will write T^l to represent the l-left cyclic shift.

Definition 2.2.16. Suppose that l is a positive integer that divides n. An *l-Quasi-Cyclic code* over $\mathbb{F}_q$ of length n is a code of length n over $\mathbb{F}_q$ that is stable by T^l.

Let $l \in \mathbb{N}, \alpha \in \mathbb{F}_{q^l}$ such that $(1, \alpha, \ldots, \alpha^{l-1})$ forms an $\mathbb{F}_q$ base of the vector space $\mathbb{F}_{q^l}$.

Definition 2.2.17. The *folding* is the $\mathbb{F}_q$-linear map,

$$\begin{aligned} \Phi : \mathbb{F}_q^l &\to \mathbb{F}_{q^l} = \mathbb{F}_q[\alpha] \\ (a_0, a_1, \ldots, a_{l-1}) &\mapsto a_0 + a_1\alpha + \ldots + \alpha_{l-1}\alpha^{l-1}. \end{aligned}$$

Definition 2.2.18. The *unfolding* is the inverse $\mathbb{F}_q$-linear map of Φ,

$$\begin{aligned} \Phi^{-1} : \mathbb{F}_{q^l} &\to \mathbb{F}_q^l \\ a = a_0 + a_1\alpha + \ldots + a_{l-1}\alpha^{l-1} &\mapsto (a_0, a_1, \ldots, a_{l-1}) \end{aligned}$$

Now, using the maps defined above, we can create two types of codes. First, let $m \in \mathbb{N}$ and $f : X \to Y$ be a map between two sets. Then define the mapping $f^{\times m}$ as follows :

$$\begin{aligned} f^{\times m} : X^m &\to Y^m \\ (x_0, x_1, \ldots, x_{m-1}) &\mapsto (f(x_0), f(x_1), \ldots, f(x_{m-1})). \end{aligned}$$

Definition 2.2.19. If $n = ml$, then the *folded code* of C is defined by $\Phi^{\times m}(C)$.

Definition 2.2.20. Let C' be a code over $\mathbb{F}_{q^l}^m$. $(\Phi^{-1})^{\times m}(C')$ is called the *unfolded code* of C'.

Note that C is an l-Quasi-Cyclic code if and only if $\Phi^{\times m}(C)$ is cyclic.

2.2.4 Generalized Quasi-Cyclic Codes

Generalized Quasi-Cyclic codes were first introduced by Siap and Kulhan in [20]. We are later going to use these codes in order to define Generalized Quasi-BCH codes. Let us give the structure of generalized Quasi-cyclic codes. First, let q be a prime power and $m_1, \ldots, m_l$ be positive integers such that $(m_i, q) = 1$ for each i.

Define $R_i = {}^{\mathbb{F}_q[x]}\big/_{(x^{m_i} - 1)}$, then $R = R_1 \times \cdots \times R_l$ is an $\mathbb{F}_q[x]$ module, where the operations are component-wise addition and scalar multiplication.

Definition 2.2.21. We call an $\mathbb{F}_q[x]$-submodule of R a *Generalized Quasi-Cyclic Code* of length $(m_1, \ldots, m_l)$.

We notice that a generalized Quasi-Cyclic code of length m is a cyclic code, and if C has length $(m_1, \ldots, m_l)$ where all the m_i have the same value, then C is a Quasi-Cyclic code of length m.

The following Lemma will help us describe Generalized Quasi Cyclic Codes.

Lemma 2.2.22. *([20]) Let C be a Generalized Quasi-Cyclic code of length $(m_1, m_2, \ldots, m_l)$ and generated by $\{g'_1(x), g'_2(x), \ldots, g'_s(x)\}$ where $g'_i(x) = (g_{i1}, g_{i2}, \ldots, g_{il})$, for all $1 \leq i \leq s$. Then, for all i, j with $1 \leq i \leq s$, $1 \leq j \leq l$, there exists $f_{ij} \in {}^{\mathbb{F}_q[x]}\!/\!_{(x^{m_j}-1)}$ such that $g_{ij}(x) = f_{ij}(x)g_j(x)$ where $g_j(x) \in {}^{\mathbb{F}_q[x]}\!/\!_{(x^{m_j}-1)}$ and $g_j(x) \mid (x^{m_j} - 1)$.*

Corollary 2.2.23. *([20]) Let C be a 1-generator Generalized Quasi-Cyclic code. Then C is generated by*

$$\overline{f}(x) = (f_1(x)g_1(x), f_2(x)g_2(x), \ldots, f_l(x)g_l(x))$$

where $f_i(x), g_i(x) \in {}^{\mathbb{F}_q[x]}\!/\!_{(x^{m_i}-1)}$ and $g_i(x)$ divides $x^{m_i} - 1$ for each $1 \leq i \leq l$.

Theorem 2.2.24. *([20]) Let C be a 1-generator Generalized Quasi-Cyclic code with generator element $\overline{f}(x)$ as seen in Corollary 2.2.23. Let $h_i(x) = \frac{x^{m_i}-1}{g_i(x)}$ and $(f_i(x), g_i(x)) = 1$ for each $1 \leq i \leq l$. Then*

(i) $\dim(C) = \deg([h_1(x), \ldots, h_l(x), x^{m_1} - 1, \ldots, x^{m_l} - 1])$*, where $[h_1(x), \ldots, h_l(x), x^{m_1} - 1, \ldots, x^{m_l} - 1]$ is the lowest common multiple between the polynomials over $\mathbb{F}_q$.*

(ii) $d(C) \geq \min\limits_{i=1,\ldots,l}\{a_i + 1\}$, where a_i is the number of consecutive powers of the m_i-th root of unity that are zeroes of $g_i(x)$.

Let us give an example of a Generalized-Quasi-Cyclic code.

Example 2.2.25. Define $C := \langle 1 + x + x^2, 1 + x + x^3 \rangle$, C is an $\mathbb{F}_2[x]$-submodule of ${}^{\mathbb{F}_2[x]}\!/\!_{(x^3-1)} \times {}^{\mathbb{F}_2[x]}\!/\!_{(x^7-1)}$. C is a Generalized Quasi-Cyclic code of length $(m_1, m_2) = (3, 7)$ and $q = 2$.

The codes generated by $1 + x + x^2$ and $1 + x + x^3$ are BCH codes over $\mathbb{F}_2[x]/_{(x^3-1)}$ and $\mathbb{F}_2[x]/_{(x^7-1)}$ respectively. The check polynomials are $h_1(x) = 1 + x$ and $h_2(x) = (1+x)(1+x^2+x^3)$. The dimension of our code C is the degree of the lowest common multiple of the two polynomials $h_1(x)$ and $h_2(x)$ which is $1+x+x^2+x^4$. Therefore $\dim(C) = 4$. What we are interested in is the minimum distance of C, to find it we can compute the Hamming weight enumerator $W(y) = 1 + y^3 + 7y^4 + 7y^7$, the minimum distance is the smallest non zero exponent and therefore $d(C) = 3$.

