AN OVERVIEW OF SCRAMBLING PROCESSES IN CODE BASED CRYPTOGRAPHY

By Marcela Gutierrez

A Thesis

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Science in Mathematics

Northern Arizona University

May 2020

Approved:

Bahattin Yildiz, Ph.D., Chair

Nandor Sieben, Ph.D.

Dana Ernst, Ph.D.

ABSTRACT

AN OVERVIEW OF SCRAMBLING PROCESSES IN CODE BASED CRYPTOGRAPHY

MARCELA GUTIERREZ

In this thesis we explore and analyze the scrambling processes used in code based cryptography. Cryptography is the study of finding secure ways to send confidential information. Code based cryptography refers to cryptosystems that rely on the hardness of decoding linear codes in order to keep processes secure. Many public key cryptosystems that have been proposed for the post-quantum era are based on code based cryptography. One way code based cryptosystems are made more secure is by scrambling. Scrambling refers to the various techniques used to hide the structure of a code and ultimately the information that was sent. We will first analyze and discuss a few of the more influential scrambling processes that have been employed in various cryptosystems within the literature. We also look at one recent cryptosystem in particular that brings a new approach to scrambling, by defining a new operation as part of their cryptosystem. Thus, instead of hiding the code, as is done in almost all classical schemes, they hide the ciphertext. This cryptosystem serves as the inspiration for our research. We will propose a new family of cryptosystems that can be generalized based on a potential operation we have found. In addition we will provide a comparative study of all the scrambling processes that are being used in the code based cryptosystems that are in the final stages of the post-quantum competition run by the National Institute of Standards and Technology.

Contents

Chapter 1

Introduction

1.1 Preliminaries

Coding theory is the study of transmitting data over noisy channels and recovering the message that was sent over the channel. This is often referred to as the study of error correcting codes. The field came about in response to an engineering problem. The main concern was being able to transmit digital information reliably over noisy channels while still being able to detect and correct the errors that occur in the transfer. Coding theory has many applications in communication systems. A few of these include satellite communications, cell phones, data compression, and others. The first main papers that marked the beginning of coding theory can be contributed to Shannon and Hamming [20, 33].

Cryptography on the other hand is the study of methods used to keeping information secure and confidential while transmitting information. One way we do this is through codes. Cryptography has many applications in the modern world. In today's world we often transmit data through electronic systems whether that be cell phones or computers. An example of this is when we put our credit card information online to buy products. Essentially, most of the information on the internet, in banks, and so on are kept secure by cryptosystems that are put in place in the modern world. This topic will continue to be relevant as we continue to rely more on communicating electronically. Because of the desire to keep cryptosystems secure for the future, the National Institute of Standards and Technology (NIST) is holding a contest to create more secure cryptosystems using codes so that our data is more secure in the modern world.

This thesis will explore the various methods that are used to keep cryptosystems secure which we will call scrambling. Scrambling is the process of hiding a codes structure in order to disguise it. This can be done in various different ways. For instance, The McEliece Cryptosystem disguises the generator matrix of a code by multiplying it by a scrambler matrix and a permutation matrix and giving the resulting matrix to the public [21]. In this

case it is essential the generator matrix is disguised so that an attacker can not simply detect what code is used based on the structure of its generator matrix. However, this is not the only way to hide the structure of a code. Scrambling can be done in many different ways and is not limited to just the key generation phase. We will explore other methods that will be used later in the paper. Scrambling has become an important aspect of the security of a cryptosystem. Hiding the structure of a code is important so that an attacker cannot just figure out what code was used, use the decryption algorithm for that well known code, and then recover the message instantly. So the security of a cryptosystem relies on how well one can disguise the code while still being able to use that code within our cryptosystem. Because of this, the various ways people attempt to scramble will play an influential role in how our data is kept secure in the future.

In the first chapter of the thesis we start by providing some key definitions and terms in coding theory. Later, we will discuss a few of the different types of codes such as Hamming, Goppa, and more that are referenced in this paper. We will then provide some background on code based cryptography and talk about one of the major public key cryptosystems: the McEliece Cryptosystem as well as its variations. In Chapter 2, we look at three different cryptosystems along with their scrambling techniques that inspired our research. In the next section we will begin to discuss a few of the different scrambling methods that exist today. One of the techniques we focus on uses the technique of scrambling the ciphertext. We will go more in depth with this cryptosystem as it was the inspiration for the attempted construction of our new cryptosystem. Next, in Chapter 3, we discuss our efforts in attempting to generalize a cryptosystem that scrambles the ciphertext. We then propose what we believe the structure of our cryptosystem should have in order to increase security. In Chapter 4, we will provide an overview of the submitted round 3 proposed cryptosystems in the NIST competition. We then compare each of the scrambling techniques used in these future cryptosystems. Finally, we wrap up the paper by providing some directions for future work in related fields.

1.2 Coding Theory

There are three main components to coding theory: encoding the message, decoding the message, which is also called the error correction phase, and recovering the original message that was sent. We give a few definitions and concepts that are essential in coding theory. First we must define what a code is. Let $q = p^m$ be a prime power and let $GF(q) = \mathbb{F}_q$ denote the finite field of size q.

Definition 1.2.1. A *code* of length n over $\mathbb{F}_q$ is a subset of $\mathbb{F}_q^n$.

Because of their structural advantages, throughout much of coding theory, liner codes are generally preferred.

Definition 1.2.2. A *linear code* C over $\mathbb{F}_q$ of length n is a vector subspace of $\mathbb{F}_q^n$.

Definition 1.2.3. A vector $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ in C is called a *codeword of* C.

Binary codes are codes over $\mathbb{F}_2$ and *ternary codes* are codes over $\mathbb{F}_3$. More generally, *q-ary codes* are codes over $\mathbb{F}_q$. If C is a k-dimensional subspace of $\mathbb{F}_q^n$, then C is an $[n, k]_q$ code.

Definition 1.2.4. The *dual code* of C is the orthogonal complement of the subspace of C of $\mathbb{F}_q^n$.

Definition 1.2.5. A *generator matrix* G for a linear code C is a matrix whose rows form a basis for C.

Every generator matrix can be put into a standard form using elementary row and column operations. Elementary row operations leave the code invariant as a subspace while the column operations transform the code to an *equivalent form*. Two codes are equivalent if one can be obtained from the other by a permutation of coordinates or by multiplying a column by a scalar. Thus any $[n, k]$-linear code over $\mathbb{F}_q$ is equivalent to a code that has a *systematic generator matrix*. A systematic generator matrix is of the form $[I_k|A]$ where I_k is the $k \times k$ identity matrix and A is a $k \times (n-k)$-matrix over $\mathbb{F}_q$.

Definition 1.2.6. The *parity check matrix* H of a code C is the matrix whose rows form a basis for the dual of C. In other words $\mathbf{c}$ is a codeword if and only if $H\boldsymbol{c}^T = 0$.

A parity check matrix of the form $H = [-A^T|I_{n-k}]$ is in standard form. It is natural to see a linear code expressed in terms of its parity check matrix or its generator matrix since one can be obtained from the other when in standard form. Below are other types of matrices.

Definition 1.2.7. A *sparse matrix* is a matrix where most of the elements are zero, i.e., the ratio of non-zero terms is smaller than a given upper bound.

Definition 1.2.8. A *dense matrix* is a matrix where most of the elements are non-zero, i.e., the ratio of non-zero terms is greater than a given lower bound.

Both sparse matrices and dense matrices are defined within a context where a measure for "sparseness" and "denseness" is given. Another concept that is crucial in coding theory is the Hamming distance.

Definition 1.2.9. The *Hamming distance* is defined between vectors in $\mathbb{F}_q^n$ as

$$d_H(\mathbf{x}, \mathbf{y}) = |\{i \mid x_i \neq y_i\}|,$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$.

Essentially, the Hamming distance finds the number of positions where the vectors $\mathbf{x}$ and $\mathbf{y}$ differ. In most cases distance is used to refer to hamming distance unless otherwise specified and we denote this by d. We will now look at an example.

Example 1.2.10. Let

$$\mathbf{x} = (0, 0, 1, 1, 0),\ \mathbf{y} = (1, 0, 0, 1, 0),\ \mathbf{z} = (0, 1, 1, 1, 1),\ \mathbf{w} = (1, 1, 1, 1, 1)$$

be vectors in $\mathbb{F}_2^5$. Then,

$$d(\mathbf{x}, \mathbf{y}) = 2 \text{ and } d(\mathbf{z}, \mathbf{w}) = 1.$$

Now we will look at the Hamming weight.

Definition 1.2.11. The *Hamming Weight* is defined to be the number of nonzero coordinates in a codeword $\mathbf{c}$.

Example 1.2.12. For instance, if $\mathbf{x} = (1, 0, 0, 1, 0)$, then the Hamming weight is given by $w_h(\mathbf{x}) = 2$.

It is important to recognize the relationship the hamming weight and hamming distance have with one another. That is, $d_H(\mathbf{x}, \mathbf{y}) = w_H(\mathbf{x} - \mathbf{y})$. This relationship is important when we talk about the minimum distance.

Definition 1.2.13. The *minimum distance* of C is $d(C) = \min\{d_H(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in C, \quad \mathbf{x} \neq \mathbf{y}\}$.

In regards to coding theory, the minimum distance is essential in telling us about a code's error correction or error detection ability.

Definition 1.2.14. A code C is *$u-$error-detecting* if, whenever a codeword incurs at least one but at most u errors, the resulting word is not a codeword. A code C is exactly *$u-$error-detecting* if it is $u-$error detecting but not $(u+1)$-error-detecting.

Theorem 1.2.15. *([26]) If a code has minimum distance d then the code C can detect up to $d-1$ errors.*

Definition 1.2.16. A code C is *$v-$error-correcting* if minimum distance decoding (equivalently maximum likelihood decoding) is able to correct v or fewer errors.

Theorem 1.2.17. *([26]) If a code has minimum distance d then the code can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors.*

Next, we go over the concept of a hash function.

Definition 1.2.18. A *hash function* is an algorithm that maps data sets of arbitrary length to smaller data sets of fixed length.

An ideal hash function guarantees that computing the hash value is easy for any input and the following are almost impossible

- to find a message that maps to a given hash value
- to find two different messages to the same hash value
- to modify a message without changing its hash value [23].

Definition 1.2.19. Let $\mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{F}_{q^m}^n$. The support E of $\mathbf{x}$, denoted by $Supp(\mathbf{x})$ is defined to be the $\mathbb{F}_q$ subspace of $\mathbb{F}_q^n$ generated by the coordinates of $\mathbf{x}$.

It is customary to denote a linear code over $\mathbb{F}_q^n$, by $[n, k, d]_q$, where n is the length of the code, k is the dimension, and d is the minimum distance.

We typically send or encode a message by $\boldsymbol{m}G$ where $\boldsymbol{m}$ is the original message that was sent and G is the generator matrix. The result is a vector of length of n. So if we start with k bits of information this becomes n bits. So the information rate $r = \frac{k}{n}$.

Definition 1.2.20. The *plaintext* is the the plain message m before encryption and the *ciphertext* is the message after encryption.

Example 1.2.21. Let C be a $[5, 3]$ linear code with generator matrix

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix},$$

and let the message $\boldsymbol{m} = (1, 0, 1)$. Now we can encode by

$$\boldsymbol{m}G = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Notice that mG is just the sum of the first and third rows of G. Note that we needed 5 bits to transmit a 3-bit message. So, the information rate is given by $r = \frac{3}{5}$. In general, if C is an $[n, k]$-code over a binary field, the information rate is given by $\frac{k}{n}$.

As mentioned above one of the three main components of coding theory is recovering the original message that was sent. Recovering the original message $\mathbf{m}$ from a codeword $\mathbf{c}$ means to find the coordinate vector of $\mathbf{c}$ in the basis that consists of the rows of G.

1.2.1 Syndrome Decoding

We now consider syndrome decoding, which is the default method used in decoding linear codes in the absence of any other decoding algorithm. The idea of syndrome decoding is based on cosets, syndromes, and the maximum likelihood principle. We let C be a linear code over $\mathbb{F}_q^n$ and let $\boldsymbol{u} \in \mathbb{F}_q^n$.

Definition 1.2.22. A *coset* is the set $C + \boldsymbol{u} = \{\boldsymbol{c} + \boldsymbol{u} : \boldsymbol{c} \in C\}$.

We call the word of least Hamming weight the *coset leader*. Let us do an example.

