Spring 5-1-2015

Goppa Codes and Their Use in the McEliece Cryptosystems

Ashley Valentijn
Syracuse University

Goppa Codes and Their Use In The McEliece Cryptosystem

A Capstone Project Submitted in Partial Fulfillment of the Requirements of the Renée Crown University Honors Program at Syracuse University

Ashley Valentijn

Candidate for Bachelor of Science
and Renée Crown University Honors
May 2015

Honors Capstone Project in Mathematics

Capstone Project Advisor: _____
Professor Steven Diaz

Capstone Project Reader: _____
Professor Claudia Miller

Honors Director: _____
Stephen Kuusisto, Director

Date:

Abstract

We explore the topic of Goppa codes and how they are used in the McEliece Cryptosystem. We first cover basic terminology that is needed to understand the rest of the paper. Then we explore the definition and limitations of a Goppa code along with how such codes can be used in a general cryptosystem. Then we go in depth on the McEliece Cryptosystem in particular and explain how the security of this method works.

Executive Summary

Efficient and secure methods of communication have been in use since before the peak of the Roman Empire. A cryptosystem is a method of secure data transmission such that no one other than the intended receiver can read the original message. The problems that are usually faced with these methods vary, but include potential interception during transmission, errors, and whether or not a method is even practical to use. Errors can be detected and corrected by error-correcting codes such as Goppa codes, which are the original codes used in the McEliece cryptosystem. This paper explains Goppa codes and how they are used in the original McEliece cryptosystem, and investigates the security of the cryptosystem.

Sometimes a sent message and the received message are not the same. This is mainly due to the occurrence of errors. Errors can be caused by random noise over a channel, nearby channels, or outside interference among other things. Error-correcting code are used to detect or correct errors in a message. This is done by adding redundancy, that is, extra information that makes the message easier to understand if an error were to occur, in the message. For example, the message "I am going to bring my gloves" is sent, but somehow an error occurred and the received message is "I am going to bring my hairbrush." The receiver may not understand that the message has an error in it and also has no idea what the original message was supposed to be. Now suppose the message had been given some added redundancy so that it read, "Because it is snowing heavily outside, I am going to bring my gloves." Because of this added redundancy, the receiver could

correctly deduce that there was an error in the received message and that the sender had meant to say he or she was going to bring some sort of warm clothing to ward off the cold.

However, errors are not always added by accident. Errors can also be purposefully added to a message in order to further encrypt it. Then, by using an error-correcting algorithm, these errors can then be reversed. Adding an error vector further scrambles a message and as a result, can make the cryptosystem even more secure.

A Goppa code is a type of error-correcting code and is based on modular arithmetic, which is when a series of numbers increases towards a certain number and upon reaching said number, starts back over at 0 again. A real-life example of modular arithmetic is using a 24-hour clock versus a 12-hour clock to tell time. A 24-hour clock represents the hour in modulo 24 whereas the 12-hour clock represents the hour in modulo 12. For example, 3:00 A.M. is the same in a 12-hour clock and a 24-hour clock. However, 15:00 P.M. in a 24-hour clock is actually 3:00 P.M. in a 12-hour clock because the clock "wraps back around" so that any hour above 12, in this case 15, is reduced so that $15 - 12 = 3$.

The most basic cryptosystem consists of an algorithm for a key, an encryption method, and a decryption method. Suppose two friends, Alice and Bob, wish to securely communicate with each other, but know that a third party named Eve could potentially eavesdrop on them. Let us assume Alice wishes to send a message to Bob. Alice then uses a key to encrypt her message, known as plaintext, before sending it to Bob. Once Bob receives the encrypted plaintext, known as ciphertext, he can use the key to decrypt and discover the original message. We always assume Eve knows the general method that is being used, but not the key that is used. Eve usually has one of the following intentions:

read the message, uncover the key and use it to read all messages encrypted by said key, alter Alice's message, and/or pretend to be Alice and exchange messages with Bob.

The McEliece cryptosystem is a Public Key Cryptosystem, which means that it uses a public key and a private key in order to encrypt and decrypt a message. In our example, Bob publishes his public key so that anyone can read it. Alice uses Bob's public key in order to encrypt a message and send it to Bob. Bob then uses his private key in order to decrypt the message. However this is only useful if Eve cannot decode a message just by knowing a public key. Instead, Eve would need to know the private key, which is only known to the receiver of the ciphertext, which Bob is in this case.