2.2.5 Goppa Codes

Goppa Codes are a class of linear cyclic codes that were described in the 1970s. They are used in the McEliece cryptosystem and Niederreiter cryptosystem. Unlike most cyclic codes, there is an easy way to estimate the minimum distance, d, of a Goppa code since we have the result $d \geq \deg(g(x)) + 1$, where $g(x)$ is, as defined later in the next definition, the Goppa polynomial.

Definition 2.2.26. Let $g(x)$ be a polynomial over $\mathbb{F}_{q^m}$ where q is some prime power, for some positive integer m. Let $L = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ be a subset of $\mathbb{F}_{q^m}$ such that $g(\alpha_i) \neq 0$ for each i. The *Goppa code* $\Gamma(L, g)$ is

$$\Gamma(L,g) := \{c \in \mathbb{F}_q^n \mid R_c(x) \equiv 0 \mod g(x)\}$$

where $R_c(x) = \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i}$. We call $g(x)$ the *Goppa polynomial.*

Goppa is an interesting type of code because of its decoding algorithm. We are going to give some preliminary definitions and then we'll give a decoding algorithm for Goppa codes described in [21].

Let $\Gamma(L, g(x))$ be a Goppa code of dimension k in $\mathbb{F}_q$ and where $L = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$. Suppose we receive $y = (y_1, y_2, \dots, y_n)$ with r errors, where r is a positive integer such that $2r - 1 \leq d$, the minimal distance. Then we know that $y = (c_1, c_2, \dots, c_n) + (e_1, e_2, \dots, e_n)$ for some codeword $c = (c_1, c_2, \dots, c_n)$ and error vector $e = (e_1, e_2, \dots, e_n)$. Now define $B := \{i : 1 \leq i \leq n \text{ and } e_i \neq 0\}$, then B is the set of error location.

Definition 2.2.27. The *syndrome* of y is defined by

$$\begin{aligned} S(y) &:= \sum_{i=1}^{n} \frac{y_i}{x-\alpha_i} \\ &= \sum_{i=1}^{n} \frac{c_i}{x-\alpha_i} + \sum_{i\in B} \frac{e_i}{x-\alpha_i} \\ &= \sum_{i\in B} \frac{e_i}{x-\alpha_i} \pmod{g(x)}. \end{aligned}$$

In order to be able to find what the error vector e is, we need to define two polynomials.

Definition 2.2.28. The *error locator polynomial* is

$$\sigma(x) = \prod_{i\in B}(x-\alpha_i).$$

The *error evaluator polynomial* is defined by

$$\sum_{i\in B} e_i \prod_{\substack{j\in B\\ j\neq i}}(x-\alpha_i).$$

Then the decoding algorithm is the following. We first compute the syndrome $S(y)$ of y. The next step is to compute $\sigma(x)S(y) \equiv w(x) \pmod{g(x)}$. If we are working with a binary Goppa code, then $w(x)$ is the first derivative of $\sigma(x)$. Then we want to determine where the errors are located, in other words, determine the set B. Next, we find the value of e_i for each $i \in B$ by calculating $e_i = \frac{w(\alpha_i)}{\sigma'(\alpha_i)}$. If the code is binary then $e_i = 1$. We get the error vector $e = (e_1, e_2, \ldots, e_n)$ where $e_j \neq 0$ if $j \notin B$ and we can compute $c = y - e$.

Chapter 3

Generalized Quasi-BCH Codes and Their Decoding

In this chapter, we will combine the notion of a BCH code with Generalized Quasi-Cyclic codes. Generalized Quasi-BCH codes have been considered in the literature by M. Barbier, C. Chabot and G. Quintin in [5]. Their approach was based on looking at Quasi-BCH codes and generalizing them. In [5], they described a decoding algorithm that involved matrices with coefficients in $\mathbb{F}_q[x]/_{(x^n-1)}$. Our goal was to define Generalized Quasi-BCH by taking multiple BCH codes and define Generalized BCH-codes by using the properties of Generalized Quasi-Cyclic codes. We give a new definition of Generalized Quasi-BCH codes and an efficient decoding algorithm.

Definition 3.0.1. A code C is a *Generalized Quasi-BCH code* if it it generated by one or more BCH codes of different lengths, in the same way as generalized quasicyclic codes were defined above.

Example 3.0.2. Let us characterize the Generalized Quasi-BCH code

$$C := \left\langle 1+x+x^2, 1+x+x^3 \right\rangle.$$

C is an $\mathbb{F}_2[x]$ submodule of $\mathbb{F}_2[x]/_{x^3-1} \times \mathbb{F}_2[x]/_{x^7-1}$ We get the codewords by multiplying by a polynomial $p(x)$, the codeword would be

$$(p(x)(1+x+x^2),\, p(x)(1+x+x^3)).$$

Define $g_1(x) = 1 + x + x^2$ and $g_2(x) = 1 + x + x^3$. Then $h_1(x) = \frac{x^3-1}{g_1(x)} = 1 + x$ and $h_2(x) = \frac{x^7-1}{g_2(x)} = (1 + x)(1 + x^2 + x^3)$.
Let $p_1(x), p_2(x)$ be polynomials such that $\deg(p_1(x)), \deg(p_2(x)) < 4$ and
$p_1(x)g_2(x) = p_2(x)g_2(x)$, then we have $(p_1(x) - p_2(x))g_2(x) = 0$. The degree of $p_1(x) - p_2(x)$ is less than 4 and $\deg(g_2(x)) = 3$, therefore we know that the degree of $(p_1(x) - p_2(x))g_2(x)$ is less than 7. Then it can only be equal to zero if $p_1(x) = p_2(x)$. Therefore, each polynomial of degree less than 4 will give us a unique codeword.
Now suppose that $p(x)$ is a polynomial in $\langle 1 + x + x^3 \rangle$ such that $\deg(p(x)) \geq 4$ then we can find $q(x), r(x)$ such that $p(x) = h_2(x)q(x) + r(x)$ where $\deg(r(x)) < 4$ since we have $\deg(h_2(x)) = 4$. Then, by multiplying on both sides by $g_2(x)$, we get $p(x)g_2(x) = h_2(x)q(x)g_2(x) + r(x)g_2(x) = r(x)g_2(x)$ since $h_2(x)g_2(x) = 0$. We can see that when we multiply $g_2(x)$ by a polynomial of degree greater than or equal to 4, it can be expressed as the product of $g_2(x)$ and a polynomial of degree less than 4. Hence we can write

$$\left\langle 1 + x + x^3 \right\rangle = \{p(x)(1 + x + x^3) \mid \deg(p(x)) < 4\}.$$

Now looking at $\langle 1+x+x^2 \rangle$, multiplying by x we get $x(1+x+x^2) = x+x^2+x^3 = 1+x+x^2$, therefore multiplying by any power of x will give us the same codeword. Hence when we multiply $g_1(x)$ by some $p(x)$, only the constant will influence the new codeword we are getting. Therefore if $p(x) = 0 + a_1x + a_2x^2 + a_3x^3$, the codeword will be $(0,\ p(x)(1+x+x^3))$ and if $p(x) = 1 + a_1x + a_2x^2 + a_3x^3$, we will get $(1 + x + x^2,\ p(x)(1 + x + x^3))$. Now we can characterize our Generalized BCH code by giving its generating matrix

$$\begin{bmatrix} 1110100111 \\ 1110011101 \\ 0000101100 \\ 0001001110 \end{bmatrix}$$

Now let us decode this Generalized Quasi-BCH code. Let α be a root of the polynomial $1 + x + x^3$. The minimum distance of this code is $3 = 2t + 1$, therefore we can only decode up to one error. We want to send the codeword (0111001), we add one error and get $y = (0011001)$ which is $y(x) = x^2 + x^3 + x^6$. Now let us decode y to get the original codeword. We need to calculate the partial syndromes S_i. First, we use α to generate $\mathbb{F}_8$.

i	α^i
0	1
1	α
2	α^2
3	$\alpha + 1$
4	$\alpha^2 + \alpha$
5	$\alpha^2 + \alpha + 1$
6	$\alpha^2 + 1$

Table 3.1: Powers of the root of $1 + x + x^3$.