Example 1.2.23. Let $C = \{(0,0,0,0), (1,0,1,1), (0,1,0,1), (1,1,1,0)\}$ then, we have the following cosets:

$$\begin{aligned}
(0,0,0,0) + C: &\quad \{(0,0,0,0), (1,0,1,1), (0,1,0,1), (1,1,1,0)\} \\
(0,0,0,1) + C: &\quad \{(0,0,0,1), (1,0,1,0), (0,1,0,0), (1,1,1,1)\} \\
(0,0,1,0) + C: &\quad \{(0,0,1,0), (1,0,0,1), (0,1,1,1), (1,1,0,0)\} \\
(1,0,0,0) + C: &\quad \{(1,0,0,0), (0,0,1,1), (1,1,0,1), (0,1,1,0)\}
\end{aligned}$$

The cosets leaders are all the words in the first position of each row.

Definition 1.2.24. For any $\boldsymbol{w} \in \mathbb{F}_q^n$ and $\boldsymbol{c} \in C$ where C is a linear code the *syndrome* of $\boldsymbol{w}$ where $\boldsymbol{w} = \boldsymbol{c} + \boldsymbol{e}$ is $S(\boldsymbol{w}) = H\boldsymbol{w}^T$ where H is the parity check matrix for the linear code C.

We know that by definition for $\mathbf{c} \in C$, $H\boldsymbol{c}^T = 0$. So when we calculate the syndrome of $\boldsymbol{w}$ we have that

$$S(\boldsymbol{w}) = H\boldsymbol{w}^T = H(\boldsymbol{c} + \boldsymbol{e})^T = H\boldsymbol{c}^T + H\boldsymbol{e}^T = 0 + H\boldsymbol{e}^T = S(\boldsymbol{e}).$$

So we have that $S(\boldsymbol{e}) = S(\boldsymbol{w})$ must be in the same coset. The decoding procedure is as follows:

1. First calculate $S(\boldsymbol{w})$ where $\boldsymbol{w}$ is the received work.
2. Next find the coset leader $\boldsymbol{u}$ where $S(\boldsymbol{u}) = S(\boldsymbol{w})$.
3. Decode $\boldsymbol{w}$ as $\boldsymbol{v} = \boldsymbol{w} - \boldsymbol{u}$

This method works because we know that cosets partition the whole set. So we have spheres of $r = \lfloor \frac{d-1}{2} \rfloor$ centered at codewords of C. So as long as $w(\boldsymbol{e}) \leq \lfloor \frac{d-1}{2} \rfloor$, then $\boldsymbol{w}$ will belong to only one sphere.

1.3 Special Types of Codes

There are many families of codes in coding theory. A few of these are Hamming codes, Golay codes, cyclic codes, BCH codes, quadratic residue codes, quasi-cyclic (QC) codes, quasi-twisted (QT) codes, Reed-Solomon codes, Goppa codes, and LDPC codes. We will now go more in depth into certain types of codes that are relevant for the thesis.

1.3.1 Hamming Codes

Hamming codes were first discovered by R.W Hamming and M.J.E Golay in 1950. They are a type of linear error correcting codes. Hamming codes are known to be easy to encode and decode and thus are used in many different applications and examples. We will look more specifically at binary Hamming codes [26].

Definition 1.3.1. Let $r \geq 2$. A *Binary Hamming code* is a binary linear code of length $2^r - 1$ where the columns of the parity check matrix H consist of all nonzero vectors of set $\mathbb{F}_2^r$. We often denote this $\text{Ham}(r, 2)$.

Proposition 1.3.2. *([26]) The following are true of Hamming Codes:*

- *The dimension of Ham$(r, 2)$ is $k = 2^r - 1 - r$.*
- *The distance of Ham$(r, 2)$ is $d = 3$. That is, binary hamming codes are single error correcting.*

Now we will talk about the process of decoding Hamming Codes.

Decoding Hamming Codes

Let $\boldsymbol{y}$ be the received word. We first calculate the syndrome $S(\boldsymbol{y}) = H\boldsymbol{y}^T$. If $S(\boldsymbol{y}) = 0$, then $\boldsymbol{y}$ is the codeword sent and we are done. Otherwise $S(\boldsymbol{y})$ will be a binary $r-$tuple, which corresponds to a column j in the parity check matrix H based on how it was constructed. Thus, the error occurs in position j, so our word sent is $\boldsymbol{y} - \boldsymbol{e}_j$ ([26]). Let us illustrate this on an example.

Example 1.3.3. Let us look at the $[7, 4]$ binary Hamming code with parity check matrix

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

Now we let

$$\boldsymbol{m} = (0,1,1,0,0,1,1) \text{ and } \boldsymbol{y} = (0,1,1,0,1,1,1)$$

We calculate the syndrome, $H\boldsymbol{y}^T = (1,0,1)$. This corresponds to the 5th entry in H. So our error occurs in the 5th position. That is, $\boldsymbol{e}_5 = (0,0,0,0,1,0,0)$.Now we can calculate

$$\boldsymbol{y} - \boldsymbol{e}_5 = (0,1,1,0,1,1,1) - (0,0,0,0,1,0,0) = (0,1,1,0,0,1,1) = \boldsymbol{m}.$$

1.3.2 Cyclic Codes

Definition 1.3.4. A subset S of $\mathbb{F}_q^n$ is cyclic if $(a_{n-1}, a_1, a_2, \ldots, a_{n-2}) \in S$ whenever $(a_0, a_1, a_2, \ldots, a_{n-1}) \in S$. A linear code is called a cyclic code if C is a cyclic set. The word $(u_{n-r}, \ldots, u_{n-1}, u_0, u_1, \ldots, u_{n-r-1})$ is said to be obtained from the word $(u_0, u_1, \ldots, u_{n-1})$ by cyclically shifting r positions.

We define π to be the linear map from $\mathbb{F}_q^n \longrightarrow \mathbb{F}_q[x]/(x^n-1)$. That is $\pi(a_0, a_1, \ldots, a_{n-1}) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$. So, we can describe any cyclic code in this algebraic nature. This will be key when talking about generator polynomials.

Theorem 1.3.5. *([26]) Let π be the linear map defined above. Then a nonempty subset C is a cyclic code if and only if $\pi(C)$ is an ideal of $\mathbb{F}_q[x]/(x^n-1)$.*

Theorem 1.3.6. *([26]) Let I be a nonzero ideal in $\mathbb{F}_q[x]/(x^n-1)$ and let $g(x)$ be a nonzero monic polynomial in I of minimal degree. Then $g(x)$ is a generator of I and divides x^n-1.*

Definition 1.3.7. The unique monic polynomial of least degree of a nonzero ideal I of $\mathbb{F}_q[x]/(x^n-1)$ is called the *generator polynomial* of I. For a cyclic code C, the generator polynomial of $\pi(C)$ is also called the generator polynomial of C.

1.3.3 Quasi-Cyclic Codes

Definition 1.3.8. An $n \times n$ circulant matrix C for $\mathbf{c} = (c_0, c_1, \ldots, c_n)$ is of the form

$$C = \begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} & c_n \\ c_n & c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_n & c_0 & \cdots & c_{n-2} \\ \vdots & & & & \\ c_2 & \cdots & & \cdots & c_1 \\ c_1 & c_2 & \cdots & c_n & c_0 \end{bmatrix}$$

Definition 1.3.9. A *block circulant matrix* is a matrix that is composed of circulant square blocks of identical size. The size of each of the circulant blocks is called the order r and the index l is the number of circulant blocks in a row.

Definition 1.3.10. A linear code C over $\mathbb{F}_q$ is *quasi cyclic* of index l if $(c_{n-l}, c_{n-l+1}, \ldots, c_0, c_1, \ldots, c_{n-l-1}) \in C$ whenever $(c_0, c_1, \ldots, c_{n-1}) \in C$. We refer to these as (l, k_0) QC codes, where l is the index, lr is the length, and has dimension $k = k_0 r$.

Definition 1.3.11. A generator matrix of the quasi cyclic code of index l and order r is of the form

$$G = \begin{bmatrix} G_{1,1} & G_{1,2} & \cdots & G_{1,l-1} & G_{1,l} \\ G_{2,1} & G_{2,2} & G_{2,3} & \cdots & \\ \cdot & G_{3,2} & G_{3,3} & & \\ \vdots & & & & \\ G_{r,1} & G_{r,2} & \cdots & G_{r,l-1} & G_{r,l} \end{bmatrix}$$

where each of $G_{i,j}$ is a circulant matrix.

1.3.4 QC-MDPC Codes

Definition 1.3.12. A *Moderate Density Parity Check Code* is a binary linear code which has a sparse parity check matrix.

Definition 1.3.13. A (l, k_0, r, w)-QC-MDPC code is an (l, k_0) quasi-cyclic code of length $n = lr$, dimension $k = k_0 r$, order r and has a parity check matrix of constant row weight $w = \mathcal{O}(\sqrt{n})$.

The parity check matrix of such a code is not exactly sparse, however it still has a bound on the number of zeros, thus the term "moderate-density" is used for these type of codes. QC-MDPC codes have been used extensively in some of the most recent successful implementations of code-based cryptographic schemes.

1.3.5 Ideal Codes

Ideal codes are analogous to cyclic codes. Essentially, Ideal codes are codes that have a systematic generator matrix that are composed of blocks of ideal matrices. First we will give some motivating definitions before we define the code.

Definition 1.3.14. Let $P \in F_q[X]$ be a polynomial of degree n and v be a vector of $\mathbb{F}_{q^m}^n$. We define the *ideal matrix generated by* $\boldsymbol{v}$ to be a square matrix of size $n \times n$ of the form,

$$\mathcal{I}M(\boldsymbol{v}) = \begin{bmatrix} \mathbf{v} \\ X\mathbf{v} \mod P \\ \vdots \\ X^{n-1}\mathbf{v} \mod P \end{bmatrix}$$

The square matrix is denoted by $\mathcal{IM}(\mathbf{v})$.

Definition 1.3.15. Let $P(X) \in \mathbb{F}_q[X]$ be a polynomial of degree n. An $[ns, nt]_{q^m}$ code $\mathcal{C}$ is an (s,t)-ideal code if its generator matrix in systematic form is, $G = [I_{tn}|A]$ where

$$A = \begin{bmatrix} \mathcal{IM}(\boldsymbol{g}_{1,1}) & \cdots & \mathcal{IM}(\boldsymbol{g}_{1,s-t}) \\ \vdots & & \\ \mathcal{IM}(\boldsymbol{g}_{t,1}) & \cdots & \mathcal{IM}(\boldsymbol{g}_{t,s-t}) \end{bmatrix}$$

where $(\boldsymbol{g}_{i,j})$ are vectors of $\mathbb{F}_{q^m}^n$ where $1 \leq i \leq s-t$ and $1 \leq j \leq t$. We say that $\mathcal{C}$ is generated by the $(\boldsymbol{g}_{i,j})$.

Notice that ideal codes are essentially just a generalization of Quasi Cyclic codes. Quasi Cyclic codes and Ideal codes have generator matrices of the same form. The only difference is that the blocks of a generator matrix of quasi cyclic codes are made up of circulant matrices as opposed to Ideal matrices.

Ideal codes are often shortly called to be $s-$ideal codes. So C is an $[sn, n]$ ideal code that is generated by $(\boldsymbol{g}_1, ..., \boldsymbol{g}_{s-1})$. That is, $\mathcal{C} = \{(u, u\boldsymbol{g}_1, ..., u\boldsymbol{g}_{s-1})\}$

1.3.6 Gabidulin Codes

Now we will talk about Gabidulin Codes. Before we do so, we need one motivating definition of $q-$ polynomials.

Definition 1.3.16. The set of $q-$polynomials over $\mathbb{F}_{q^m}$ is the set of the polynomials such that,

$$\{P(X) = \textstyle\sum_{i=0}^{r} p_i X^{q^i}, \text{ with } p_i \in \mathbb{F}_{q^m} \text{ and } p_r \neq 0\}.$$

Now we can define the Gabidulin codes. Essentially, we are going to think of these codes as evaluating $q-$polynomials over a given vector.

Definition 1.3.17. Let $k, n, m \in \mathbb{N}$ such that $k \leq n \leq m$ and $\mathbf{g} = (g_1, ..., g_n)$ be a $\mathbb{F}_q$ linearly independent family of elements of $\mathbb{F}_{q^m}$. $\mathcal{G}_{\mathbf{g}}$ is

$$\{P(\mathbf{g}), deg_q P \leq k\}$$

Note that $P(\mathbf{g}) = (P(g_1, \cdots, P(g_v))$.

The generator matrix for the Gabiludin codes is given by

$$G = \begin{pmatrix} g_1 & \cdots & g_n \\ g_1^q & \cdots & g_n^q \\ \vdots & \ddots & \vdots \\ g_1^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{pmatrix}.$$

1.3.7 BCH codes

BCH codes, named after Bose, Chaudhuri, and Hocquerghem, were first discovered in 1959-1960. They are essentially a special type of cyclic codes. They can also be viewed as a generalization of Hamming codes for multiple error correction. We will provide some defintions about finite fields to help motivate the definition for BCH codes [26].