Specifically, in the McEliece Cryptosystem, Bob would construct his public key by selecting a Goppa polynomial $g(z)$ of degree t and compute the generator matrix G of the Goppa code. Bob would then choose a random invertible matrix S and a random permutation matrix P and use them to compute $G' = SGP$. Bob would then publish his public key that consists of (G', t). His private key, which he would not publish, would consist of (S, G, P).

Alice would first write her message and represent it in binary strings of bits and then she would encrypt every string. Then she would choose a random error vector with weight of t or less, add it to the encryption, and then send the final encryption so that the sent code vector $y = mG' + e$.

Bob would then use his matrix P to compute $y' = yP^{-1}$. Then, Bob would use the decoding algorithm for his Goppa code G to error correct y' into the codeword $m' = mS$ by

finding e'. And because he already knows S^{-1}, he can calculate the original message $m = m'S^{-1}$.

Eve would have a difficult time trying to decrypt Alice's message without the private key. This is because she would need to separate matrix G from matrix G'. And because the matrix G' is not invertible, Eve would need to know the inverse of the chosen random matrix S, which was not published. Eve also does not know what the matrix P is and so cannot find y' in order to find m'. Basically, in order to keep this cryptosystem secure, it needs to be very laborious to decode y' and find m'. In order to do this, the Goppa code is selected to be as large as possible. For example, in the original McEliece cryptosystem published in 1978, McEliece suggested the use of a [1024, 524] Goppa code, i.e. a Goppa code of length 1024 and dimension 524. This Goppa code can correct for up to 50 errors. However, this is a major issue because the larger the code is, the less practical it is to use the cryptosystem. But as technology advances and memory capacity increases, this cryptosystem is likely to become more useful.

Another disadvantage of this system is that using the same encryption matrix G' to send the same message several times will make the system more vulnerable to attack. Also, there is no explicit way to use this cryptosystem for signatures, such as in the Rivest-Shamir-Adleman cryptosystem (RSA). This means that unless there was a password that was previously determined by the two friends, Bob has no idea whether or not Alice sent the message because anyone can use his public key to send him a message.

On the other hand, the advantages of this system are that the system is one of the more simple cryptosystems available and it has been widely studied since its introduction in 1978. Also, using this system allows for high encryption and decryption speeds.

Table of Contents

Part I: Basic Terminology **11**

1.1 Cryptology 11

1.2 Error-Correcting Codes 12

1.3 Fields 13

1.4 Modular Arithmetic 15

1.5 Binary 16

1.6 Hamming Distance and Weight 17

1.7 Linear Codes 19

Part II: Goppa Codes **20**

2.1 Definition of a Goppa Code 20

2.2 Parameters 21

2.3 Binary Goppa Codes 21

2.4 Parity Check Matrix 22

2.5 Encoding 23

2.6 Irreducible Binary Goppa Code Example 24

2.7 Error Correction 29

2.8 Decoding 30

Part III: The McEliece Cryptosystem **31**

3.1 Overview 31

3.2 Example 32

3.3 Attacking 35

3.4 Security 37

References **39**

Part I: Basic Terminology

1.1 Cryptology

Cryptology is the age-old study of how to send and receive messages without any outside sources interfering, reading, or altering the message in some way.

Let us consider two friends Alice and Bob. They wish to securely communicate with each other, but know a third party named Eve could potentially eavesdrop on them. Let us say Alice wants to send a message to Bob. Using a method, or cryptosystem, that was previously agreed upon between the two friends, Alice encrypts her message, also called the plaintext, into a series of codewords, which is referred to as the ciphertext. Usually, we assume Eve knows which cipher or cryptosystem is used and so the only thing keeping the message secure is the specific key or keys that are used in the encoding and decoding processes. Once Bob receives the codewords, he uses the key to decrypt the codewords back into Alice's original message. Eve usually has one of the following goals: read the message, discover the key and so read all messages encrypted with said key, alter Alice's message in such a way that Bob still thinks the message is from Alice, pretend to be Alice and exchange messages with Bob even though Bob believes he is talking to Eve and not Alice.

Depending on the cryptosystem, the method used to encode a message is not necessarily the same as the method used to decode a message. One example of this is the group of cryptosystems commonly called the Public Key Cryptosystems. These

cryptosystems are based on a public key and a private key. Bob publishes his public key so that anyone, Alice and Eve included, can read it. Alice then uses the public key in order to encode the message and sends it to Bob, who then uses his private key to decode the message. The method used, however, is such that Eve cannot decode the message just by knowing the public key. The private key is what is needed in order to decode a message, and as the receiver of the ciphertext, Bob is the only one who knows the private key.