We have

$$\begin{aligned} S_1 &= y(\alpha) = \alpha^2 + \alpha^3 + \alpha^6 \\ &= \alpha^2 + \alpha + 1 + \alpha^2 + 1 \\ &= \alpha \\ S_2 &= y(\alpha^2) = \alpha^4 + \alpha^6 + \alpha^{12} \\ &= \alpha^2 + \alpha + \alpha^2 + 1 + \alpha^5 \\ &= \alpha + 1 + \alpha^2 + \alpha + 1 \\ &= \alpha^2. \end{aligned}$$

Since we can only have one error, the error locating polynomial is $\Lambda(x) = 1 - xX = 1 + \Lambda_1 x$. We know that X^{-1} is the zero of $\Lambda(x)$ so $1 - \Lambda_1 X^{-1} = 0$. We get the equation $X^2 - \Lambda_1 X = 0$ by multiplying by $X^2 = S_2$. Now we have $S_2 - \Lambda_1 S_1 = 0$, which is $\Lambda_1 = \frac{S_2}{S_1} = \frac{\alpha^2}{\alpha} = \alpha$. Therefore $\Lambda(x) = 1 + \alpha x$. The last step is to find the zero X^{-1} for this polynomial, $1 + \alpha X^{-1} = 0$ implies $\alpha X^{-1} = 1$, and so $X^{-1} = \alpha^6$. Taking the inverse, $X = \alpha$, therefore the sent code is (0111001).

In order to decode these Generalized Quasi-BCH codes, we are going to look at each block separately. Instead of taking the whole codeword that we need to decode, we take each corresponding block in the codeword and use the decoding algorithm described in Section 2.2.1 on each of these.

Chapter 4

Designing Code-Based Cryptosystems Using Generalized Quasi-BCH Codes

We are designing a cryptosystem the same way as McEliece. Usually, cyclic codes have blocks of the same length when generalized codes have blocks of different lengths. We are designing Generalized Quasi-BCH codes so that we have BCH codes with multiple blocks of different lengths. We encode words by transforming them into a tuple of polynomials that are each in a different block, which have potentially different lengths. Now knowing the length and the generator polynomial for the BCH code of each block, we can add an error vector by selecting how many errors will be in each of them since we know their error decoding capability. We send the modified word to the receiver for them to decode it. Decoding Generalized Quasi-BCH codes requires to use the decoding algorithm described in section 2.2.1. That is, to calculate, for each block, the syndromes of the received word to then find the coefficients of the error locating polynomial. Once we find these coefficients, we determine the zeros of the polynomial to finally find the positions of the errors. Therefore we can find the original polynomial for each block and hence the original codeword.

We will give two examples of Generalized Quasi-BCH codes to understand the structure of the cryptosystem.

Example 4.0.1. Let us give an example of a Generalized BCH code with three blocks, each of different lengths. Let $C = \langle 1+x+x^2, 1+x+x^3, 1+x^5+x^{10}\rangle$ which is an $\mathbb{F}_2[x]$-submodule of ${}^{\mathbb{F}_2[x]}\!/_{\langle x^3-1\rangle} \times {}^{\mathbb{F}_2[x]}\!/_{\langle x^7-1\rangle} \times {}^{\mathbb{F}_2[x]}\!/_{\langle x^15-1\rangle}$ we want to describe the code and use the decoding algorithm that we described. Since by the example above we know that the code generated by $1 + x + x^2$ will only be affected by the constant coefficient since multiplying by a power of x greater than or equal to one gives us the same polynomial. Let $g_1(x) = 1+x+x^2$, $g_2(x) = 1+x+x^3$, and $g_3(x) = 1+x^5+x^{10}$, define $p_1(x)$, $p_2(x)$ such that $\deg(p_1(x)), \deg(p_2(x)) < 5$ and $p_1(x)g_3(x) = p_2(x)g_3(x)$, then $(p_1(x)-p_2(x))g_3(x) = 0$. Since $\deg(g_3(x)) = 10$ and $\deg(p_1(x) - p_2(x)) < 5$, $\deg((p_1(x) - p_2(x))g_3(x)) < 15$ and can only be equal to zero if $p_1(x) = p_2(x)$. Now let $p(x)$ be a polynomial such that $\deg(p(x)) \geq 5$, by the division algorithm, there exist $q(x)$ and $r(x)$ such that $p(x) = h_3(x)q(x) + r(x)$ where $\deg(r(x)) < 5$. By multiplying on both sides by $g_3(x)$ we obtain $p(x)g_3(x) = h_3(x)q(x)g_3(x)+ r(x)g_3(x) = r(x)g_3(x)$. Therefore multiplying $g_3(x)$ by a polynomial of degree greater than or equal to five would result in multiplying by a polynomial of degree less than five. Hence we have $\langle x^{10} + x^5 + 1\rangle = \{p(x)(1 + x^5 + x^{10}) \mid \deg(p(x)) < 5\}$.

To decode this Generalized BCH code, we are going to apply the BCH algorithm described in section 2.2.1 to each block. Let us decode this generalized BCH code. First we find a word in our code C, we multiply our generator polynomials by a polynomial $p(x) = 1 + x^3 + x^4$, we get

$$\begin{aligned}(1 + x^3 + x^4)(1 + x + x^2) &= 1 + x + x^2\\ (1 + x^3 + x^4)(1 + x + x^3) &= x + x^5 + x^6\\ (1 + x^3 + x^4)(1 + x^5 + x^{10}) &= 1 + x^3 + x^4 + x^5 + x^8 + x^9 + x^{10} + x^{13} + x^{14}\end{aligned}$$

Combining all of them we get $(1+x^3+x^4)(1+x+x^2, 1+x+x^3, 1+x^5+x^{10}) = (1+x+x^2, x+x^5+x^6, 1+x^3+x^4+x^5+x^8+x^9+x^{10}+x^{13}+x^{14})$ which is coded by (1110100011100111001110011) where the first three coordinates represent the first block of the code, the next seven, the second block and the last fourteen, the third and last block. Since every block corresponds to its own BCH code and we saw in earlier examples their error decoding capability, let us add an error to each block. We receive the word $y = (1010110011100111000110011)$ which translate to

$$y(x) := (y_1(x), y_2(x), y_3(x)) = (1+x^2, x+x^2+x^5+x^6, 1+x^3+x^4+x^5+x^9+x^{10}+x^{13}+x^{14}).$$