Definition 1.3.18. An element α in a finite field $\mathbb{F}_q$ is called a *primitive element* of $\mathbb{F}_q$ if $\mathbb{F}_q = \{0, \alpha, \alpha^2, \ldots, \alpha^{q-1}\}$.

Definition 1.3.19. The *order* of a nonzero element $\alpha \in \mathbb{F}_q$ denoted by $\text{ord}(\alpha)$ is the smallest positive integer k such that $\alpha^k = 1$.

Definition 1.3.20. *A minimal polynomial* of $\alpha \in \mathbb{F}_{q^m}$ with respect to $\mathbb{F}_q$ is a nonzero monic polynomial $f(x)$ of the least degree in $\mathbb{F}_q[x]$ such that $f(\alpha) = 0$.

Definition 1.3.21. The $\text{lcm}(f_1(x), f_2(x))$ of two nonzero polynomials $f_1(x), f_2(x) \in \mathbb{F}_q[x]$ is defined to be the monic polynomial of lowest degree which is both a multiple of $f_1(x)$ and $f_2(x)$. So, in general the $\text{lcm}(f_1(x), f_2(x), \cdots, f_k(x))$ is said to be the monic polynomial of lowest degree which is a multiple of each of $f_1(x), f_2(x), ..., f_k(x)$.

Definition 1.3.22. Let α be a primitive element of $\mathbb{F}_{q^m}$ and let $M^i(x)$ be the minimal polynomial of α^i with respect to $\mathbb{F}_q$. A *(primitive) BCH code* of length $n = q^m - 1$ with distance δ is a $q-$ary cyclic code generated by $g(x) : \text{lcm}(M^a(x), M^{a+1}(x), ..., M^{a+\delta-2}(x))$ for some integer a.

The parameters of a binary BCH code are of the following: Block length $n = 2^m - 1$, minimum distance $d_{min} \geq 2t + 1$ where $t \leq 2^{m-1}$ and $m \geq 3$. We call it a $t-$error-correcting BCH code [21].

Decoding BCH codes

Now we will discuss one of the methods for decoding BCH codes that will be used in this thesis. Let w be our received vector. Recall that $\boldsymbol{w} = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{c}$ is a codeword and $\boldsymbol{e}$ is the error vector. Let $\boldsymbol{w}(x) = w_0 + w_1x + w_2x^2 + ... + w_{n-1}x^{n-1}$ be the received vector and let $\boldsymbol{e}(x)$ be the error vector. Also, let $\boldsymbol{c}(x) = c_0 + c_1x + ...c_{n-1}x^{n-1}$. Then the syndrome of $w(x)$ is $(S_0, S_1, \cdots, S_{2t}) := (w_0, w_1, \cdots, w_{n-1})H^T$. So

$$S_i = \boldsymbol{w}(\alpha^i) = w_0 + w_1\alpha^i + w_2\alpha^{2i} + ... + w_{n-1}\alpha^{(n-1)i}$$

where α_i is a root of the respective polynomial $M^i(x)$. So $S_i = \boldsymbol{w}(\alpha^i) = \boldsymbol{e}(\alpha^i)$.

Then if we assume that the errors take place at positions $i_0, i_1, ..., i_{l-1}$ with $l \leq t$, then $\boldsymbol{e}(x) = x^{i_0} + x^{i_1} + \cdots + x^{i_{l-1}}$. So now we have the following system of equations,

$$\begin{aligned}
S_0 &= \alpha^{i_0} + \alpha^{i_1} + ... + \alpha^{i_{l-1}} = \boldsymbol{w}(\alpha) \\
S_1 &= (\alpha^{i_0})^2 + (\alpha^{i_1})^2 + ... + (\alpha^{i_{l-1}})^2 = \boldsymbol{w}(\alpha^2) \\
&\vdots \\
S_{2t} &= (\alpha^{i_0})^{2t} + (\alpha^{i_1})^{2t} + ... + (\alpha^{i_{l-1}})^{2t} = \boldsymbol{w}(\alpha^{2t})
\end{aligned}$$

Now any method that will solve the system of equations is a decoding algorithm for BCH codes. We will describe the specific algorithm used for a $[15, 7, 5]$ BCH code as it will be essential in this thesis. For this BCH code, it is at most 2 error correcting. So, let us assume that our received vector w has exactly 2 errors and that its errors occur at positions i and j. So we know $S_i = w(\alpha_i)$. Define $X : \alpha^{i_1}$ and $Y := \alpha^{i_2}$. So our system of equations becomes,

$$\begin{aligned}
X + Y &= S_1 \\
X^2 + Y^2 &= S_2 \\
X^3 + Y^3 &= S_3
\end{aligned}$$

We could keep writing the corresponding system of equations but the above is all that is needed for the decoding process. Our goal is find our error positions and that is to essentially find X and Y. We know that $(X-z)(Y-z) = z^2-(X+Y)z+XY$. So if we are able to factor the above polynomial then we will reveal what X and Y are. We know that $(X + Y) = S_1$. So, now all we need to do is calculate XY. We can also say that,

$$S_1^3 = (X + Y)^3 = (X^2 + Y^2)(X + Y) = X^3 + Y^3 + XY(X + Y) = S_3 + XYS_1.$$

Solving the above for XY we get, $XY = S_1^2 - \frac{S_3}{S_1} = S_2 - \frac{S_3}{S_1}$. So, once we calculate XY, then we can substitute into the above polynomial and factor to find X and Y. This decoding process will be essential later in the thesis [34].

1.3.8 Reed Solomon Codes

Reed Solomon codes are just a special class of BCH codes. These codes will be crucial in defining Goppa codes.

Definition 1.3.23. Let α be a primitive element of $\mathbb{F}_q$. A *Reed Solomon Code* (RS) over $\mathbb{F}_q$ is a BCH code over $\mathbb{F}_q$ of length $q-1$ generated by

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

with $a \geq 1$ and $q-1 \geq \delta \geq 2$.

Theorem 1.3.24. *([26]) A Reed Solomon code has parameters* $[q-1, q-\delta, \delta]$.

A narrow sense RS code is a RS code where $a = 1$.

Theorem 1.3.25. *([26]) Let* α *be a primitive element of the finite field* $\mathbb{F}_q$ *and let* $q-1 \geq \delta \geq 2$. *The narrow-sense* $q-$*ary RS code with generator polynomial.*

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

is equal to,

$$\{(f(1), f(\alpha), f(\alpha^2), \ldots, f(\alpha^{q-2})) : f(x) \in \mathbb{F}_q[x] \text{ and } deg(f(x)) < q-\delta\}.$$

Corollary 1.3.26. *([26]) Let* α *be a primitive element of* $\mathbb{F}_q$ *and let* $q-1 \geq \delta \geq 2$. *The following is a generator matrix for a RS code generated by* $g(x) = (x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{\delta-1})$

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{q-2} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(q-2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{q-\delta-1} & \alpha^{2(q-\delta-1)} & \cdots & \alpha^{(q-\delta-1)(q-2)} \end{pmatrix}.$$

We let $\mathbb{F}^* = \mathbb{F}\backslash\{0\}$.

Definition 1.3.27. Let $n \leq q$. Let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ where $(1 \leq i \leq n)$, α_i are distinct elements of $\mathbb{F}_q$ and let $\boldsymbol{v} = (v_1, v_2, \cdots, v_n)$ where $v_i \in \mathbb{F}_q^*$ for $i \in [1, n]$. For $k \leq n$, the *generalized Reed-Solomon code* $\mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{v})$ is,

$$\{(v_1 f(\alpha_1), v_2 f(\alpha_2), \cdots, v_n f(\alpha_n) : f(x) \in \mathbb{F}_q[x] \text{ and } deg(f(x)) < k\}.$$

Theorem 1.3.28. *([26]) The parameters of a* $GRS_k(\boldsymbol{\alpha}, \boldsymbol{v})$ *are* $[n, k, n-k+1]$.

A special class of GRS codes is the alternat codes, which are defined as follows:

Definition 1.3.29. An *alternant* code $\mathcal{A}_k(\boldsymbol{\alpha}, \boldsymbol{v'})$ over the finite field $\mathbb{F}_q$ is the sub field subcode $GRS_k(\boldsymbol{\alpha}, \boldsymbol{\alpha_v}) \mid_{\mathbb{F}_q}$.

1.3.9 Goppa Codes

Goppa codes are one of the most important families of codes because of their use in cryptosystems such as the McEliece Cryptosystem which we will describe later. Goppa codes are a special subcategory of alternant codes.

Definition 1.3.30. Let $g(z)$ be a polynomial in $\mathbb{F}_{q^m}[z]$ for some fixed m and let $L = \{\alpha_1, \cdots, \alpha_n\}$ be a subset of $\mathbb{F}_q^m$ where $L \cap \{\text{zeroes of } g(z)\} = \emptyset$. For $\boldsymbol{c} = (c_1, \cdots, c_n) \in \mathbb{F}_q^n$ we let

$$R_c(z) = \sum_{i-1}^{n} \frac{c_i}{z - \alpha_i}.$$

The Goppa code is defined by $\mathcal{T}(L, g) = \{c \in \mathbb{F}_q^n : R_c(z) \equiv 0 \mod P\}$.

The polynomial $g(z)$ is called the *Goppa polynomial*.

Proposition 1.3.31. *([26]) The parity check matrix for a Goppa code $\mathcal{T}(L, g) = \{\boldsymbol{c} \in \mathbb{F}_q^n : H\boldsymbol{c}^T = 0\}$ with Goppa polynomial $g(z)$ of degree t is given by*

$$H = \begin{pmatrix} g(\alpha_1)^{-1} & \cdots & g(\alpha_n)^{-1} \\ \alpha_1 g(\alpha_1)^{-1} & \cdots & \alpha_n g(\alpha_n)^{-1} \\ \vdots & \cdots & \vdots \\ \alpha_1^{t-1} g(\alpha_1)^{-1} & \cdots & \alpha_n^{t-1} g(\alpha_n)^{-1} \end{pmatrix}.$$

1.3.10 Rank Metric Codes

Most codes are constructed over the Hamming metric as described above. But not all codes are. Below we will describe a new metric (the rank metric) and discuss some codes that are based on the rank metric. First we will define the rank metric of a code.

Definition 1.3.32. Let q be a power of a prime p, m an integer, and let $\boldsymbol{V}_n$ be a $\boldsymbol{n}$ dimensional vector space over the finite field $\mathbb{F}_{q^m}$. Let $\mathcal{B} = (\beta_1, \cdots, \beta_m)$ be a basis of $\mathbb{F}_{q^m}$. We view this basis as a basis for the m-dimensional vector space over $\mathbb{F}_q$. By definition of basis, each coordinate $x_i = \sum_{i=1}^{m} x_{ij}\beta_i$. Let $\mathcal{F}_i(x)$ be the map from $\mathbb{F}_{q^m}$ to $\mathbb{F}_q$ where $\mathcal{F}_i(x)$ is the $i-$th coordinate of x in the basis $\mathcal{B}$. The $m \times n$ matrix associated to x is $\mathcal{M}(x)$ where $\mathcal{M}(x) = (x_{i,j}) =$ where $1 \leq i \leq m$ and $1 \leq j \leq n$. Now, the *rank weight* $||x||$ of x is defined as

$$||x|| = Rank(\mathcal{M}(x)).$$

Definition 1.3.33. A rank code C of length n and dimension k over $\mathbb{F}_{q^m}$ is a subspace of dimension k of $\mathbb{F}_{q^m}$ embedded with the rank metric.

Definition 1.3.34. Let $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{F}_{q^m}^n$. The support E of $\mathbf{x}$ is denoted by $Supp(\mathbf{x})$ is the $\mathbb{F}_q$ subspace of $\mathbb{F}_q^m$ generated by the coordinates of $\mathbf{x}$.

From the above definition of rank we can say that $\dim E = ||x||$.

1.3.11 Low Rank Parity Check Codes

A type of code that is based on the Rank Metric is the class of Low Rank Parity Check Codes. We give the following definition below.

Definition 1.3.35. A *Low Rank Parity Check* (LRPC) code of rank d, length n and dimension k over $\mathbb{F}_{q^m}$ is a code such that the parity check matrix for the code is a $(n-k) \times n$ matrix $H(h_{ij})$ such that the sub-vector space of $\mathbb{F}_{q^m}$ generated by its coefficients h_{ij} has dimension at most d. Denoting $\boldsymbol{F}$ the sub-vector space of $\mathbb{F}_{q^m}$ generated by its coefficients h_{ij} of H we denote $\{\boldsymbol{F}_1, \boldsymbol{F}_2, \cdots, \boldsymbol{F}_d\}$ as one of its basis.