1.2 Error Correcting Codes

Bob has received a message from Alice; however, the message does not make any sense. What happened? Well, either this was a crude attempt at misdirection from Eve or something happened during the transmission of the message that altered it. Errors can come from random noise over a channel, nearby channels, the fading of a channel, physical defects, or outside interference among other things.

Error correcting codes (ECC) are codes that can be used to detect errors in a sent message and correct them. This is done by adding redundancy, that is, extra information that makes the message easier to understand if an error were to occur, to the message. In coding theory, redundancy can also be called parity or parity check. This is called encoding. Once a message is encoded, it becomes a codeword that contains both the message and redundancy. These codes have specific decoding algorithms that can correct up to a specific amount of errors. Once this decoding algorithm is applied, some of the errors that may have occurred during the transmission of the message can be corrected, and the original message can be recovered. This process is called decoding.

Another way to use error-correcting codes is to purposefully add error to a message such that the received ciphertext is $y = c + e$, where c is the codeword and e is an error vector with a weight less than or equal to some fixed number r. This is helpful because Eve would have a hard time retrieving the codeword from the received ciphertext.

1.3 Fields

A field is a basic building block of mathematics as it allows us to assign certain properties to a set of numbers. These properties are necessary as otherwise it would be impossible to manipulate numbers the way we do.

Definition 1.3.1 A $field\ F$ is defined as a set of elements that is closed under two operations $(+,\cdot)$ and satisfies the following properties:

1. $There\ exists\ an\ element\ a \in F\ such\ that, for\ all\ x \in F,\ x + a = x.$
2. $There\ exists\ an\ element\ b \in F\ such\ that, for\ all\ x \in F, xb = x.$

$For\ all\ x, y, z \in Z,$

3. $x + y = y + x$
4. $xy = yx$
5. $(x + y) + z = x + (y + z)$
6. $(xy)z = x(yz)$
7. $x(y + z) = xy + xz$
8. $For\ each\ x, there\ exists\ an\ element - x\ such\ that\ x + (-x) = a.$
9. $For\ each\ x \neq a, there\ exists\ an\ element\ x^{-1}\ such\ that\ xx^{-1} = b.$

Example 1.3.1. The integers, written $\mathbb{Z} \coloneqq \{\dots -3, -2, -1, 0, 1, 2, 3, \dots\}$, are not a field under $(+,*)$. This is because other than the integers $1, -1$, and 0, the integers do not have elements that satisfy property number 9.

Example 1.3.2 The set of rational numbers $\mathbb{Q} \coloneqq \left\{\frac{a}{b} \; such \; that \; a, b \in \mathbb{Z} \; and \; b \neq 0\right\}$ is a field closed under $(+,\cdot)$ and as such, upholds the previous nine rules.

1. $y = \frac{0}{z} \; for \; any \; element \; z \; \neq 0 \; so \; that \frac{0}{z} + x = x.$
2. $y = \frac{1}{1} \; such \; that \frac{1}{1} \cdot x = x.$

$For \; all \; a, b, c, d \in \mathbb{Z},$

3. $\frac{a}{b} + \frac{c}{d} = \frac{c}{d} + \frac{a}{b}$
4. $\frac{a}{b} \cdot \frac{c}{d} = \frac{c}{d} \cdot \frac{a}{b}$
5. $\left(\frac{a}{b} + \frac{c}{d}\right) + \frac{e}{f} = \frac{a}{b} + \left(\frac{c}{d} + \frac{e}{f}\right)$
6. $\left(\frac{a}{b} \cdot \frac{c}{d}\right) \cdot \frac{e}{f} = \frac{a}{b} \cdot \left(\frac{c}{d} \cdot \frac{e}{f}\right)$
7. $\frac{a}{b} \cdot \left(\frac{c}{d} + \frac{e}{f}\right) = \frac{a}{b} \cdot \frac{c}{d} + \frac{a}{b} \cdot \frac{e}{f}$
8. $For \; each \frac{a}{b} \in \mathbb{Q}, one \; has \; \frac{-a}{b} + \frac{a}{b} = 0.$
9. $For \; each \frac{a}{b} \in \mathbb{Q}, with \; a \neq 0, the \; element \; \frac{b}{a} \; satisfies \; \frac{a}{b} \cdot \left(\frac{b}{a}\right) = 1.$

Definition 1.3.2 A field with a finite number of elements is called a $finite \; field$, or a $Galois \; field$. A Galois field is written as $GF \; (q)$ with q being the order of a field.

Theorem 1.3.1 Let p be prime. For every power p^m, there exists a unique finite field with the order p^m and these are the only possible finite fields.

Proof

See Artin [A, pages 510-515].