We can see that we added errors in the second, sixth and nineteenth coordinates. Now let us use the decoding algorithm that we described in section 2.2.1 on each block to find where the errors are. Let α be a root of $1+x+x^2$, β be a root of $1+x+x^3$ and γ be a root of $1+x+x^4$. Let us calculate the syndromes of y using Table 4.3. We have

$$\begin{aligned}
S_1(y) = (S_{1,1}, S_{1,2}, S_{1,3}) = (y_1(\alpha), y_2(\beta), y_3(\gamma)) &= (1+\alpha^2, \beta+\beta^2+\beta^5+\beta^6, 1+\gamma^3+\gamma^5 \\
&\quad +\gamma^9+\gamma^{10}+\gamma^{13}+\gamma^{14}) \\
&= (\alpha, \beta^2, 1+\gamma^2), \\
S_2(y) = (y_1(\alpha^2), y_2(\beta^2), y_3(\gamma^2)) &= (1+\alpha^4, \beta^2+\beta^4+\beta^{10}+\beta^{12}, 1+\gamma^6+\gamma^{10} \\
&\quad +\gamma^{18}+\gamma^{20}+\gamma^{26}+\gamma^{28}), \\
&= (1+\alpha, \beta^2+\beta^4+\beta^3+\beta^5, 1+\gamma^6+\gamma^{10} \\
&\quad +\gamma^3+\gamma^5+\gamma^{11}+\gamma^{13}) \\
&= (1+\alpha, \beta+\beta^2, \gamma).
\end{aligned}$$

Let us find the error locating polynomial. Since we can only decode one error for each block, the error locating polynomial is $\Lambda(x) = (\Lambda_1(x), \Lambda_2(x), \Lambda_3(x)) = (1+\Lambda_1 x, 1+\Lambda_2 x, 1+\Lambda_3 x)$. Now we want to find the coefficients Λ_i for each $i = 1,2,3$. Since $X^{-1} = (X_1^{-1}, X_2^{-1}, X_3^{-1})$ is the zero of $\Lambda(x)$, $(1+\Lambda_1 X_1^{-1}, 1+\Lambda_2 X_2^{-1}, 1+\Lambda_3 X_3^{-1}) = (0,0,0)$. Now by multiplying by $S_2 = X^2$ on both sides we get $(S_{2,1} - \Lambda_1 S_{1,1}, S_{2,2} - \Lambda_2 S_{1,2}, S_{2,3} - \Lambda_3 S_{1,3}) = (0,0,0)$, which means that

$$\begin{aligned}
\Lambda_1 &= \frac{S_{2,1}}{S_{1,1}} = \frac{1+\alpha}{\alpha} = \alpha, \\
\Lambda_2 &= \frac{S_{2,2}}{S_{1,2}} = \frac{\beta+\beta^2}{\beta^2} = \beta^2, \\
\Lambda_3 &= \frac{S_{2,3}}{S_{1,3}} = \frac{\gamma}{1+\gamma^2} = \gamma^8.
\end{aligned}$$

Therefore the error locating polynomial is $\Lambda(x) = (1+\alpha x, 1+\beta^2 x, 1+\gamma^8 x)$. Thus $X_1 = \alpha, X_2 = \beta^2, X_3 = \gamma^8$. Hence the locations of the errors, which are the powers plus one of α, β and γ in X_1, X_2 and X_3 respectively, are in the second coordinate for the first, the third coordinate for second block and in the ninth one for the last block.

Example 4.0.2. Let us look at the Generalized BCH code $C = \langle 1+x+x^2+x^5+x^9+x^{11}+x^{13}+x^{14}+x^{15}+x^{16}+x^{18}+x^{19}+x^{21}+x^{24}+x^{25}, 1+x+x^6+x^9+x^{10}+x^{12}+x^{16}+x^{17}+x^{18}+x^{22}+x^{24}+x^{26}+x^{29}+x^{34}+x^{39}+x^{40}+x^{42}+x^{43}+x^{44}+x^{46}+x^{48}+x^{49}+x^{50}\rangle$ of length 94 and dimension 24. We find a codeword in C by multiplying our generators polynomial by a random polynomial, we pick the polynomial $p(x) = 1+x^2+x^5+x^8+x^{10}+x^{15}$. We get

$$\begin{aligned} p(x)g_1(x) &= 1+x^3+x^5+x^6+x^9+x^{10}+x^{11}+x^{12}+x^{13}+x^{17}+x^{18}+x^{20}+x^{21} \\ &\quad +x^{22}+^{24}+x^{26}, \\ p(x)g_2(x) &= x^3+x^5+x^{11}+x^{14}+x^{17}+x^{18}+x^{20}+x^{22}+x^{23}+x^{27}+x^{30}+x^{31} \\ &\quad +x^{32}+x^{33}+x^{40}+x^{42}+x^{43}+x^{44}+x^{51}+x^{52}+x^{58}+x^{60}+x^{61}. \end{aligned}$$

We know we can correct seven errors in the first block of the code, and fifteen in the second. Say the word with errors that we receive is $y(x) = (y_1(x), y_2(x))$ where

$$\begin{aligned} y_1(x) =& 1+x+x^3+x^6+x^{10}+x^{11}+x^{12}+x^{13}+x^{15}+x^{17}+x^{18}+x^{21}+x^{22}+x^{24}+x^{26} \\ &+x^{27}+x^{29} \\ y_2(x) =& 1+x+x^3+x^5+x^7+x^8+x^{11}+x^{12}+x^{14}+x^{15}+x^{17}+x^{18}+x^{19}+x^{20}+x^{22}+x^{23} \\ &+x^{24}+x^{27}+x^{31}+x^{32}+x^{33}+x^{35}+x^{39}+x^{40}+x^{42}+x^{44}+x^{52}+x^{55}+x^{60}+x^{61} \end{aligned}$$

Let α and β be roots of $1+x^2+x^5$ and $1+x+x^3+x^4+x^6$, respectively. We use α and β to construct $\mathbb{F}_{32}$ and $\mathbb{F}_{64}$ respectively. The tables showing the constructions can be found at the end of this chapter. Now we calculate the syndromes of y.