1.4 Code Based Cryptography

Cryptography is the study of finding various methods and techniques in order to keep our information secure when it is transmitted. The basic model for cryptography can be summarized in the following figure:

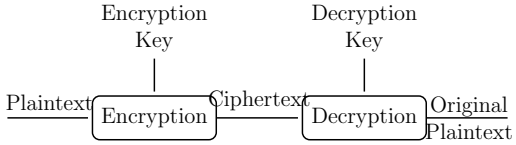


Figure 1.1: Basic public-key encryption system model.

Quantum computers have been known to solve many of the public key cryptosystems that are working today. More precisely, efficient quantum algorthms have been developed to solve problems such as the discrete logarithm problem, integer factorization problem, which are essential in the security of many known cryptosystems such as the RSA, the Elliptic Curve Cryptographu, etc. However, to date, no efficient algorithm has been found for the decoding problem of a random code, which has made cryptosystems that depend on

the hardness of decoding random codes potentially resistant to quantum computers. Such cryptosystems are generically known as code based cryptosystems. As mentioned before, code cryptosystems rely on the hardness of decoding random linear codes, which has been shown to be an NP-hard problem.

Figure 1.2 is an illustration that gives an overview of the use of code-based cryptography, where the encoded codeword from the original message is intentionally modified before up to t bit errors. The decryption is easy since the receiver knows the secret code (decrypt) the message, while, for anyone else, inferring the message from the transmitted codeword is an NP-Complete problem.

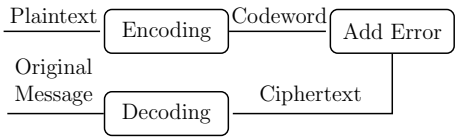


Figure 1.2: Code-based cryptography model in a nutshell.

In the next few sections we will discuss one of the most well known and first code based cryptosystems that was based on the algebraic capability of error correcting codes, namely the McEliece Cryposystem as well as some of the variations of the McEliece cryptosystem.

1.4.1 The Classical McEliece Cryptosystems

The McEliece cryptosystem is one of two major public key cryptosystems along with RSA that came up around the same time. The cryptosystem was introduced in 1978 [31]. The McEliece Cryposystem was not accepted and used in security implementations due to its large key size even though it has faster encryption and decryption rates then most public key systems. However, recently it has started to generate more interest along with other code-based cryptosystems due to the emerging threat of quantum computers. The scheme is based on error correcting codes, in particular on Goppa codes, and relies on the difficulty of decoding a random error message in linear codes. In contrast most public key systems are based on the difficulty of certain computation problems such as integer factorization problem (in RSA) and the discrete logarithm problem (in elliptic curve cryptography) [31]. But both of these problems are known to be solved quite efficiently by quantum computers since their introduction.

Now we will describe the structure of the original scheme. The McEliece cryptosystem is typically based on Goppa codes, which are an algebraically defined family of error correcting codes. Goppa codes have a well known decoding algorithm due to Patterson [30]. Let C be

a binary t-error correcting Goppa code of length n and dimension k. G is a $k \times n$ generator matrix for C. G is constructed via a randomly chosen irreducible polynomial of degree t over $GF(2^t)$. S is a $k \times k$ random, non singular, dense scrambler matrix, and P is an $n \times n$ permutation matrix. The matrix $G' = SGP$ is made public while S, G, and P are secret [31]. Below is the structure of the McEliece cryptosystem:

Key Generation

1. Public Key: G' and t
2. Private Key: G, S, P
3. Plaintext: $\mathbf{m} \in \mathbb{F}_2^k$, the set of all k-bit vectors

Encryption

1. 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 where $w_H(\mathbf{e}) \leq t$.

Decryption

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

We will now give an example of how the McEliece cryptosystem works. Note that even though Goppa codes are used in the original McEliece Cryposystem, we can use other codes as well to demonstrate how the scheme works, which is what we will be doing. (The Goppa code used in the original scheme had a 512×1024 generator matrix.)

Example 1.4.1. Consider the code C to be a binary $[7, 4]$-Hamming Code. Let

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

$$S = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix},$$

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

Then our public matrix is as follows

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

We can see that G' is an equivalent code to G. Let

$$\boldsymbol{m} = (1, 1, 0, 1) \text{ and } \boldsymbol{e} = (0, 0, 0, 0, 1, 0, 0)$$

Then

$$\boldsymbol{c} = \boldsymbol{m}G' + \boldsymbol{e} = (0, 1, 1, 0, 0, 1, 0) + (0, 0, 0, 1, 0, 0) = (0, 1, 1, 0, 1, 1, 0)$$

Now, we calculate $\boldsymbol{c'}$.

$$\boldsymbol{c'} = \big(0, 1, 1, 0, 1, 1, 0\big) \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} = \big(1, 0, 0, 0, 1, 1, 1\big)$$

Now we apply the efficient decoding algorithm for Hamming codes to $\boldsymbol{c'}$ to find out the error position happens in position 7. So $\boldsymbol{m'} = (1, 0, 0, 0, 1, 1, 0)$. Now we multiply by S^{-1} to get back $\boldsymbol{m}$.

1.4.2 Variations on McEliece

After its inception, many variations of the McEliece Cryptosystem were introduced. One such variation of the McEliece cryptosystem was introduced in 1986 by Niederreiter and is

referred to as the Niederreiter cryptosystem [23]. There are few changes that are made such as the public key scrambles the parity check matrix as opposed to the generator matrix. The reasoning behind this is because the cryptosystem uses the syndrome as the ciphertext instead of the actual codeword [23]. This results in more work being done in the encryption phase rather than the decryption phase. Some of the advantages of this cryptosystem are the smaller key size and a faster implementation than McEliece [25].

The scheme can be described as follows: Let C be a $[n, k, d]$-Goppa code capable of correcting up to t errors. H is an $(n-k) \times n$ parity check matrix. S is the $k \times k$ random dense scrambler matrix and P is the $n \times n$ permutation matrix. The matrix $H' = SHP$ is made public. We also calculate the inverses of S and P. Then, H, S^{-1}, P^{-1} are made secret [23]. Below is the structure of the Niederreiter cryptosystem which is very similar to the original McEliece:

Key Generation

1. Public Key: $H' = SHP$ and t
2. Private Key: H, S^{-1}, P^{-1}
3. Plaintext: $\mathbf{m} \in \mathbb{F}_2^k$, the set of all k-bit vectors

Encryption

1. First we let the message the $\boldsymbol{m}$ be denoted as a binary string $\boldsymbol{e}$ of length n and weight t.
2. Encrypt the plain text $\mathbf{m}$ into $\mathbf{c} = H'\mathbf{e}^T$.

Decryption

1. Compute $\mathbf{c}' = S^{-1}\boldsymbol{c}$
2. Decode $\mathbf{e}'$ from $\mathbf{c}'$ with the fast decoding algorithm for C to obtain $\mathbf{m}'$.
3. Compute $\mathbf{e} = P^{-1}\boldsymbol{e}'$.
4. Represent $\boldsymbol{e}$ as a message $\boldsymbol{m}$.

1.4.3 Another Method for Public Key Encryption

There is another proposed model for public key encryption that has recently been proposed. More precisely it is called a KEM (Key Encapsulation Mechanism). This is not a cryptosystem, but rather a model that is used in many recent cryptosystems. We will describe some of the cryptosystems with this model in chapter 4, when we talk about the code-based schemes that are in the semi-final stage of the NIST competition. Briefly, KEM consists of the following algorithms:

- Key Generation: takes as an input security parameters 1^λ and outputs the private encapsulation key **sk** and public encapsulation key **pk**.
- Encapsulation (Encap): The encapsulation algorithm takes as an input a public key **pk** and outputs an encapsulated key K and a ciphertext (K, C).
- Decapsulation (Decap): The decapsulation algorithm takes as an input the ciphertext C and the decapsulation key **sk**. The output is a key K encapsulated in C or a decapsulation failure symbol $\perp$ [5].

Chapter 2

Summary of Scrambling Processes in the Literature

In this chapter we will summarize some of the resent cryptosystems that have modified the scrambling process from the ones used in the McEliece Cryptosystem and its early variants.

2.1 Baldi's Twist

This cryptosystem is another variation of the McEliece cryptosystem and was introduced in 2016 by Baldi et al. in [10]. The main idea with this scrambling method is to replace the permutation matrix P with a denser transformation matrix Q as described in the McEliece cryptosystem. One of the advantages of doing this is if the permutation matrix is replaced with a denser transformation matrix than the the public key is no longer permutation equivalent to our message $\boldsymbol{m}$. As a result, this provides enhanced security of the public key. The transformation matrix Q is constructed in a way that ensures that the density of the public code is increased so that it is extremely difficult for an attacker to look for low weight codewords in the dual code, which is a common type of attack on the McEliece Cryptosystem. Also, the transformation matrix is constructed in a way that still limits the number of intentional errors.

We will now describe how this transformation matrix is constructed. The transformation matrix Q is a non singular $n \times n$ matrix and is of the form $Q = R + T$ where R is a dense matrix and T is a sparse matrix. Now we will describe the construction of R and T. First R is made by starting with two sets A and B. Each set consists of w matrices with size $z \times n$ where $z \leq n$. That is, $A = \{a_1, a_2, ...a_w\}$ and $b = \{b_1, b_2, ..., b_w\}$ where $a = \sum_{i=1}^{w} a_i$. Each of the w matrices are randomly chosen and then we define R as follows:

$$R = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ . \\ . \\ . \\ a_w \end{bmatrix}^T \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ . \\ . \\ . \\ b_w \end{bmatrix}$$

T is constructed in the following way. Define T to be a $n \times n$ non-singular sparse matrix with elements in $\mathbb{Q}$ and average row and weight equal to $m < n$. If m is an integer, then T is simply the sum of m generalized permutation matrices. But if m is rational, we make sure that the rows are columns of T are equal to $\lfloor m \rfloor$ or $\lceil m \rceil$ [10]. Below is the outline of the basic structure of this new modified McEliece cryptosystem.

Key Generation

1. Public Key: G' and t where $G' = S^{-1}GQ^{-1}$
2. Private Key: G, S, Q
3. Plaintext: $\mathbf{m} \in \mathbb{F}_2^k$, the set of all k-bit vectors

Encryption

1. 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 where $w_H(\mathbf{e}) \leq t$.

Decryption

1. Compute $\mathbf{c}' = \mathbf{c}Q = \boldsymbol{m}S^{-1} + \boldsymbol{e} \cdot Q$
2. Decode $\boldsymbol{c'}$ with the fast decoding algorithm for C to obtain $\boldsymbol{m'}$.
3. Recover $\boldsymbol{m'} = \boldsymbol{m}S^{-1}$.

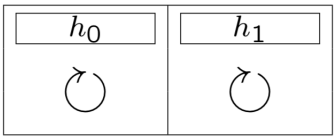
There is also a Niederreiter version of this proposed cryptosystem.

2.2 QC-MDPC

Another variation is to use Quasi Cyclic Mid Density Parity Check (QC-MDPC) codes with the McEliece framework as proposed by Nicolas Sendrier. Sendrier proposed using Quasi Cyclic codes with the McEliece framework. As mentioned earlier, the generator matrix for QC-MDPC codes are composed of circulant blocks [32]. Because G is in systematic form as well as composed of circulant blocks, this leads to a smaller key size and an easier encryption. The parameters of this cryptosystem are n, k, w, t. Now we will describe the key generation where $n = 2p$ and $k = p$.

Key Generation

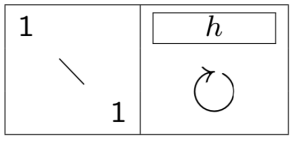
First, pick a sparse vector $(h_0, h_1) \in \{0,1\}^p \times \{0,1\}^p$ of weight w. The secret key H is obtained by taking cyclic shifts of the first row as seen below.

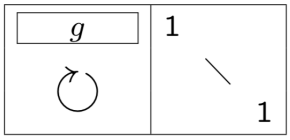
$H_{\mathsf{secret}} =$ 

where $h_0(x)$ is invertible in $F_2[x]/(x^p - 1)$.

Then we make known $h(x) = h_1(x)h_0^{-1}(x) \mathrm{mod}(x^p - 1)$ or $g(x) = \overline{h(x) \backslash x}$

Then the parity check matrix is H is a circular matrix constructed via sparse vectors h_0 and h_1 of a given weight and G is the generator matrix as seen below.

$H =$ 

$G =$ 

Encryption

Encryption is done as follows

$$F_2[x]/(x^p-1) \to F_2[x]/(x^p-1) \times F_2[x]/(x^p-1)$$
$$m(x) \to (m(x)g(x)+e_0(x), m(x)+e_1(x)).$$

The error $e(x) = (e_0(x), e_1(x)$ has weight t.

Decryption

Decryption only requires the sparse parity check matrix [32]. Iterative decoding is done like that of Low Density Parity Check Codes.