Definition 1.3.3 Let p be a prime number, $k > 0$, and $k \in \mathbb{Z}$. The Galois field of order $q = p^k$, that is, the amount of elements in the field, is called the *extension Galois field* of $GF(p)$ of degree m and is written as $GF(p^m)$.

Definition 1.3.4 A polynomial over $GF(p^m)$ is called *irreducible* if it is not divisible by any polynomial over $GF(p^m)$ with a lesser degree.

Example 1.3.3 Take $1 + X + X^3$ over $GF(2)$. Any polynomial with a degree less that 3 would need to contain X and/or X^2. No such polynomial completely divides $1 + X + X^3$, and so $1 + X + X^3$ is an irreducible polynomial over $GF(2)$.

Example 1.3.4 $X + X^5$ is not an irreducible polynomial over $GF(2)$ because

$\frac{X+X^5}{X} = 1 + X^4$.

Definition 1.3.5 Suppose we have an irreducible polynomial $f(x)$ with degree m over $GF(p)$. Such a polynomial is said to be primitive if $n = p^m - 1$ is the smallest possible integer for which $f(x)$ divides $X^n - 1$.

Example 1.3.5 Suppose $f(x) = 1 + X + X^3$ with degree 3 over $GF(2)$. $n = 2^3 - 1 = 7$ and so we have $X^7 - 1$. $X^7 - 1$ can be factored into irreducible polynomials as $(X + 1)(1 + X + X^3)(X^3 + X^2 + 1)$. It can be checked that $f(x)$ does not divide X^v such that $v < 7$. Therefore, $1 + X + X^3$ is a primitive polynomial of degree 3 over $GF(2)$.

1.4 Modular Arithmetic

Definition 1.4.1 For a positive integer n, integers a and b are said to be $congruent\ modulo\ n$, written $a \equiv b\ (mod\ n)$ if and only if $a - b = kn$ for some integer k.

Example 1.4.1 Write the number 49 congruent modulo 5.

5 divides 49-b such that the answer is an integer. According to definition, we need to find a k such that $49 - b = 5k$. But simply finding $5k$ is enough. Some multiples would be 45, 40, and 35, which would lead to $b = 4$,9, and 14 respectively.

Ideally, we want a b such that $0 \leq b < 5$. Such a b is called the smallest possible non-negative residue. And so the smallest possible non-negative residue is when $b = 4$. And so we would write, $49 \equiv 4\ (mod\ 5)$.

Example 1.4.2 Write $81 \equiv b(mod\ 2)$ such that b is the smallest possible non-negative residue.

$$81 - b = 2k$$

$$81 - b = 2(40)$$

$$b = 1$$

Therefore, $81 \equiv 1\ (mod\ 2)$.

Example 1.4.3 $GF(2) = \mathbb{Z}_2 = \{0,1\}$=the integers modulo 2. This is also known as the binary field.

1.5 Binary

For many situations involving computers, it has become accepted practice to convert information into a string of 1s and 0s. This conversion is referred to as writing information in binary. Usually, we think of numbers in base 10. For example, $321 = 3 * 10^2 + 2 * 10^1 + 1 * 10^0$. However, binary is written in base 2. This means numbers are converted by writing the number in additive terms of powers of 2, with the coefficients being either 0 or 1.Then, by taking the coefficients and putting them in order, we have a binary representation of the number. For example the number 37 can be written as $1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1$. As a result, 37 can be written as 100101 in binary.

Only the integers 0 and 1 are used in binary form and they have the additive properties of elements in modulo 2, that is, 0+1=1+0=1, 0+0=0, and 1+1=0.

Example 1.5.1 If we were to add 37+37 together in binary, then 100101+100101=1001010.

Each 0 or 1 is referred to as a bit. A string of bits is referred to as a byte and is defined to have a specific length. Using our previous example, 100101 is a 6-bit number.

1.6 Hamming Distance

Definition 1.6.1 Let A be an alphabet and let A_n: ={all sequences of length n of elements in A}. A code C of length n is a nonempty subset of A_n. An element in this subset, written as $c = (c_1, c_2, \dots, c_n)$, is called a code vector, legal codeword, or just a codeword. An illegal codeword is sequence of length n that is in A_n, but is not in a code.

Throughout this paper, we will assume alphabet A will consist of the binary bits 0 and 1. In other words, the alphabet is the field $\mathbb{Z}_2 = \{0,1\}$=the integers modulo 2. A code using this alphabet is referred to as a binary code.