$$
\begin{aligned}
S_1(y) &= (\alpha^2+\alpha^3, \beta^2+\beta^5) = (\alpha^{20}, \beta^{15})\\
S_2(y) &= (\alpha+\alpha^3+\alpha^4, 1+\beta^5) = (\alpha^{9}, \beta^{30})\\
S_3(y) &= (1+\alpha+\alpha^2+\alpha^4, \beta+\beta^2+\beta^5) = (\alpha^{26}, \beta^{17})\\
S_4(y) &= (1+\alpha, \beta^4+\beta^5) = (\alpha^{18}, \beta^{60})\\
S_5(y) &= (\alpha+\alpha^2+\alpha^4, 1+\beta+\beta^2+\beta^3) = (\alpha^{28}, \beta^{42})\\
S_6(y) &= (\alpha^3+\alpha^4, 1+\beta^2+\beta^5) = (\alpha^{21}, \beta^{34})\\
S_7(y) &= (\alpha^3, 1+\beta^2) = (\alpha^{3}, \beta^{49})\\
S_8(y) &= (1+\alpha^2, \beta+\beta^2) = (\alpha^{5}, \beta^{57})\\
S_9(y) &= (\alpha^2+\alpha^4, 1) = (\alpha^{7}, \beta^{0})\\
S_{10}(y) &= (1+\alpha^3+\alpha^4, \beta+\beta^2+\beta^3) = (\alpha^{25}, \beta^{21})\\
S_{11}(y) &= (1+\alpha+\alpha^4, \beta^2) = (\alpha^{17}, \beta^{2})\\
S_{12}(y) &= (1+\alpha+\alpha^2, \beta^5) = (\alpha^{11}, \beta^{5})\\
S_{13}(y) &= (\alpha+\alpha^3, \beta+\beta^2+\beta^3+\beta^4+\beta^5) = (\alpha^{6}, \beta^{38})\\
S_{14}(y) &= (\alpha+\alpha^3, 0) = (\alpha^{6}, 0)\\
S_{15}(y) &= (\alpha+\alpha^2+\alpha^4, \beta+\beta^5) = (\alpha^{28}, \beta^{36})\\
S_{16}(y) &= (1+\alpha^4, \beta^2+\beta^4) = (\alpha^{10}, \beta^{51})\\
S_{17}(y) &= (\alpha^2+\alpha^3+\alpha^4, 1+\beta+\beta^2+\beta^3) = (\alpha^{13}, \beta^{42})\\
S_{18}(y) &= (1+\alpha^2+\alpha^3+\alpha^4, 1) = (\alpha^{14}, \beta^{0})\\
S_{19}(y) &= (1+\alpha+\alpha^4, \beta+\beta^3+\beta^5) = (\alpha^{17}, \beta^{41})\\
S_{20}(y) &= (\alpha+\alpha^2, 1+\beta+\beta^2+\beta^3) = (\alpha^{19}, \beta^{42})\\
S_{21}(y) &= (\alpha+\alpha^2+\alpha^3+\alpha^4, 1+\beta+\beta^2+\beta^3) = (\alpha^{24}, \beta^{42})\\
S_{22}(y) &= (\alpha^3, \beta^4) = (\alpha^{3}, \beta^{4})\\
S_{23}(y) &= (1+\alpha^2+\alpha^3+\alpha^4, 1+\beta^3+\beta^4+\beta^5) = (\alpha^{14}, \beta^{11})\\
S_{24}(y) &= (1+\alpha^2+\alpha^4, 1+\beta^4+\beta^5) = (\alpha^{22}, \beta^{10})
\end{aligned}
$$

$$S_{25}(y) = (\alpha + \alpha^2 + \alpha^3 + \alpha^4, 1 + \beta + \beta^4) = (\alpha^{24}, \beta^{16})$$
$$S_{26}(y) = (\alpha + \alpha^2 + \alpha^3, 1 + \beta^3) = (\alpha^{12}, \beta^{13})$$
$$S_{27}(y) = (\alpha^2 + \alpha^4, 1 + \beta + \beta^2 + \beta^4) = (\alpha^{7}, \beta^{18})$$
$$S_{28}(y) = (\alpha + \alpha^2 + \alpha^3, 0) = (\alpha^{12}, 0)$$
$$S_{29}(y) = (\alpha + \alpha^2, \beta^2 + \beta^3 + \beta^4 + \beta^5) = (\alpha^{19}, \beta^{44})$$
$$S_{30}(y) = (1 + \alpha^3 + \alpha^4, 1 + \beta^2 + \beta^4 + \beta^5) = (\alpha^{25}, \beta^{9}).$$

The error locating polynomial is $\Lambda(x) = 1 + \Lambda_1 x + \Lambda_2 x^2 + \cdots + \Lambda_{15} x^{15}$.Our next step is to find what the coefficients of this polynomial are. Since we know we have two blocks, we are going to find the coefficients for each of the blocks. For the first block, we are going to solve the following system of equations,

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_7 \\ S_2 & S_3 & \cdots & S_8 \\ \vdots & \vdots & \ddots & \vdots \\ S_7 & S_8 & \cdots & S_{13} \end{bmatrix} \begin{bmatrix} \Lambda_7 \\ \Lambda_6 \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} S_8 \\ S_9 \\ \vdots \\ S_{14} \end{bmatrix}.$$

We can use the syndromes we calculated to get the system,

$$\begin{bmatrix} \alpha^{20} & \alpha^{9} & \alpha^{26} & \alpha^{18} & \alpha^{28} & \alpha^{21} & \alpha^{3} \\ \alpha^{9} & \alpha^{26} & \alpha^{18} & \alpha^{28} & \alpha^{21} & \alpha^{3} & \alpha^{5} \\ \alpha^{26} & \alpha^{18} & \alpha^{28} & \alpha^{21} & \alpha^{3} & \alpha^{5} & \alpha^{7} \\ \alpha^{18} & \alpha^{28} & \alpha^{21} & \alpha^{3} & \alpha^{5} & \alpha^{7} & \alpha^{25} \\ \alpha^{28} & \alpha^{21} & \alpha^{3} & \alpha^{5} & \alpha^{7} & \alpha^{25} & \alpha^{17} \\ \alpha^{21} & \alpha^{3} & \alpha^{5} & \alpha^{7} & \alpha^{25} & \alpha^{17} & \alpha^{11} \\ \alpha^{3} & \alpha^{5} & \alpha^{7} & \alpha^{25} & \alpha^{17} & \alpha^{11} & \alpha^{6} \end{bmatrix} \begin{bmatrix} \Lambda_7 \\ \Lambda_6 \\ \Lambda_5 \\ \Lambda_4 \\ \Lambda_3 \\ \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} \alpha^{5} \\ \alpha^{7} \\ \alpha^{25} \\ \alpha^{17} \\ \alpha^{11} \\ \alpha^{6} \\ \alpha^{6} \end{bmatrix}.$$

Since we do not have more errors than the error decoding capability of the code, the syndrome matrix is invertible. We find its inverse

$$\begin{bmatrix} \alpha^9 & \alpha^{24} & \alpha^{13} & \alpha^{25} & \alpha^3 & \alpha^6 & \alpha^{11} \\ \alpha^{24} & \alpha^5 & \alpha^{26} & 1 & \alpha^4 & 1 & \alpha^{19} \\ \alpha^{13} & \alpha^{26} & \alpha^{15} & \alpha^3 & \alpha^6 & \alpha^{11} & \alpha^{25} \\ \alpha^{25} & 1 & \alpha^3 & \alpha^{25} & \alpha & \alpha^8 & \alpha^{17} \\ \alpha^3 & \alpha^4 & \alpha^6 & \alpha & \alpha^{22} & \alpha^{29} & \alpha^6 \\ \alpha^6 & 1 & \alpha^{11} & \alpha^8 & \alpha^{29} & \alpha^{17} & \alpha^7 \\ \alpha^{11} & \alpha^{19} & \alpha^{25} & \alpha^{17} & \alpha^6 & \alpha^7 & \alpha^{18} \end{bmatrix}.$$