Essentially the McEliece framework is used with H_{secret}. That is, define G and H as above. The cryptosystem is more secure as long as two conditions hold. The first condition is the pseudorandomness of the public code and the second condition is the hardness of decoding QC codes [32]. This cryptosystem is generating a considerable interest due to its very little structure.

2.3 A New Direction in Scrambling

Aguilar-Melchor et al. introduced a new direction in scrambling in code-based cryptography in [1]. In the past, typically scrambling has only been done on the generator matrix or its variations. In this cryptosystem the scrambling is done on the ciphertext. The cryptosystem they are proposing is based on two different types of codes. The first code is a linear code C with parameters n and k. For this code C an efficient decoding algorithm is known. Examples of these would be Hamming codes, Goppa codes, BCH codes, etc. The code C will be publicly known along with its generator matrix G. The second code used in the cryptosystem is a random double circulant code. Aguilar-Melchor et al. also give the following definition that is essential in the design of the cryptosystem [1].

Definition 2.3.1. Let $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{F}^n$. The *circulant matrix* for a vector x is defined as follows,

$$\mathbf{rot(x)} = \begin{pmatrix} x_1 & x_n & ... & x_2 \\ x_2 & x_1 & ... & x_3 \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ x_n & x_{n-1} & \dots & x_1 \end{pmatrix}$$

Then for the second code in the cryptosystem, the generator matrix is of the form $H = (I_n \mid rot(\boldsymbol{h}))$. Also, for this cryptosystem the following operation is defined.

$$\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x} \times rot(\boldsymbol{y})^T = (rot(\boldsymbol{x}) \times \boldsymbol{y}^T)^T = \boldsymbol{y} \times rot(\boldsymbol{x})^T = \boldsymbol{y} \cdot \boldsymbol{x}.$$

As we can see in the above, the operation is commutative. It turns out that associativity of this operation plays an important role in the implementation of the cryptosystem. The proof of this was not given in the paper. So we give the proof of this property:

Theorem 2.3.2. *This operation $\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x} \times rot(\boldsymbol{y})^T$ is associative.*

Proof. Let

$$\boldsymbol{x} = \left(x_1, x_2, x_3, \cdots, x_n\right), \boldsymbol{y} = \left(y_1, y_2, y_3, \ldots, y_n\right) \text{ and } \boldsymbol{z} = \left(z_1, z_2, z_3, \ldots, z_n\right).$$

Then

$$\mathbf{rot(y)} = \begin{pmatrix} y_1 & y_n & \ldots & y_2 \\ y_2 & y_1 & \ldots & y_3 \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ y_n & y_{n-1} & \ldots & y_1 \end{pmatrix} \text{ and } \mathbf{rot(z)} = \begin{pmatrix} z_1 & z_n & \ldots & z_2 \\ z_2 & z_1 & \ldots & z_3 \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ z_n & z_{n-1} & \ldots & z_1 \end{pmatrix}.$$

Notice that $\mathbf{rot(y)}^T \times \mathbf{rot(z)}^T = \mathbf{rot(z)}^T \times \mathbf{rot(y)}^T$ due to its circulant nature. Then, we have

$$(\mathbf{x} \cdot \mathbf{y}) \cdot \mathbf{z} = (\mathbf{x} \times \mathbf{rot(y)}^T) \times \mathbf{rot(z)}^T = (\mathbf{x} \times \mathbf{rot(z)}^T) \times \mathbf{rot(y)}^T = (\mathbf{x} \times \mathbf{rot(z)}^T) \cdot \mathbf{y} = (\mathbf{x} \cdot \mathbf{z}) \cdot \mathbf{y}.$$

□

Now, we will describe the encryption and decryption process. Our first code C is a decodable $[n, k]$ code that can correct up to δ errors and a random double-circulant $[2n, n]$ code. The global parameters are (n, k, w, w_r, w_e).

Key Generation

1. Let $\boldsymbol{h} \in \mathcal{R}$ and a generator matrix G of C.
2. The secret key is $(\mathbf{x}, \mathbf{y}) \in \mathcal{R}^2$ where $w_{\mathbf{x}} = w_{\mathbf{y}} = w$.
3. The private key is $(\boldsymbol{h}, \boldsymbol{s})$ where $\boldsymbol{s} = \boldsymbol{x} + \boldsymbol{h} \cdot \boldsymbol{y}$ and $\boldsymbol{h}$ is sampled from $\mathcal{R}$.

Encryption

1. Let $\boldsymbol{e} \in \mathcal{R}, \boldsymbol{r} = (\boldsymbol{r_1}, \boldsymbol{r_2}) \in \mathcal{R}^2$ where $w(e) = w_e$ and $w(\boldsymbol{r_1}) = w(\boldsymbol{r_2}) = w_{\boldsymbol{r}}$.
2. Let $\boldsymbol{u} = \boldsymbol{r_1} + \boldsymbol{h} \cdot \boldsymbol{r_2}$ and $\boldsymbol{v} = \boldsymbol{m}G + \boldsymbol{s} \cdot \boldsymbol{r_2} + \boldsymbol{e}$.
3. The ciphertext then is $(\boldsymbol{u}, \boldsymbol{v})$.

Decryption

1. Decode $\mathbf{v} - \mathbf{u} \cdot \mathbf{y}$ by a special decoding algorithm.

The decoding alogrithm $C.decode$ only decodes correctly whenever

$$w(\boldsymbol{s} \cdot \boldsymbol{r_2} - \boldsymbol{u} \cdot \boldsymbol{y} + \boldsymbol{e}) = w((\boldsymbol{x} + \boldsymbol{h} \cdot \boldsymbol{y})\boldsymbol{r_2} - (\boldsymbol{r_1} + \boldsymbol{h} \cdot \boldsymbol{r_2})\boldsymbol{y} + \boldsymbol{e} = w(\boldsymbol{x} \cdot \boldsymbol{r_2} - \boldsymbol{r_1} \cdot \boldsymbol{y} + \boldsymbol{e}) \leq \delta.$$

So, instead of our error vector just being $\mathbf{e}$, the added error is $\boldsymbol{e}' = \boldsymbol{x} \cdot \boldsymbol{r_2} - \boldsymbol{r_s} \cdot \boldsymbol{y} + \boldsymbol{e}$. In this cryptosystem the first thing we notice is that our generator matrix G is known. So, the security and the ability to decrypt is not reliant on knowing C. The purpose of the random circualant code is to just generate noise. The security relies on the random circulant code being able to hide the structure of the code [1], while the ability to decrypt correctly is guaranteed by the linear code C.

We will now do an example of this cryptosystem with a BCH code.

Example 2.3.3. Let $n = 15$. This is a binary BCH $[15, 7, 5]$-code. Let α be a primitive root of F_{16} such that α is a root of $x^4 + x + 1$. The generator polynomial for the 2-error correcting BCH code is $(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$. We have that the parity check matrix for this code is given by

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

and a generator matrix given by

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

This is a 2-error correcting code. We will show that this in fact true. Let our message $\boldsymbol{m} = (0, 1, 0, 0, 1, 1, 1)$. Then we have $\boldsymbol{m}G = (0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1)$. We will let

$$\boldsymbol{x} = (1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1) \text{ and } \boldsymbol{y} = (0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0).$$

Notice $w(\mathbf{x}) = w(\mathbf{y}) = 5$. Now we let

$$\boldsymbol{h} = (0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1).$$

Notice that $w(\boldsymbol{h}) = 7$. We calculate $\boldsymbol{h} \cdot \boldsymbol{y}$ to get,

$$\boldsymbol{h} \cdot \boldsymbol{y} = (1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0).$$

Then

$$\boldsymbol{s} = \boldsymbol{x} + \boldsymbol{h} \cdot \boldsymbol{y} = (0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1)$$

So the secret key $\mathbf{sk} = (\mathbf{x}, \mathbf{y})$ and the private key $\mathbf{pk} = (\mathbf{h}, \mathbf{s})$. Now we let

$$\boldsymbol{r_1} = (0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1) \text{ and } \boldsymbol{r_2} = (1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1)$$

Also, we choose the error to be

$$\boldsymbol{e} = (1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0)$$

Next we calculate $\boldsymbol{u}$ and $\boldsymbol{v}$ where $\boldsymbol{u} = \boldsymbol{r_1} + \boldsymbol{h} \cdot \boldsymbol{r_2}$ and $\boldsymbol{v} = \boldsymbol{m}G + \boldsymbol{s} \cdot \boldsymbol{r_2} + \boldsymbol{e}$ and get the following,

$$\boldsymbol{u} = (1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1) \text{ and } \boldsymbol{v} = (1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1)$$

Then the ciphertext is $\mathbf{c} = (\mathbf{u}, \mathbf{v})$. Now when we will decode $\boldsymbol{v} - \boldsymbol{u} \cdot \boldsymbol{y}$ which is

$$\boldsymbol{v} - \boldsymbol{u} \cdot \boldsymbol{y} = (0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1).$$

i	α^i
0	1
1	α
2	α^2
3	α^3
4	$\alpha + 1$
5	$\alpha^2 + \alpha$
6	$\alpha^3 + \alpha^2$
7	$\alpha^3 + \alpha + 1$

i	α^i
8	$\alpha^2 + 1$
9	$\alpha^3 + \alpha$
10	$\alpha^2 + \alpha + 1$
11	$\alpha^3 + \alpha^2 + \alpha$
12	$\alpha^3 + \alpha^2 + \alpha + 1$
13	$\alpha^3 + \alpha^2 + 1$
14	$\alpha^3 + 1$

Table 2.1: Values of α^i.

We will follow the algorithm for decoding the BCH code as described above. We will look at $\boldsymbol{v} - \boldsymbol{u} \cdot \boldsymbol{y}$ which corresponds to the polynomial

$$\boldsymbol{w}(x) = x + x^4 + x^6 + x^7 + x^9 + x^{10} + x^{11} + x^{13} + x^{14}.$$

We calculate the syndromes and get the following,

$$S_1 = w(\alpha) = \alpha + \alpha^4 + \alpha^6 + \alpha^7 + \alpha^9 + \alpha^{10} + \alpha^{11} + \alpha^{13} + \alpha^{14} = 1$$

$$S_2 = w(\alpha^2) = \alpha^2 + \alpha^8 + \alpha^{12} + \alpha^{18} + \alpha^{20} + \alpha^{22} + \alpha^{26} + \alpha^{28} = 1$$

$$S_3 = w(\alpha^3) = \alpha^3 + \alpha^{12} + \alpha^{18} + \alpha^{21} + \alpha^{27} + \alpha^{30} + \alpha^{33} + \alpha^{39} + \alpha^{42} = 0$$

Now we calculate $XY = S_2 - \frac{S_1}{S_3} = 1$. The polynomial we are then factoring for is $z^2 + z + 1$. The zeroes of this polynomial are α^5 and α^{10}. So the received word has errors in position 5 and position 10, which it does. So the code is 2-error correcting. Thus, decoding was done correctly.

The structure this cryptosystem follows was the motivation for what we attempted to do in our research. In the next chapter we will describe an attempt at our proposed cryptosystem and just like [1] the scrambling that is done is on the ciphertext.

Chapter 3

A Framework for Future Constructions

The scrambling process used in [1] has brought a new direction to the construction of code-based cryptosystems. The idea that the structure of the code does not have to be hidden and that the scrambling is done on the ciphertext is a novel idea that has many potential advantages. The main tool that is used in this cryptosystem is an operation that sends vectors to other vectors. This operation turns out to have many properties, which turn out to be essential for the scheme to work properly. We believe that Linear Algebra would provide a rich source to find many new operations for potential future schemes instead of the $\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x} \times rot(\boldsymbol{y})^T$ operation that was used in [1]. In this chapter, we consider an attempt of finding such an operation and we analyze why the attempt was not successful, which leads us to build a framework for such operations that would successfully lead to construction of new schemes that might have better properties than the ones already built.

3.1 Our Attempt

The properties that we were able to discern at the beginning were that the operation should take two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_2^n$ and send it to another vector in $\mathbb{F}_2^n$, and that it has to be commutative.

The operation we initially proposed is the following. Define "$\cdot$" from $\mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$, by

$$\boldsymbol{u} \cdot \boldsymbol{v} = [L(\boldsymbol{u})^T L(\boldsymbol{v}) + L(\boldsymbol{v})^T L(\boldsymbol{u})](\boldsymbol{u} + \boldsymbol{v})^T.$$

where $L(u)$ is a given linear transformation from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Notice that our linear operation is commutative. That is, $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{v} \cdot \boldsymbol{u}$. So with the operation we thus proposed, the linear transformation can vary, which will increase the security of the cryptosystem as it is another added unknown. Some examples of a linear transformation we could define are the following:

Example 3.1.1. Let $\boldsymbol{u} = \big(1, 0, 1, 1\big)$. This is a vector in $\mathbb{F}_2^4$. We could let $L_1(\boldsymbol{u}) = \big(1, 1, 0, 1\big)$. In this example, we are permuting the second and third coordinates.