Example 1.6.1 Suppose we determine that all codewords of length 3 are only legal if they are even when added together and illegal if they are odd. Thus, the code vector $(1,0,1) = 0$ is legal whereas $(0,0,1) = 1$ is illegal.

For error-correcting purposes, we want these codewords to be as far away from each other as possible. Otherwise, a simple error could turn one codeword into another. This is typically done by spacing out the illegal codewords among the legal codewords. By doing this, we further the Hamming distance between two legal codewords.

Definition 1.6.2 A *Hamming distance* is the distance between two codewords c_i and c_j in a message m. This is measured by counting the amount of differing bits in the two codewords.

Example 1.6.1 Suppose $c_i = (0,0,0)$ and $c_j = (1,1,1)$. The Hamming distance between these two codewords is three.

Definition 1.6.3 The smallest Hamming distance, denoted $d(C)$, between two legal codewords c_i and c_j in the code C is called the *minimum Hamming distance* of C. This reflects the error-correcting capability of a code.

Definition 1.6.4 For a codeword c_i, the *Hamming weight*, denoted $wt(c_i)$, is the number of nonzero places in c_i. For a binary codeword, this would be the number of ones in the codeword.

Example 1.6.2 Let $c = (1,0,1)$. The Hamming weight of this number would therefore be two.

1.7 Linear Codes

Definition 1.7.1 A *linear code* of dimension k and length n over a field F is a k-dimensional subspace of the vector space F^n, a set of n-dimensional vectors and can be referred to as an $[n, k]$ code. If the minimum Hamming distance of the code is d, then the code is called a $[n, k, d]$ code.

For our use, a linear code is a binary code of length n and dimension k, a set of 2^k binary n-tuples, i.e. the codewords, such that a sum of two codewords is always another codeword.

Part II: Goppa Codes

2.1 Definition

A Goppa code is a linear, error-correcting code that can be used to encrypt and decrypt a message. Such a code has the following definition.

Definition 2.1.1 Let a Goppa polynomial be defined as a polynomial over $GF(p^m)$, that is,

$$g(x) = g_0 + g_1 x \dots + g_t x^t = \sum_{i=0}^{t} g_i x^i,$$

with each $g_i \in GF(p^m)$. Let L be a finite subset of the extension field $GF(p^m)$, p being a prime number, say

$$L = \{\alpha_1, \dots, \alpha_n\} \subseteq GF(p^m)$$

such that $g(\alpha_i) \neq 0$ for all $\alpha_i \in L$. Given a codeword vector $c = (c_1, \dots, c_n)$ over $GF(q)$, we have the function

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

where $\frac{1}{x-\alpha_i}$ is the unique polynomial with $(x-\alpha_i) * \frac{1}{x-\alpha_i} \equiv 1 \left(mod\ g(x)\right)$ with a degree less than or equal to $t-1$. Then, a *Goppa code* $\Gamma\left(L, g(x)\right)$ is made up of all code vectors c such that $R_c(x) \equiv 0 \left(mod\ g(x)\right)$. This means that the polynomial $g(x)$ divides $R_c(x)$.

2.2 Parameters

Recall that Goppa codes are linear codes. As such, we can use the notation $[n, k, d]$ to describe a Goppa code with the parameters of length n, dimension k, and minimum Hamming distance d. The length n depends solely on the subset L.

Theorem 2.2.1 The dimension k of a Goppa code $\Gamma\big(L, g(x)\big)$ of length n is greater than or equal to $n - mt$, that is, $k \geq n - mt$.

Theorem 2.2.2 The minimum distance d of a Goppa code $\Gamma\big(L, g(x)\big)$ of length n is greater than or equal to $t + 1$, that is $d \geq t + 1$.

Proof

Please refer to Jochemsz [E] for the proof of the above theorems.

2.3 Binary Goppa Codes

A binary Goppa code is when $\Gamma\big(L, g(x)\big)$ uses a polynomial $g(x)$ over $GF(2^m)$ of degree t. For this paper, we want to particularly emphasize irreducible binary Goppa codes. Recall that binary refers to modulo 2. Irreducible means we choose $g(x)$ to be an irreducible Goppa polynomial. This is because the generating polynomial of such a code allows us to generate an efficiently error-correcting algorithm. Also, we can more accurately estimate the lower bound of the Hamming distance.

Theorem 2.3.1 An irreducible, binary Goppa code $\Gamma\big(L, g(x)\big)$ has a minimum distance d of greater than or equal to $2t + 1$, that is, $d \geq 2t + 1$.

Proof

Please refer to section 2.2 in Engelbert, Overbeck, and Schmidt [C] for this proof.

Q.E.D.