We solve the following system

$$\begin{bmatrix} \Lambda_7 \\ \Lambda_6 \\ \Lambda_5 \\ \Lambda_4 \\ \Lambda_3 \\ \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} \alpha^9 & \alpha^{24} & \alpha^{13} & \alpha^{25} & \alpha^3 & \alpha^6 & \alpha^{11} \\ \alpha^{24} & \alpha^5 & \alpha^{26} & 1 & \alpha^4 & 1 & \alpha^{19} \\ \alpha^{13} & \alpha^{26} & \alpha^{15} & \alpha^3 & \alpha^6 & \alpha^{11} & \alpha^{25} \\ \alpha^{25} & 1 & \alpha^3 & \alpha^{25} & \alpha & \alpha^8 & \alpha^{17} \\ \alpha^3 & \alpha^4 & \alpha^6 & \alpha & \alpha^{22} & \alpha^{29} & \alpha^6 \\ \alpha^6 & 1 & \alpha^{11} & \alpha^8 & \alpha^{29} & \alpha^{17} & \alpha^7 \\ \alpha^{11} & \alpha^{19} & \alpha^{25} & \alpha^{17} & \alpha^6 & \alpha^7 & \alpha^{18} \end{bmatrix} \begin{bmatrix} \alpha^5 \\ \alpha^7 \\ \alpha^{25} \\ \alpha^{17} \\ \alpha^{11} \\ \alpha^6 \\ \alpha^6 \end{bmatrix}.$$

and then we find the zeroes of our error locating functions which are the inverses of the error positions.

Then to find the error locating polynomial for the second block, we solve the following system of equations,

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_{15} \\ S_2 & S_3 & \cdots & S_{16} \\ \vdots & \vdots & \ddots & \vdots \\ S_{15} & S_{16} & \cdots & S_{29} \end{bmatrix} \begin{bmatrix} \Lambda_{15} \\ \Lambda_{14} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} S_{16} \\ S_{17} \\ \vdots \\ S_{30} \end{bmatrix},$$

which is equivalent to

$$\begin{bmatrix}
\beta^{15} & \beta^{30} & \beta^{17} & \beta^{60} & \beta^{42} & \beta^{34} & \beta^{49} & \beta^{57} & 1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} \\
\beta^{30} & \beta^{17} & \beta^{60} & \beta^{42} & \beta^{34} & \beta^{49} & \beta^{57} & 1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} \\
\beta^{17} & \beta^{60} & \beta^{42} & \beta^{34} & \beta^{49} & \beta^{57} & 1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} \\
\beta^{60} & \beta^{42} & \beta^{34} & \beta^{49} & \beta^{57} & 1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 \\
\beta^{42} & \beta^{34} & \beta^{49} & \beta^{57} & 1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} \\
\beta^{34} & \beta^{49} & \beta^{57} & 1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} \\
\beta^{49} & \beta^{57} & 1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} \\
\beta^{57} & 1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} & \beta^{4} \\
1 & \beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} & \beta^{4} & \beta^{11} \\
\beta^{21} & \beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} & \beta^{4} & \beta^{11} & \beta^{10} \\
\beta^{2} & \beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} & \beta^{4} & \beta^{11} & \beta^{10} & \beta^{16} \\
\beta^{5} & \beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} & \beta^{4} & \beta^{11} & \beta^{10} & \beta^{16} & \beta^{13} \\
\beta^{38} & 0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} & \beta^{4} & \beta^{11} & \beta^{10} & \beta^{16} & \beta^{13} & \beta^{18} \\
0 & \beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} & \beta^{4} & \beta^{11} & \beta^{10} & \beta^{16} & \beta^{13} & \beta^{18} & 0 \\
\beta^{36} & \beta^{51} & \beta^{42} & 1 & \beta^{41} & \beta^{42} & \beta^{42} & \beta^{4} & \beta^{11} & \beta^{10} & \beta^{16} & \beta^{13} & \beta^{18} & 0 & \beta^{44}
\end{bmatrix}
\begin{bmatrix}
\Lambda_{15} \\ \Lambda_{14} \\ \Lambda_{13} \\ \Lambda_{12} \\ \Lambda_{11} \\ \Lambda_{10} \\ \Lambda_{9} \\ \Lambda_{8} \\ \Lambda_{7} \\ \Lambda_{6} \\ \Lambda_{5} \\ \Lambda_{4} \\ \Lambda_{3} \\ \Lambda_{2} \\ \Lambda_{1}
\end{bmatrix}
=
\begin{bmatrix}
\beta^{51} \\ \beta^{42} \\ 1 \\ \beta^{41} \\ \beta^{42} \\ \beta^{42} \\ \beta^{4} \\ \beta^{11} \\ \beta^{10} \\ \beta^{16} \\ \beta^{13} \\ \beta^{18} \\ 0 \\ \beta^{44} \\ \beta^{9}
\end{bmatrix}.$$

This cryptosystem has a few properties that are making it secure. First of all, the code has a compact description, which means that the key size will not be too large. This characteristic is one the prerequisites to a fully secure cryptosystem since having a large key size makes the encryption and decryption more difficult because of the length that the calculation requires. Furthermore, it is complicated for someone to find the structure of a word that was sent because of these blocks of different lengths. The structure of the Generalized Quasi-BCH codes is hidden well enough so that it is secure. Finally we notice in Example 4.0.2 that there is a possibility of decoding more errors with Generalized Quasi-BCH codes depending on where they are situated than a regular linear code. Indeed, we have seen that with the code of Example 4.0.2, we can correct up to 22 errors. If we look at all known linear codes of length 94 with dimension 24, which are the same parameters than our code, the maximum number of errors we can correct is 16 errors.

i	α^i
0	1
1	α
2	$\alpha+1$

i	β^i
0	1
1	β
2	β^2
3	$1+\beta$
4	$\beta+\beta^2$
5	$1+\beta+\beta^2$
6	$1+\beta^2$

i	γ^i
0	1
1	γ
2	γ^2
3	γ^3
4	$1+\gamma$
5	$\gamma+\gamma^2$
6	$\gamma^2+\gamma^3$
7	$1+\gamma+\gamma^3$
8	$1+\gamma^2$
9	$\gamma+\gamma^3$
10	$1+\gamma+\gamma^2$
11	$\gamma+\gamma^2+\gamma^3$
12	$1+\gamma+\gamma^2+\gamma^3$
13	$1+\gamma^2+\gamma^3$
14	$1+\gamma^3$

Table 4.1: Powers of the roots of $1+x+x^2$, $1+x+x^3$, and $1+x+x^4$ respectively.