Example 3.1.2. Let $\boldsymbol{u} = (1, 0, 1, 1)$. We let L_2 be the transformation that permutes the third and fourth coordinate. So, $L_2(\boldsymbol{u}) = (1, 0, 1, 1)$. Notice this leaves the vector unchanged.

We were projecting this new operation to replace the operation in the cryptosystem proposed by Aguilar-Melchor. However, while implementing our cryptosystem structure on an example with BCH code of length $n = 15$, we found that our proposed operation is not associative. This is a crucial requirement that we must have of the operation. Otherwise, the cryptosystem will not decode properly. The reasoning for this is because as we saw above, C.decode (i.e., the decoding algorithm for C) will decode correctly whenever

$$\boldsymbol{w}(\boldsymbol{s} \cdot \boldsymbol{r_2} - \boldsymbol{u} \cdot \boldsymbol{y} + \boldsymbol{e}) = w((\boldsymbol{x} + \boldsymbol{h} \cdot \boldsymbol{y})\boldsymbol{r_2} - (\boldsymbol{r_1} + \boldsymbol{h} \cdot \boldsymbol{r_2})\boldsymbol{y} + \boldsymbol{e} = w(\boldsymbol{x} \cdot \boldsymbol{r_2} - \boldsymbol{r_1} \cdot \boldsymbol{y} + \boldsymbol{e}) \leq \delta.$$

But, if the operation is not associative, then $(\boldsymbol{h} \cdot \boldsymbol{y}) \cdot \boldsymbol{r_2} \neq (\boldsymbol{h} \cdot \boldsymbol{r_2}) \cdot \boldsymbol{y}$. So the cryptosystem will not decode properly. Thus, our attempt at defining a new operation does not work entirely. However, we are able to build a framework for the operations that will be useful for consrtcuting such cryptosystems, which we described in the next section.

3.1.1 Requirements of a New Operation

Our proposal was to essentially define a new operation to generalize the cryptosystem as proposed by [1]. Based off our research we believe the operation should have the following properties:

1. This new operation should be commutative.
2. The new operation should be associative on the vectors.
3. Some type of linear transformation should be done on the vector.
4. The operation should be generalized.

However notice that the proposed operation cannot be just a linear combination of vectors. If the operation is a linear combination on the vectors, then the operation will not be commutative. We believe that defining a new operation in this way will allow us to generalize a method to scrambling the ciphertext. The purpose of defining a more general operation is that a different linear transformation on the vector $\boldsymbol{v}$ will correspond to a different operation and a new cryptosystem. This operation will create a more generalized cryptosystem and a more secure one.

3.1.2 Structure of the Cryptosystem With the "New" Operation

The cryptosystem is described in the following way, the global parameters are (n, k, w, w_r, w_e) and let $\cdot$ be our new operation described above.

Key Generation

1. Let $\boldsymbol{h} \in \mathcal{R}$, a generator matrix G of C.
2. The secret key is $(\mathbf{x}, \mathbf{y}) \in \mathcal{R}^2$ where $w_{\mathbf{x}} = w_{\mathbf{y}} = w$.
3. The private key is $(\boldsymbol{h}, \boldsymbol{s})$ where $\boldsymbol{s} = \boldsymbol{x} + \boldsymbol{h} \cdot \boldsymbol{y}$.

Encryption

1. Let $\boldsymbol{e} \in \mathcal{R}, \boldsymbol{r} = (\boldsymbol{r_1}, \boldsymbol{r_2}) \in \mathcal{R}^2$ where $w(\boldsymbol{e}) = w_{\boldsymbol{e}}$ and $\boldsymbol{w}(\boldsymbol{r_1}) = w(\boldsymbol{r_2}) = w_{\boldsymbol{r}}$.
2. Let $\boldsymbol{u} = \boldsymbol{r_1} + \boldsymbol{h} \cdot \boldsymbol{r_2}$ and $\boldsymbol{v} = \boldsymbol{m}G + \boldsymbol{s} \cdot \boldsymbol{r_2} + \mathbf{e}$.
3. The cipher text then is $(\mathbf{u}, \mathbf{v})$.

Decryption

1. Decode $\mathbf{v} - \mathbf{u} \cdot \mathbf{y}$ by a special decoding algorithm.

Chapter 4

Scrambling Processes for Recent Cryptosystems

In this chapter we will be revisiting some of the code-based cryptosystems we have considered in the previous chapters in terms of the scrambling processes they use but our emphasis will be on the most recent variants that are part of the NIST competition for post-quantum cryptography.

4.1 McEliece

The original McEliece Cryptosystem is very well known. The design of the cryptosystem is described above. The scrambling that is done is on the generator matrix G. The generator matrix G is disguised so that no one can know what it is. The public instead is given a new matrix G' which is constructed from multiplication by a scrambler matrix S and a permutation matrix P. The result is a matrix G' which is permutation equivalent to G. Some of the advantages of the McEliece Cryptosystem are its fast encryption and decryption rate so it has fast implementation. Another advantage of the cryptosystem is that there is an inherent randomness in the encryption process. This increases the security of the cryptosystem. One of the cons of the McEliece cryptosystem is that the public key size is too large. The public key is a $k \times n$ matrix, which can be hard to store as n increases. Thus, the cryptosystem is hard to implement in the real world. Notice that in this process the McEliece cryptosystem produces an equivalent code because of this it may be easier for an attacker to attack the public key and figure out the code. Another disadvantage is that the ciphertext is much larger than the original message because of the redundancy added by the encoding process [14].

4.2 Baldi

The description of this cryptosystem is described above. Similar to McEliece the scrambling is also being done on the generator matrix. But unlike McEliece the permutation matrix is replaced with a denser transformation matrix. One of the advantages of this is that the public key is no longer permutation equivalent to the secret code. This increases the security of the cryptosystem. Now an attacker can no longer exploit the equivalence structure to get the secret code [10].

4.3 QC-MDPC

We will look at the scrambling technique of the code described above. This proposed cryptosystem is built on QC-MDPC codes. The scrambling is done on the private code. In [32] the private code structure is hidden by calculating the inversion of one of the cyclic blocks (namely $h_0(x)$) and multiplying it by the $h_1(x)$ to get systematic form. Essentially, the scrambling is done by attempting to normalize the matrix and putting it in systematic form. Unscrambling can only be done if you were to know what h_0 is [32].

4.4 Scrambling Techniques Used in Most Recent Variants

As mentioned earlier, quantum computers have had a huge impact on the future of cryptosystems. Due to their large computing capacity as well as the presence of special algorithms that can be implemented on them, quantum computers have the ability to solve some of the difficult mathematical problems that have formed the basis for many information security applications. Because of this, if quantum computers are built large enough, they will have the ability to break many of the cryptosystems that are in place today that help keep our data secure. Post quantum cryptography is the study of developing cryptosystems that will be secure enough against both quantum and regular computers. The National Institute of Standards and Technology (NIST) has published a report on the current state of post quantum cryptography in 2016. In the same year, they decided to host a competition that focuses on standardizing the public key cryptosystems that will be resistant to quantum computation. The competition consists of three rounds. Initially, 26 algorithms were proposed. Today, only seven have moved on to the third and final round. We will examine the main algorithms used in each of the code-based finalists, and especially look at the scrambling techniques they are proposing [29].

4.4.1 HQC

HQC stands for Hamming Quasi Cyclic. This public key cryptosystem is the one proposed by [1] that uses two different codes, a code C with an efficient decoding algorithm and a random double circulant code. The scrambling in this cryptosystem is done on the cipher text. They do this by defining an new operation $(\cdot)$. One of the key advantages of this cryptosystem is that we are not limited by what code C can be used. So they can use different codes in the cryptosystem that may have been difficult to hide before, but they are known to be good for decoding. Another advantage of this cryptosystem is that it is general and can be adapted and modified by using different codes, different operation, or even a different metric to create a more secure cryptosystem [3].

4.4.2 Bike

Bike is a set of algorithms for key encapsulation that are based on Quasi-cyclic moderate density parity check codes. There are three different variants of Bike. The cryptosystem is similar to one proposed by [32]. We will look at Bike 1 [13] [6] [18] [19] [27].
In this cryptosystem, we let $\mathbf{K} : \{0,1\}^n \to \{0,1\}^{l_K}$ be a hash function in encryption and decryption, where l_K is the desired symmetric key length. Next we describe the resulting cryptosystem.

Key Generation

Let λ be the target security level. We define the following parameters r and w. Let r be a prime such that $(X^r - 1)/(X - 1) \in \mathbb{F}_2[x]$ is irreducible. Let w be the weight of the rows of parity check matrix H. So we do the following steps:

1. Generate $\boldsymbol{h_0}, \boldsymbol{h_1} \in \mathcal{R}$ both of odd weight and $|\boldsymbol{h_0}| = |\boldsymbol{h_1}| = \frac{\boldsymbol{w}}{2}$.
2. Generate $\boldsymbol{g} \in \mathcal{R}$ where $\boldsymbol{g}$ has odd weight and $|\boldsymbol{g}| \approx \frac{\boldsymbol{r}}{2}$.
3. Compute $(\boldsymbol{f_0}, \boldsymbol{f_1}) = (\boldsymbol{gh_1}, \boldsymbol{gh_0})$.

Encapsulation

1. Let $(\boldsymbol{e_0}, \boldsymbol{e_1}) \in \mathcal{R}^2$ so that $|\boldsymbol{e_0}| = |\boldsymbol{e_1}| = t$.
2. Now generate $\boldsymbol{m} \in \mathcal{R}$ where $\boldsymbol{m}$ is our message.
3. Calculate the cipher text $\boldsymbol{c} = (\boldsymbol{c_0}, \boldsymbol{c_1}) = (\boldsymbol{mf_0} + \boldsymbol{e_0}, \boldsymbol{mf_1} + \boldsymbol{e_1})$.
4. Finally, compute $K \leftarrow \mathbf{K}(e_0, e_1)$

Decapsulation

1. First we break up $\boldsymbol{c}$ as $\boldsymbol{c_0}, \boldsymbol{c_1}$ and calculate the syndrome $\boldsymbol{s} = (\boldsymbol{c_0 h_0} + \boldsymbol{c_1 h_1})$.
2. We decode $\boldsymbol{s}$ to recover the error vector $\boldsymbol{e'_0}, \boldsymbol{e'_1}$.
3. If $(\boldsymbol{e'_0}, \boldsymbol{e'_1}) \neq t$ then the decoding has failed and we output the failure symbol.
4. Otherwise compute $K = \boldsymbol{K}(\boldsymbol{e'_0}, \boldsymbol{e'_1})$.

Now we will look the variant Bike 2. For this variation, instead of following the McEliece Framework, we are going to follow the same structure but with the Niederrieter framework. Now we describe the cryptosystem.

Key Generation

Let λ be the target security level. We define the following parameters r and w as above.

1. Generate $\boldsymbol{h_0}, \boldsymbol{h_1} \in \mathcal{R}$ both of odd weight and $|\boldsymbol{h_0}| = |\boldsymbol{h_1}| = \frac{w}{2}$.
2. Generate $\boldsymbol{g} \in \mathcal{R}$ where $\boldsymbol{g}$ has odd weight and $|\boldsymbol{g}| \approx \frac{r}{2}$.
3. Find $\boldsymbol{h} = (\boldsymbol{h_1 h_0^{-1}})$.

Encapsulation

1. Let $(\boldsymbol{e_0}, \boldsymbol{e_1}) \in \mathcal{R}^2$ so that $|\boldsymbol{e_0}| = |\boldsymbol{e_1}| = t$.
2. Now generate $\boldsymbol{m} \in \mathcal{R}$ where $\boldsymbol{m}$ is our message.
3. Calculate the cipher text $\boldsymbol{c} = \boldsymbol{e_0} + \boldsymbol{e_1 h}$
4. Finally, compute $K \leftarrow \mathbf{K}(e_0, e_1)$

Decapsulation

1. First we break up $\boldsymbol{c}$ as $\boldsymbol{c_0}, \boldsymbol{c_1}$ and calculate the syndrome $\boldsymbol{s} = (\boldsymbol{c_0 h_0})$
2. We decode $\boldsymbol{s}$ to recover the error vector $\boldsymbol{e'_0}, \boldsymbol{e'_1}$.
3. If $(\boldsymbol{e'_0}, \boldsymbol{e'_1}) \neq t$ then the decoding has failed and we output the failure symbol.
4. Otherwise compute the following $K = \boldsymbol{K}(\boldsymbol{e'_0}, \boldsymbol{e'_1})$.