Therefore, the parameters for an irreducible, binary Goppa code would be $[n, \geq n - mt, \geq 2t + 1]$.

For the rest of this paper, the reader should assume that by Goppa code, we mean a binary, irreducible Goppa code.

2.4 Parity Check Matrix

The parity check matrix of a Goppa code is used in order to decode a message.

Definition 2.4.1 A *parity check matrix* of a Goppa code is defined to be a matrix H such that $Hc^T = 0$ for all code vectors c in $GF(2^m)$ that satisfy the Goppa code requirement.

Proposition 2.4.1 If we set $H = XYZ$ such that

$$X = \begin{pmatrix} g_t & 0 & 0 & \cdots & 0 \\ g_{t-1} & g_t & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \cdots & g_t \end{pmatrix}, Y = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix}, \text{ and } Z =$$

$$\begin{pmatrix} \frac{1}{g(\alpha_1)} & 0 & \cdots & 0 \\ 0 & \frac{1}{g(\alpha_2)} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{g(\alpha_n)} \end{pmatrix}, \text{ then matrix } H \text{ is a parity check matrix for a Goppa code } \Gamma(L, g(x)).$$

Proof [According to Engelbert, Overbert, Schmidt [F]]

Since $g(x)$ is irreducible, there exists a primitive element α for all $\alpha \in GF(2^m)$ such that $g(\alpha) \neq 0$. And so the subset L can contain all elements of $GF(2^m)$.

Notice that $\frac{g(x)-g(\alpha_i)}{x-\alpha_i} = \sum_{j=0}^{t} g_j \cdot \frac{x^i-\alpha_i^j}{x-\alpha_i} = \sum_{w=0}^{t-1} x^w \sum_{j=w+1}^{t} g_j \, \alpha_i^{j-1-w}, for\ all\ 1 \leq i <$ $n+1$. An arbitrary vector $c \in \Gamma(L, g(x))$ if and only if $\sum_{i=1}^{n}(\frac{1}{g(\alpha_i)}\sum_{w+1}^{t} g_j \alpha_i^{j-1-w}) \cdot c_i =$ 0, for all $w = 0, \dots, t-1$.

And so the parity check matrix H can be written as $H = XYZ$, where $X =$

$$\begin{pmatrix} g_t & 0 & 0 & \cdots & 0 \\ g_{t-1} & g_t & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \cdots & g_t \end{pmatrix}, Y = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix}, \text{and } Z = \begin{pmatrix} \frac{1}{g(\alpha_1)} & 0 & \cdots & 0 \\ 0 & \frac{1}{g(\alpha_2)} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{g(\alpha_n)} \end{pmatrix}.$$

Therefore, we have that any codeword $c \in \Gamma(L, g(z))$ if and only if $Hc^T = 0$.

Q.E.D.

2.5 Encoding

The encoding of a Goppa code involves multiplying the message by the generator matrix of the Goppa code.

Definition 2.5.1 The *generator matrix* of a Goppa Code is defined to be the $k \times n$ matrix G such that the rows of G for the basis of the Goppa code $\Gamma\big(L, g(x)\big)$.

Proposition 2.5.1 Any $n \times k$ matrix G with rank k, such that $GH^T = 0$, is a generator matrix.

Proof

This proposition follows immediately from Proposition 2.4.1.

Q.E.D.

In order to send a message using Goppa codes, the message is first written in blocks of k symbols. Then, each block is multiplied by the generator matrix G. The resulting vectors are a set of codewords. An example of one block being encoded follows:

$$(m_1, m_2, \dots, m_k) * G = (c_1, \dots, c_n).$$

2.6 Irreducible Binary Goppa Code Example.

Note that $GF(2^4) \cong GF(2)[X]/(k(X))$ for every irreducible polynomial $k(X)$ of degree 4. First, we find a primitive element α. We can factor $X^{15} - 1 \ (mod\ 2)$ into irreducible factors.

$$X^{15} - 1(mod\ 2) = (X+1)(X^2+X+1)(X^4+X+1)(X^4+X^3+1)(X^4+X^3+X^2+X+1)$$

We will leave the actual calculations for the above equation to the reader.

If we let $k(X) = X^4 + X + 1$, then α, a root of $k(X)$, is a primitive element if and only if the order of α is 15. Because $\alpha \neq 1$ and the order of an element must divide the order of the group, we only need to check when $\alpha^3 \neq 1$ and $\alpha^5 \neq 1$. By using $\alpha^4 = \alpha + 1$, we compute $\alpha^3 = \alpha^3 \neq 1$ and $\alpha^5 = \alpha \cdot \alpha^4 = \alpha(1+\alpha) = \alpha^2 + \alpha \neq 1$.