i	α^i
0	1
1	α
2	α^2
3	α^3
4	α^4
5	$1+\alpha^2$
6	$\alpha+\alpha^3$
7	$\alpha^2+\alpha^4$
8	$1+\alpha^2+\alpha^3$
9	$\alpha+\alpha^3+\alpha^4$
10	$1+\alpha^4$
11	$1+\alpha+\alpha^2$
12	$\alpha+\alpha^2+\alpha^3$
13	$\alpha^2+\alpha^3+\alpha^4$
14	$1+\alpha^2+\alpha^3+\alpha^4$
15	$1+\alpha+\alpha^2+\alpha^3+\alpha^4$
16	$1+\alpha+\alpha^3+\alpha^4$
17	$1+\alpha+\alpha^4$
18	$1+\alpha$
19	$\alpha+\alpha^2$
20	$\alpha^2+\alpha^3$
21	$\alpha^3+\alpha^4$
22	$1+\alpha^2+\alpha^4$
23	$1+\alpha+\alpha^2+\alpha^3$
24	$\alpha+\alpha^2+\alpha^3+\alpha^4$
25	$1+\alpha^3+\alpha^4$
26	$1+\alpha+\alpha^2+\alpha^4$
27	$1+\alpha+\alpha^3$
28	$\alpha+\alpha^2+\alpha^4$
29	$1+\alpha^3$
30	$\alpha+\alpha^4$

Table 4.2: Powers of the root α of $1+x^2+x^5$.

i	β^i
0	1
1	β
2	β^2
3	β^3
4	β^4
5	β^5
6	$1+\beta+\beta^3+\beta^4$
7	$\beta+\beta^2+\beta^4+\beta^5$
8	$1+\beta+\beta^2+\beta^4+\beta^5$
9	$1+\beta^2+\beta^4+\beta^5$
10	$1+\beta^4+\beta^5$
11	$1+\beta^3+\beta^4+\beta^5$
12	$1+\beta^3+\beta^5$
13	$1+\beta^3$
14	$\beta+\beta^4$
15	$\beta^2+\beta^5$
16	$1+\beta+\beta^4$
17	$\beta+\beta^2+\beta^5$
18	$1+\beta+\beta^2+\beta^4$
19	$\beta+\beta^2+\beta^3+\beta^5$
20	$1+\beta+\beta^2$
21	$\beta+\beta^2+\beta^3$
22	$\beta^2+\beta^3+\beta^4$
23	$\beta^3+\beta^4+\beta^5$
24	$1+\beta+\beta^3+\beta^5$
25	$1+\beta^2+\beta^3$
26	$\beta+\beta^3+\beta^4$
27	$\beta^2+\beta^4+\beta^5$
28	$1+\beta+\beta^4+\beta^5$
29	$1+\beta^2+\beta^3+\beta^4+\beta^5$
30	$1+\beta^5$
31	$1+\beta^3+\beta^4$
32	$\beta+\beta^4+\beta^5$
33	$1+\beta+\beta^2+\beta^3+\beta^4+\beta^5$
34	$1+\beta^2+\beta^5$
35	$1+\beta^4$
36	$\beta+\beta^5$
37	$1+\beta+\beta^2+\beta^3+\beta^4$
38	$\beta+\beta^2+\beta^3+\beta^4+\beta^5$
39	$1+\beta+\beta^2+\beta^5$
40	$1+\beta^2+\beta^4$
41	$\beta+\beta^3+\beta^5$
42	$1+\beta+\beta^2+\beta^3$
43	$\beta+\beta^2+\beta^3+\beta^4$
44	$\beta^2+\beta^3+\beta^4+\beta^5$
45	$1+\beta+\beta^5$
46	$1+\beta^2+\beta^3+\beta^4$
47	$\beta+\beta^3+\beta^4+\beta^5$
48	$1+\beta+\beta^2+\beta^3+\beta^5$
49	$1+\beta^2$
50	$\beta+\beta^3$
51	$\beta^2+\beta^4$
52	$\beta^3+\beta^5$
53	$1+\beta+\beta^3$
54	$\beta+\beta^2+\beta^4$
55	$\beta^2+\beta^3+\beta^5$
56	$1+\beta$
57	$\beta+\beta^2$
58	$\beta^2+\beta^3$
59	$\beta^3+\beta^4$
60	$\beta^4+\beta^5$
61	$1+\beta+\beta^3+\beta^4+\beta^5$
62	$1+\beta^2+\beta^3+\beta^5$

Table 4.3: Powers of the root β of $1+x+x^3+x^4+x^6$.

Chapter 5

Conclusion

We started to give some necessary definitions and descriptions, in Chapter 2, in order to understand the context of the research. Then, in Chapter 3, we described the family of codes we are working on, Generalized Quasi-BCH codes and gave some examples on how to define and decode these codes. In the last chapter, we analyzed these codes and gave some interesting properties that make it an attractive type of codes.

We described a type of codes that are a generalization of BCH codes. These latter codes have been studied for years by mathematicians and have multiple interesting properties, including having an efficient decoding algorithm which makes our code possible to decode efficiently thanks to the same algorithm but generalized. Since we have not used any of the most common problems that are solvable using quantum computers, it makes Generalized Quasi-BCH codes a good candidate for post-quantum cryptography.

We have noticed in Example 4.0.2 that our code had a compelling property. Depending where the errors were situated, we could decode more errors than any linear code with the same parameters. Therefore, if someone intercepted the transmission, they would not be able to decode the errors and hence making it impossible to find the message sent. Since the structure of the code is hidden, nobody can know that we have blocks of different lengths, without the information that we are using Generalized Quasi-BCH codes, it is unachievable to distinguish this code from a usual linear code.

Now we have some questions unanswered that we need to work on in the future to improve this code. Could we use scramble matrices in order to hide even more the structure of our code ? Could we generalize these codes like the constacyclic codes, twisted codes, quasi twisted codes ?

Appendix A

Mathematica Codes

The Mathematica code below is the one used to make the tables and calculate the syndromes in Example 2.2.11.

```
Table[{PolynomialMod[x^i, {2, 1 + x + x^4}], i}, {i, 0, 14}]
Table[{PolynomialMod[1 + x^(5i) + x^(7i) + x^(8i) + x^(9i),
    {2, 1 + x + x^4}], i}, {i, 1, 4}]
Table[{PolynomialMod[x^i, {2, 1 + x^3 + x^4}], i}, {i, 0, 14}]
Table[{PolynomialMod[1 + x^(5i) + x^(7i) + x^(8i) + x^(9i),
    {2, 1 + x^3 + x^4}], i}, {i, 1, 4}]
```

The Mathematica code below is the code used to make the table and calculate the syndromes in Example 3.0.2.

```
Table[{PolynomialMod[x^i, {2, 1 + x + x^3}], i}, {i, 0, 6}]
Table[{PolynomialMod[x^(2i) + x^(3i) + x^(6i),
    {2, 1 + x + x^3}], i}, {i, 1, 2}]
```