Scrambling Techniques

The main scrambling technique done in the BIKE cryptosystem is that the private code is hidden by first multiplying a sparse private matrix by any random dense cyclic block of the parity check matrix H. One of the advantages of this cryptosystem is the use of Quasi Cyclic Codes. One of the disadvantages of this is the size of the public key is doubled because the public key does not contain an identity block anymore.

4.4.3 Ledacrypt

This cryptosystem is based on Low Density Parity Check codes. Low Density Parity Check codes have a sparse parity check matrix. There are two variants based on the McEliece and Niederreiter cryptosystems. We will describe each of these in detail. The first one we will describe is the Niederreiter cryptosystem. After, we will describe is the McEliece variant of the cryptosystem [12] [9] [11] [28].

Key Generation

Consider we have a QC-LDPC with codeword $n = rn_0$ and n_0 where $n_0 \in \{1, 2, 3, 4\}$ and r is a prime number such that $ord_r(2) = r - 1$. First we generate two random binary matrices that correspond to H and Q. H is a secret quasi-cyclic $r \times rn_0$ parity check matrix. Q is a $rn_0 \times rn_0$ quasi-cyclic sparse binary matrix. The matrix H is composed of $1 \times n_0$ circulant blocks that each have size $r \times r$. Also, each row and column should have a fixed number of odd elements. We denote this as d_v. So, H has the following structure:

$$H = [H_0, H_1, ..., H_{n_0-1}]$$

and $w(H_i) = d_v$ and $0 \leq iMn_0$.

Now we describe Q. Each block of Q will have a certain weight. This will then correspond to a circulant matrix of integers that we will denote w_Q. We must have that the matrix Q is invertible. So because of this, the possible weights of each block are restrictive. Next we compute the product $L = HQ$ where L has the same rank as H. So

$$L = [L_0, L_1, \ldots, L_{n_0-1}]$$

Next, the following matrix M is computed,

$$M = L_{n_0-1}^{-1} L = [M_0|M_1|M_2| \ldots |M_{n_0-2}|I_r] = [M_l|I]$$

In this case, I will be a $r \times r$ identity matrix. We let the Matrix M be the parity check matrix of the public code. The private key is (H, Q).

Encryption

We let the message $\boldsymbol{m}$ be a $1 \times rn_0$ binary vector $\boldsymbol{e}$. To get the syndrome $\boldsymbol{s}$ which is a $r \times 1$ vector we multiply the following,

$$\boldsymbol{s} = [M_0| \cdots \cdots M_j| \cdots |M_{n_0-2}|I]e^T.$$

The syndrome is then the ciphertext.

Decryption

First we will compute the following,

$$\boldsymbol{s'} = L_{n_0-1}\boldsymbol{s} = HQ\boldsymbol{e}^t = H(Q\boldsymbol{e}^T) = H(\boldsymbol{e}QT)^T.$$

The new error vector $\boldsymbol{e'} = \boldsymbol{e}Q^T$. Then apply the decoding algorithm for QC-LDPC on $\boldsymbol{s'}$ to recover $\boldsymbol{e'}$. Now we look at the McEliece version of Ledacrypt.

Key Generation

The Key Generation for the McEliece cryptosystem is the same as the Niederreiter cryptosystem.

Encryption

For the McEliece variant, we have a $1 \times r(n_0 - 1)$ vector $\boldsymbol{u}$ as the message and an error vector $\boldsymbol{e}$. The public key is is the Quasi cyclic generator matrix with size $(n_0 - 1) \times n_0$ so, the public key is

$$G = [Z|[M_0| \ldots |M_{n_0-2}]^T]$$

where Z is a diagonal block matrix that is composed of $n_0 - 1$ copies of the identity circulant block I. We do this so the generator matrix is in systematic form. The output is a ciphertext $\boldsymbol{c}$ that has an associated error. We compose $\boldsymbol{c}$ in the following way:

$$\boldsymbol{c} = [\boldsymbol{e}_0 \mid \cdots \mid e_{n_0-2} \mid e_{n_0-1}] + [u_0 \mid \cdots \mid u_{n_0-2} \mid \textstyle\sum_{j=0}^{n_0-2} u_j M_j].$$

Decryption

We compute $\boldsymbol{s} = L\boldsymbol{e}^T$. Apply the decoding algorithm for QC-LDPC codes on $\boldsymbol{s}$ to get back $\boldsymbol{u}$.

Scrambling Techniques

The scrambling done on the Ledacrypt cryptosystem is done by scrambling the parity check matrix and constructing it as described above.

4.4.4 Rollo

Rollo (Rank-Ouroboros, Lake, and Locker) are three different cryptosystems that are based on rank metric codes. These cryptosystems are referred to as Rollo-I, Rollo-II, and Rollo-III. For our purposes, we will only look at Rollo-II which is based in Public Key Encryption. One of the interesting properties about this cryptosystem are they have the same decoding procedures as Low Rank Parity Codes. We will define some necessary terms, describe the cryptosystems, as well as discuss the advantages and disadvantages of the scrambling process they employ [17] [7] [22] [4].

We will denote $S_w^n(\mathbb{F}_q^m)$ be the set of vectors of length n and rank weight w over $\mathbb{F}_q^m$. Similarly let $S_{1,w}^n(\mathbb{F}_q^m)$ be the set of vectors of length n and rank weight w over $\mathbb{F}_q^m$ such that its support contains 1. Let P be an irreducible polynomial of $F_q[x]$ of degree n and is one of the given parameters of the cryptosystem. The RSR (Rank Support Recover) algorithm is a decoding algorithm that is based on Low Rank Parity Codes. Note that $\oplus$ is the Bit XOR algorithm. The algorithm takes two inputs values (in this case two vectors of equal length) and compares each bit side by side and returns a 0 if both values are 0 or 1 and returns 1 otherwise. Now we will describe the cryptosystem.

Key Generation

1. Choose $(\mathbf{x}, \mathbf{y}) \in S_d^{2n}(\mathbb{F}_q^m)$.
2. Let $\boldsymbol{h} = \boldsymbol{x^{-1}y} \bmod P$.
3. The private key is $\mathbf{h}$ and the secret key $\mathbf{sk} = \mathbf{(x,y)}$.

Encryption

1. Choose $(\boldsymbol{e_1}, \boldsymbol{e_2}) \in S_r^{2n}(\mathbb{F}_q^m)$.
2. Let $E = \text{Supp}(\boldsymbol{e_1}, \boldsymbol{e_2})$.
3. Let $\mathbf{c} = \boldsymbol{e_1} + \boldsymbol{e_2 h} \bmod P$.
4. Next, compute cipher $= M \oplus$ Hash(E). The ciphertext is then $C = (\mathbf{c}, \text{cipher})$.

Decryption

1. Let $\mathbf{s} = \mathbf{xc} \bmod P$ and $F = \text{Supp}(\mathbf{x}, \mathbf{y})$.
2. Let $E \leftarrow \text{RSR}(F, \boldsymbol{s}, r)$.
3. Finally, return $M = \text{cipher} \oplus \text{Hash}(E)$.

Scrambling Techniques

The scrambling for this cryptosystem is done on the error vectors as well as on the ciphertext. As we can see in the above we first hide the message by performing a Hash function on the error and then combining this with an operator on the original message. We then have a two part ciphertext. The interesting thing about this method is that no generator matrix or parity check matrix is involved. The scrambling is done on the actual ciphertext itself. Also, there is a two part ciphertext which also helps hide the message. One of the disadvantages of this though is that the ciphertext size is doubled. However, because rank metric codes are hard to decode in general, this does not have a huge impact.

4.4.5 RQC

RQC (Rank Quasi Cyclic) is an encryption scheme based on two codes. The structure of this cryptosystem is very similar to HQC. The two codes that RQC use are ideal codes and Gabidulin codes. We will discuss the structure of this cryptosystem [8] [2]. For the setup, we look a the following:

Let $\mathcal{G}_{\mathbf{g}}(n, k, m)$ be a Gabidulin code with generator matrix G that is capable of correcting δ errors by its efficient decoding algorithm. As mentioned above the second code is a random ideal $[2n, n]$ code with parity check matrix $(\mathbf{1}, \mathbf{h})$. We will denote $S_w^n(\mathbb{F}_q^m)$ be the set of vectors of length n and rank weight w over $\mathbb{F}_q^m$ and similarly let $S_{1,w}^n(\mathbb{F}_q^m)$ be the set of vectors of length n and rank weight w over $\mathbb{F}_q^m$ such that its support contains 1. Now we look at the following:

Key Generation

1. Let $\mathbf{h} \in \mathbb{F}_{q^m}^n$.
2. Let $\mathbf{g} \in S_n^n(\mathbb{F}_q^m)$.
3. Let the secret key be $(\mathbf{x}, \mathbf{y}) \in S_{1,w}^{2n}(\mathbb{F}_q^m)$.
4. Let the private key $pk = (\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{s} = \boldsymbol{x} + \boldsymbol{h} \cdot \boldsymbol{y}) \mod P$.

Encryption

1. Generate $(\boldsymbol{e},\boldsymbol{r_1},\boldsymbol{r_2}) \in S_{w_r}^{3n}(\boldsymbol{F}_q^m)$.
2. Let $\boldsymbol{u} = \boldsymbol{r_1} + \boldsymbol{h} \cdot \boldsymbol{r_2} \mod P$.
3. Let $\boldsymbol{v} = \boldsymbol{m}G + \boldsymbol{s} \cdot \boldsymbol{r_2} + \boldsymbol{e} \mod P$. The ciphertext is $\boldsymbol{c} = (\boldsymbol{u}, \boldsymbol{v})$

Decryption

For decryption the decoding algorithm $\mathcal{G}_g$.Decode is going to decode $(\mathbf{v} - \mathbf{u} \cdot \mathbf{y}) \mod P$.

Scrambling Techniques

The scrambling on the cryptosystem is done on the ciphertext. It is essentially very similar to the one proposed by [1] but it is implemented with two different codes Ideal and Gaibuldin. The advantages of this are similar to the ones as described above in [1].

4.4.6 NTS KEM

The last encryption scheme we will discuss is NTS KEM. NTS KEM is a variant of the two cryptosystems McEliece and Niederietter (both public encryption schemes) but using a key encapsulation scheme. This cryptosystem is focused on the secure communication that goes along with the random key. Now we will introduce some notation and ideas that are essential in this cryptosystem. After we will discuss the key encapsulation, and key decapsulation. Finally, we discuss the scrambling techniques that were used [25] [24] [15] [14] [16] [35] [5] .

Definition 4.4.1. Let $\boldsymbol{v} \in \mathbb{F}^{n_1}$ and $\boldsymbol{w} \in \mathbb{F}^{n_2}$. We will denote $(\boldsymbol{v}|\boldsymbol{w})$ as the *concatenation* of vectors $\boldsymbol{v}$ and $\boldsymbol{w}$. Note, $(\boldsymbol{v}|\boldsymbol{w}) \in \mathbb{F}^{n_1+n_2}$.

Let $\boldsymbol{e}$ be a vector of length n. Then we can partition $\boldsymbol{e}$ into three sub-vectors by: $\boldsymbol{e} = (\boldsymbol{e}_a, \boldsymbol{e}_b, \boldsymbol{e}_c)$ where $\boldsymbol{e}_a \in \mathbb{F}^{k-l}, \boldsymbol{e}_b \in \mathbb{F}^1, \boldsymbol{e}_c \in \mathbb{F}^{n-k}$.

Definition 4.4.2. A *permutation* π is an ordered sequence of n elements. We can often represent the permutation as a matrix $\boldsymbol{P} \in \mathbb{F}_2^{n \times n}$ where there is exactly an entry of 1 in each row and column and zeroes everywhere else. Instead of a matrix, a permutation can also be represented by a *permutation vector* $\boldsymbol{p}$ where $\boldsymbol{p} = (p_0, p_1, ..., p_{n-1})$ where p_i is row of p that that has 1 at column i.

Now let $H_l(.)$ be a function which is a pseudo random bit generator that produces a l-bit binary string and a hash function is used to implement this generator. More details are provided in [5].

Parameters

The following parameters are needed:

1. $n = 2^m$.
2. τ : the number of errors corrected by the code.
3. $f(x)$ an irreducible polynomial of a degree m over $\mathbb{F}_2$.
4. $l = 256$ which is the length of the random key to be encapsulated.

Key Generation

Below are the steps:

1. Generate a monic Goppa polynomial of degree r. So we have

$$G(z) = g_0 + g_1 z + \cdots + g_{\tau-1} z^{\tau-1} + z^{\tau}.$$

   The polynomial $G(z)$ corresponds to a binary Goppa code of length $n = 2^m$, dimension $k = n - \tau m$.