Therefore, $GF(2^4)^*$, the multiplicative group of nonzero elements of $GF(2^4)$, is actually a cyclic subgroup generated by α, that is, $< \alpha >$, and so

$$GF(2^4) = GF(2^4) \cup \{0\} = \{0,1,\alpha,\alpha^2,\alpha^3,\dots,\alpha^{14}\}.$$

As a result, we can write the elements of $GF(2^4)$ as the powers of α, plus the element 0,much like how we write binary form. Again, we use the fact that $\alpha^4 = \alpha + 1$.

$$0 = 0 \cdot 1 + 0 \cdot \alpha + 0 \cdot \alpha^2 + 0 \cdot \alpha^3 = (0,0,0,0)^T$$

$$1 = 1 \cdot 1 + 0 \cdot \alpha + 0 \cdot \alpha^2 + 0 \cdot \alpha^3 = (1,0,0,0)^T$$

$$\alpha = 0 \cdot 1 + 1 \cdot \alpha + 0 \cdot \alpha^2 + 0 \cdot \alpha^3 = (0,1,0,0)^T$$

$$\alpha^2 = 0 \cdot 1 + 0 \cdot \alpha + 1 \cdot \alpha^2 + 0 \cdot \alpha^3 = (0,0,1,0)^T$$

$$\alpha^3 = 0 \cdot 1 + 0 \cdot \alpha + 0 \cdot \alpha^2 + 1 \cdot \alpha^3 = (0,0,0,1)^T$$

$$\alpha^4 = 1 \cdot 1 + 1 \cdot \alpha + 0 \cdot \alpha^2 + 0 \cdot \alpha^3 = (1,1,0,0)^T$$

$$\alpha^5 = 0 \cdot 1 + 1 \cdot \alpha + 1 \cdot \alpha^2 + 0 \cdot \alpha^3 = (0,1,1,0)^T$$

$$\alpha^6 = 0 \cdot 1 + 0 \cdot \alpha + 1 \cdot \alpha^2 + 1 \cdot \alpha^3 = (0,0,1,1)^T$$

$$\alpha^7 = 1 \cdot 1 + 1 \cdot \alpha + 0 \cdot \alpha^2 + 1 \cdot \alpha^3 = (1,1,0,1)^T$$

$$\alpha^8 = 1 \cdot 1 + 0 \cdot \alpha + 1 \cdot \alpha^2 + 0 \cdot \alpha^3 = (1,0,1,0)^T$$

$$\alpha^9 = 0 \cdot 1 + 1 \cdot \alpha + 0 \cdot \alpha^2 + 1 \cdot \alpha^3 = (0,1,0,1)^T$$

$$\alpha^{10} = 1 \cdot 1 + 1 \cdot \alpha + 1 \cdot \alpha^2 + 0 \cdot \alpha^3 = (1,1,1,0)^T$$

$$\alpha^{11} = 0 \cdot 1 + 1 \cdot \alpha + 1 \cdot \alpha^2 + 1 \cdot \alpha^3 = (0,1,1,1)^T$$

$$\alpha^{12} = 1 \cdot 1 + 1 \cdot \alpha + 1 \cdot \alpha^2 + 1 \cdot \alpha^3 = (1,1,1,1)^T$$

$$\alpha^{13} = 1 \cdot 1 + 0 \cdot \alpha + 1 \cdot \alpha^2 + 1 \cdot \alpha^3 = (1,0,1,1)^T$$

$$\alpha^{14} = 1 \cdot 1 + 0 \cdot \alpha + 0 \cdot \alpha^2 + 1 \cdot \alpha^3 = (1,0,0,1)^T$$

Consider the following Goppa code over subset $L = \{\alpha^i \; such \; that \; 2 \leq i \leq 13\}$ with

$$g(x) = x^2 + x + \alpha^3.$$

Note that this code is irreducible over $GF(2^4)$. Therefore, this code has the parameters $p = 2, m = 4, n = 12$, and $t = 2$. We know from Theorem 2.2.1 that $k \geq n - mt = 12 - 4 \cdot 2 = 4$. And from Theorem 2.3.1, we know $d \geq 2t + 1 = 2 \cdot 2 + 1 = 5$. And so we have a $[12, \geq 4, \geq 5]$ Goppa code.