The Mathematica code below is the code that was used to make the tables and to calculate the syndromes in Example 4.0.1.

```
Table[{PolynomialMod[x^i, {2, 1 + x + x^2}], i}, {i, 0, 2}]
Table[{PolynomialMod[x^i, {2, 1 + x + x^3}], i}, {i, 0, 6}]
Table[{PolynomialMod[x^i, {2, 1 + x + x^4}], i}, {i, 0, 14}]
```

```
Table[{PolynomialMod[1 + x^(2i),{2, 1 + x + x^2}], i}, {i, 1, 2}]
Table[{PolynomialMod[x + x^(2i) + x^(5i) + x^(6i), {2, 1 + x + x^3}],
    i},{i, 1, 2}]
Table[{PolynomialMod[1 + x^(3i) + x^(4i) + x^(5i) + x^(9i) + x^(10i)
    + x^(13i) + x^(14i),{2, 1 + x + x^3}], i}, {i, 1, 2}]
```

The Mathematica code below is the code used to make the tables, calculate the syndromes and find the inverse of the matrix in Example 4.0.2.

```
Table[{PolynomialMod[x^i, {2, 1 + x^2 + x^5}], i}, {i, 0, 30}]
Table[{PolynomialMod[x^i, {2, 1 + x + x^3 + x^4 + x^6}],
    i}, {i, 0, 62}]
Table[{PolynomialMod[1 + x^i + x^(3i) + x^(6i) + x^(10i) + x^(11i)
    + x^(12i) + x^(13i) + x^(15i) + x^(17i) + x^(18i) + x^(21i)
    + x^(22i)+ x^(24i) + x^(26i) + x^(27i) + x^(29i), {2, 1 + x^2 + x^5}],
    i},{i, 0, 30}]
Table[{PolynomialMod[1 + x^i + x^(3i) + x^(5i) + x^(7i) + x^(8i) + x^(11i)
    + x^(12i) + x^(14i) + x^(15i) + x^(17i) + x^(18i) + x^(19i) + x^(20i)
    + x^(22i) + x^(23i)+ x^(24i) + x^(27i) + x^(31i) + x^(32i) + x^(33i)
    + x^(35i) + x^(39i) + x^(40i) + x^(42i) + x^(44i) + x^(52i) + x^(55i)
    + x^(60i) + x^(61i)), {2, 1 + x + x^3 + x^4 + x^6}], i}, {i, 0, 30}]

<< FiniteFields`
SetFieldFormat[GF[2, 5], FormatType -> FunctionOfCoefficients[f32]]
ffmatrix = {{f32[0, 0, 1, 1], f32[0, 1, 0, 1, 1], f32[1, 1, 1, 0, 1],
   f32[1, 1], f32[0, 1, 1, 0, 1], f32[0, 0, 0, 1, 1], f32[0, 0, 0, 1]},
   {f32[0, 1, 0, 1, 1], f32[1, 1, 1, 0, 1], f32[1, 1],
   f32[0, 1, 1, 0, 1], f32[0, 0, 0, 1, 1], f32[0, 0, 0, 1],f32[1, 0, 1]},
   {f32[1, 1, 1, 0, 1], f32[1, 1, 0, 0, 0], f32[0, 1, 1, 0, 1],
   f32[0, 0, 0, 1, 1], f32[0, 0, 0, 1, 0], f32[1, 0, 1, 0, 0],
   f32[0, 0, 1, 0, 1]},
   {f32[1, 1, 0, 0, 0], f32[0, 1, 1, 0, 1], f32[0, 0, 0, 1, 1],
   f32[0, 0, 0, 1, 0], f32[1, 0, 1, 0, 0], f32[0, 0, 1, 0, 1],
   f32[1, 0, 0, 1, 1]},
   {f32[0, 1, 1, 0, 1], f32[0, 0, 0, 1, 1], f32[0, 0, 0, 1, 0],
   f32[1, 0, 1, 0, 0], f32[0, 0, 1, 0, 1], f32[1, 0, 0, 1, 1],
   f32[1, 1, 0, 0, 1]},
   {f32[0, 0, 0, 1, 1], f32[0, 0, 0, 1, 0], f32[1, 0, 1, 0, 0],
   f32[0, 0, 1, 0, 1], f32[1, 0, 0, 1, 1], f32[1, 1, 0, 0, 1],
   f32[1, 1, 1]},
   {f32[0, 0, 0, 1, 0], f32[1, 0, 1, 0, 0], f32[0, 0, 1, 0, 1],
   f32[1, 0, 0, 1, 1], f32[1, 1, 0, 0, 1], f32[1, 1, 1, 0, 0],
   f32[0, 1, 0, 1]}}
Inverse[ffmatrix]
```

Bibliography

[1] BCH codes. In F J MacWilliams and N J A Sloane, editors, *The Theory of Error-Correcting Codes*, volume 16 of *North-Holland Mathematical Library*, pages 257–293. Elsevier, 1977.

[2] Cyclic codes. In F J MacWilliams and N J A Sloane, editors, *The Theory of Error-Correcting Codes*, volume 16 of *North-Holland Mathematical Library*, pages 188–215. Elsevier, 1977.

[3] Linear codes. In F J MacWilliams and N J A Sloane, editors, *The Theory of Error-Correcting Codes*, volume 16 of *North-Holland Mathematical Library*, pages 1–37. Elsevier, 1977.

[4] Marco Baldi and Franco Chiaraluce. Cryptanalysis of a new instance of McEliece crypto system based on QC-LDPC codes. In *IEEE International Symposium on Information Theory - Proceedings*, pages 2591–2595, jun 2007.

[5] M. Barbier, C. Chabot, and G. Quintin. On quasi-cyclic codes as a generalization of cyclic codes. *Finite Fields and Their Applications*, 18(5):904 – 919, 2012.

[6] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding Random Binary Linear Codes in 2 n/20: How 1+1=0 Improves Information Set Decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 520–536, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[7] E Berlekamp. On decoding binary Bose-Chadhuri- Hocquenghem codes. *IEEE Transactions on Information Theory*, 11(4):577–579, oct 1965.

[8] E. Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19(5):590–592, sep 1973.

[9] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (Corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, may 1978.

[10] R.C. Bose and D.K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68 – 79, 1960.

[11] Marcel J E Golay. Notes on digital coding. *Proc. IEEE*, 37:657, 1949.

[12] Valerii Denisovich Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970.

[13] Richard Wesley Hamming. Error-detecting and error-correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[14] Alexis Hocquenghem. Codes correcteurs derreurs. *Chiffres*, 2(2):147–56, 1959.

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

[16] Robert J McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. Technical Report 44, Jet Propulsion Lab., CA, 1978.

[17] C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, page 215. IEEE, 2000.

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

[19] C E Shannon. A Mathematical Theory of Communication. *Bell Systems Technical Journal*, 27:623–656, 1948.

[20] Irfan Siap and Nilgun Kulhan. The structure of generalized quasi cyclic codes. *Appl. Math. E-Notes*, 5:24–30, 2005.

[21] Harshdeep Singh. Code based cryptography: Classic mceliece, 2019.

[22] G. Solomon and I Reed. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, (8):300–304, 1960.

[23] A Vardy. The Intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, (43):1757–1766, 1997.