2. Generate a permutation vector $\boldsymbol{p}$ of length n.

3. Generate a generator matrix of the form $\boldsymbol{G} = [\boldsymbol{I_k}|\boldsymbol{Q}]$ as follows:

   Let β be a root of $f(x)$ and B be a basis of $\mathbb{F}_{2^m}$ where $B = \langle \beta^{n-1}, \ldots, \beta, 1 \rangle$. Then the i-th element in $\mathbb{F}_{2^m}$ in the basis of B is defined as

$$B[i] = \{b_0\beta^{m-1} + b_1\beta^{m-2} + \cdots + b_{m-2}\beta + b_{m-1}\}.$$

   We let $\boldsymbol{a'} = (a_0, a_1, \ldots, a_{n-2}, a_{n-1}) = (B[0], B[1], \ldots, B[n-2], B[n-1])$. Let $\boldsymbol{a} = \pi_{\boldsymbol{p}}(\boldsymbol{a'}) = (a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}})$.

   Construct parity check matrix H_m as using the sequence $\boldsymbol{a}$ and Let $\boldsymbol{h} = (h_{p_0}, h_{p_1}, \ldots, h_{p_{n-1}})$ be the first row of H_m. See [5] to see how $\boldsymbol{h}$ is constructed. Let $B(a_i) = (b_{i0}, b_{i1}, \ldots, b_{i(m-1)})$ be a representation of a_i over $\mathbb{F}_2$, that is,

$$a_i = b_{i0} + b_{i1}\beta + b_{i2}\beta^2 + \cdots + b_{i(m-1)}\beta^{m-1}$$

where $b_{ij} \in \mathbb{F}_2$. We then replace each entry of H_m with $B(\cdot)^T$ to get $\boldsymbol{H}$. $B(\cdot)^T$ is described in more detail in [5].

Put $\boldsymbol{H}$ into reduced row echelon form. Let ρ denote the reordering of the columns of H. Apply the reordering to the vectors $\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{p}$ that is $\boldsymbol{a} = \rho(\boldsymbol{a})$, $\boldsymbol{h} = \rho(\boldsymbol{h})$, and $\boldsymbol{p} = \rho(\boldsymbol{p})$,

Let $\boldsymbol{H} = [\boldsymbol{Q}^T \mid \boldsymbol{I}_{n-k}]$ and so $G = [\boldsymbol{I}_k | \boldsymbol{Q}]$

4. Generate $\boldsymbol{z} \in F_2^l$ at random.
5. Partition $\boldsymbol{a}$ and $\boldsymbol{h}$ so $\boldsymbol{a} = (\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_c)$ and $h = (\boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c)$. Now we will let $\boldsymbol{a}^* = (\boldsymbol{a}_b, \boldsymbol{a}_c)$ and $\boldsymbol{h}^* = (\boldsymbol{h}_b, \boldsymbol{h}_c)$.

The public key is $(\boldsymbol{Q}, \tau, l)$ and the private key is $(\boldsymbol{a}^*, \boldsymbol{h}^*, \boldsymbol{p}, \boldsymbol{z}, pk)$.

Encapsulation

1. Partition $\mathbf{e}$
2. Compute $\boldsymbol{k}_e = H_l(\boldsymbol{e}) \in \mathbb{F}_2^l$.
3. Let $\boldsymbol{m} = (\boldsymbol{e}_a \mid \boldsymbol{k}_e) \in \mathbb{F}_2^k$
4. Let

$$\begin{aligned} c &= (\boldsymbol{m} \mid \boldsymbol{m} \cdot \boldsymbol{Q}) + \boldsymbol{e} \\ &= (\boldsymbol{e}_a \mid \boldsymbol{k}_e \mid (\boldsymbol{e}_a \mid \boldsymbol{k}_e) \cdot \boldsymbol{Q}) + (\boldsymbol{e}_a \mid \boldsymbol{e}_b + \boldsymbol{e}_c) \\ &= (\mathbf{0}_a \mid \boldsymbol{k}_e + \boldsymbol{e}_b \mid (\boldsymbol{e}_a \mid \boldsymbol{k}_e) \cdot \boldsymbol{Q} + \boldsymbol{e}_c) \\ &= (\mathbf{0}_a \mid \boldsymbol{c}_b \mid \boldsymbol{c}_c) \end{aligned}$$

5. Output: $(\mathbf{k_r}, \mathbf{c}^*)$

Decapsulation

1. Look at the vector $\boldsymbol{c} = (\mathbf{0}_a | \boldsymbol{c}_b, \boldsymbol{c}_c) \in \mathbb{F}_2^n$ and apply a decoding algorithm.
2. Compute the error vector $\boldsymbol{e} = \pi_{\boldsymbol{p}}(\boldsymbol{e'})$ and partition $\boldsymbol{e}$ into three parts.
3. Calculate $(\boldsymbol{k}_r, \boldsymbol{c'})$. Check to see that $\boldsymbol{c'} = \boldsymbol{c}^*$ and the hamming weight of $\boldsymbol{e}$ is τ. Return $\boldsymbol{k}_r$ if yes. Otherwise, return $H_l(\boldsymbol{z} \mid 1_a \mid 1_b \mid \boldsymbol{c}_c)$.

Scrambling Techniques

Several scrambling techniques are done on this cryptosystem. One scrambling method is disguising the parity check matrix by using the function $B(\cdot)^T$ as well as using a permutation to permute $\boldsymbol{a}$. The ciphertext is also scrambled by having a two part ciphertext that is the message itself as well as $m \cdot Q$. So in this cryptosystem they use both techniques we have seen in the past to create a more secure cryptosystem.

Chapter 5

Conclusion

Our research was aimed at finding a new scrambling technique that could be generalized for future cryptosystems. In the thesis we first looked at various scrambling techniques that are used in place today such as the ones used in [10], [32], and [1] to create a more secure cryptosystem. We then looked in more detail at the cryptosystem proposed by Aguilar and Melchor [1] and demonstrated their algorithm on an example. Our goal was to model a more generalized cryptosystem based on the scrambling technique that they proposed. We then described some of the challenges we had in developing this new cryptosystem. Through these challenges, we then discovered several conditions we believe a new cryptosystem should have in order to be secure, generalized, as well as efficient, thus building a framework for the mathematical requirements for such cryptosystems. Finally, we analyzed and discussed the various scrambling techniques that are being proposed in the NIST post quantum competition.

Finding different ways to scramble is an integral part of keeping a cryptosystem secure. This will be more evident in the future as we saw with the NIST cryptosystems. The NIST cryptosystems use a few different techniques to scramble such as scrambling the ciphertext, scrambling the parity check matrix, scrambling the generator matrix, etc. So, the different ways scrambling occurs in these NIST post quantum cryptography finalists is going to have a considerable impact on whether the data we transmit in our electronic based world will be secure or not in the future. This is why the methods that we can develop for scrambling cryptosystems play such a key role in code based cryptography.

5.1 Open Questions

Future work will include finding an operation for the proposed cryptosystem that follows the conditions of being commutative, associative, as well as generalizing the operation based on

linear transformations. In connection with such a cryptosystem, the concept of scrambling the structure seen further can come into picture. To this end, it will be of practical value to see if we can hide the linear transformation. A possible direction is to see if we can create a two part operation, one that is public and one that is private. These measures would make the cryptosystem much more secure than the current schemes that are in use and may potentially perform better.

Bibliography

[1] C. Aguilar-Melchor, O. Blazy, J. Deneuville, P. Gaborit, and G. Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Transactions on Information Theory*, 64(5):3927–3943, May 2018.

[2] Carlos Aguilar Melchor, Nicholas Aragon, Bettaieb Slim, Bidoux Loic, Blazy Olivier, Couvreur Alain, Deneuville Jean-Christophe, Gaborit Phillipe, Hauteville Adrien, and Zemor Gilles. Rank Quasi-Cyclic (RQC), 2019.

[3] Carlos Aguilar Melchor, Nicolas Aragon, Jean-Christophe Deneuville, Phillipe Gaborit, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Edoardo Perisichetti, and Gilles Zemor. Hamming Quasi-Cyclic (HQC), 2019.

[4] Carlos Aguilar Melchor, Nicolas Aragon, Bardet Magali, Bettaieb Slim, Bidoux Loic, Blazy Olivier, Deneuville Jean-Christophe, Gaborit Phillipe, Adrien Hauteville, Otmani Ayoub, Ruatta Olivier, Tillich Jean-Pierre, and Gilles Zemor. OLLO-Rank-Ouroboros, LAKE, LOCKER, 2019.

[5] Martin Albrecht, Carlos Cid, Kenneth Paterson, Jung Tjhai Cen, and Martin Tomlinson. NST-KEM Second Round Submission, 2019.

[6] Nicholas Aragon, Paulo Barreto, Bidoux Bettaieb, Slim, Blazy Olivier Loic, Gaborit Deneuville, Jean-Christophe, Guero Phillipe, Guneysu-Tim , Shay, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Perischetti, Nicholas Sendrier, Jean-Pierre Tillich, Vasseur Valetin, and Gilles Zemor. BIKE: Bit Flipping Key Encapsulation, 2019.

[7] Nicolas Aragon, Gaborit Philippe, Ruatta Olivier Hauteville Adrien, and Gilles Zemor. Low rank parity check codes: New decoding algorithms and application to cryptography. 2018.

[8] Loidreau Pierre Augot, Daniel and Robert Gwezheneg. Generalized Gabidulin codes over fields of any characteristic. 86(5):1807–1848, 2018.

[9] M Baldi. *QC-LDPC Code-Based Cryptography*. SpringerBriefs in Electrical and Computer Engineering. Springer International Publishing, 2014.

[10] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. Enhanced Public Key Security for the McEliece Cryptosystem. *Journal of Cryptology*, 29(1):1–27, jan 2016.

[11] 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.

[12] Marco Baldi, Franco Chiaraluce, and Marco Bianchi. Security and complexity of the McEliece cryptosystem based on quasi-cyclic low-density parity-check codes. *IET Information Security*, 7(3):212–220, sep 2013.

[13] Paulo S. L. M. Barreto, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, and Jean-Pierre Tillich. CAKE: Code-Based Algorithm for Key Encapsulation. pages 207–226. Springer, Cham, dec 2017.

[14] Anne Canteaut and Nicolas Sendrier. Cryptanalysis of the Original McEliece Cryptosystem. pages 187–199. Springer, Berlin, Heidelberg, oct 2000.

[15] Alexander Dent. A designer's guide to KEMs. *Lecture Notes in Computer Science*, 2898:133–151, 2013.

[16] Eiichrio Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, jan 2013.

[17] Philippe Gaborit, Gaétan Murat, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes and their application to cryptography. In *Proceedings of the Workshop on Coding and Cryptography WCC*, volume 2013, 2013.

[18] R.G Gallager. *Low Desnity Parity-Check Codes*. PhD thesis, M.I.T, 1964.

[19] Qian Guo, Thomas Johansson, and Paul Stankovski. A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors. pages 789–815. Springer, Berlin, Heidelberg, 2016.

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

[21] Yunghsiang S. Han. BCH Codes.

[22] Adrien Hauteville and Jean-Pierre Tillich. New algorithms for decoding in the rank metric and an attack on the LRPC crptosystem. 2015.

[23] Hans Christoph Hudde. Development and Evaluation of a Code-based Cryptography Library for Constrained Devices. Master's thesis, Ruhr-Universitat Bochum, 2013.

[24] P. J. Lee and E. F. Brickell. An Observation on the Security of McEliece's Public-Key Cryptosystem. In *Advances in Cryptology — EUROCRYPT '88*, pages 275–280. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.

[25] Yuan Xing Li, R H Deng, and Xin Mei Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273, jan 1994.

[26] S Ling and Xing C. *Coding Theory: A First Course*. Cambridge University Press, 2004.

[27] Gianluigi Liva and Hannes Bartz. Protograph-based Quasi-Cyclic MDPC Codes for McEliece Cryptosystems. *arXiv preprint arXiv:1801.07484*, 2018.

[28] Baldi Marco, Barenghi Alessandro, Gerardo Pelsoim Franco, Chiraluce and, and Paolo Santini. LEDAcrypt: Low-dEnsity parity-check coDe-bAsed cryptographic systems, 2020.

[29] NIST. Post-Quantum Cryptography Standardization: Call for Proposals, 12 2016.

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

[31] Marek Repka and Pavol Zajac. Overview of the McEliece cryptosystem and its security. *Tatra Mountains Mathematical Publications*, 60(1):57–83, 2014.

[32] Nicolas Sendrier. QC-MDPC-McEliece: A public-key code-based encryption scheme based on quasi-cyclic moderate density parity check codes. In *Workshop "Post-Quantum Cryptography: Recent Results and Trends"*, Fukuoka, Japan, nov 2014.

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

[34] Amin Shokrollahi. Decoding Binary BCH Codes, 2016.

[35] Falko Strenzke. *Efficeny and Implementation Security of Code-Based Cryptosystems*. PhD thesis, Technischen Universitat Darmstadt, 2013.