To find the parity check matrix H, we can use Proposition 2.4.1 by assigning $g_1 = \alpha_7, g_2 = 1, and \; \alpha_1 = \alpha^2, \alpha_2 = \alpha^3, \dots, \alpha_{12} = \alpha^{13}$. We can then calculate the factors $\frac{1}{g(\alpha_i)}$ for $i = 1, \dots, 12$.

$$\frac{1}{g(\alpha_1)} = \frac{1}{(\alpha^2)^2 + x + \alpha^3} = \frac{1}{\alpha^4 + \alpha^2 + \alpha^3} = [(1,1,0,0)^T + (0,0,1,0)^T + (0,0,0,1)^T]^{-1}$$

$$= [(1,1,1,1)^T]^{-1} = (\alpha^{12})^{-1} = \alpha^3$$

$$\frac{1}{g(\alpha_2)} = \frac{1}{(\alpha^3)^2 + x + \alpha^3} = \frac{1}{\alpha^6 + \alpha^3 + \alpha^3} = [(0,0,1,1)^T + (0,0,0,1)^T + (0,0,0,1)^T]^{-1}$$

$$= [(0,0,1,1)^T]^{-1} = (\alpha^6)^{-1} = \alpha^9$$

$$\frac{1}{g(\alpha_3)} = \frac{1}{(\alpha^4)^2 + \alpha^4 + \alpha^3} = \frac{1}{\alpha^8 + \alpha^4 + \alpha^3} = [(1,0,1,0)^T + (1,1,0,0)^T + (0,0,0,1)^T]^{-1}$$

$$= [(0,1,1,1)^T]^{-1} = (\alpha^{11})^{-1} = \alpha^4$$

$$\frac{1}{g(\alpha_4)} = \frac{1}{(\alpha^5)^2 + \alpha^5 + \alpha^3} = \frac{1}{\alpha^{10} + \alpha^5 + \alpha^3} = [(1,1,1,0)^T + (0,1,1,0)^T + (0,0,0,1)^T]^{-1}$$

$$= [(1,0,0,1)^T]^{-1} = (\alpha^{14})^{-1} = \alpha$$

$$\frac{1}{g(\alpha_5)} = \frac{1}{(\alpha^6)^2 + \alpha^6 + \alpha^3} = \frac{1}{\alpha^{12} + \alpha^6 + \alpha^3} = [(1,1,1,1)^T + (0,0,1,1)^T + (0,0,0,1)^T]^{-1}$$

$$= [(1,1,0,1)^T]^{-1} = (\alpha^7)^{-1} = \alpha^8$$

$$\frac{1}{g(\alpha_6)} = \frac{1}{(\alpha^7)^2 + \alpha^7 + \alpha^3} = \frac{1}{\alpha^{14} + \alpha^7 + \alpha^3} = [(1,0,0,1)^T + (1,1,0,1)^T + (0,0,0,1)^T]^{-1}$$

$$= [(0,1,0,1)^T]^{-1} = (\alpha^9)^{-1} = \alpha^6$$

$$\frac{1}{g(\alpha_7)} = \frac{1}{(\alpha^8)^2 + \alpha^8 + \alpha^3} = \frac{1}{\alpha + \alpha^8 + \alpha^3} = [(0,1,0,0)^T + (1,0,1,0)^T + (0,0,0,1)^T]^{-1}$$

$$= [1,1,1,1]^{-1} = (\alpha^{12})^{-1} = \alpha^3$$

$$\frac{1}{g(\alpha_8)} = \frac{1}{(\alpha^9)^2 + \alpha^9 + \alpha^3} = \frac{1}{\alpha^3 + \alpha^9 + \alpha^3} = [(0,0,0,1)^T + (0,1,0,1)^T + (0,0,0,1)^T]^{-1}$$

$$= [(0,1,0,1)^T]^{-1} = (\alpha^9)^{-1} = \alpha^6$$

$$\frac{1}{g(\alpha_9)} = \frac{1}{(\alpha^{10})^2 + \alpha^{10} + \alpha^3} = \frac{1}{\alpha^5 + \alpha^{10} + \alpha^3} = [(0,1,1,0)^T + (1,1,1,0)^T + (0,0,0,1)^T]^{-1}$$

$$= [(1,0,0,1)^T]^{-1} = (\alpha^{14})^{-1} = \alpha$$

$$\frac{1}{g(\alpha_{10})} = \frac{1}{(\alpha^{11})^2 + \alpha^{11} + \alpha^3} = \frac{1}{\alpha^7 + \alpha^{11} + \alpha^3} = [(1,1,0,1)^T + (0,1,1,1)^T + (0,0,0,1)^T]^{-1}$$

$$= [(1,0,1,1)^T]^{-1} = (\alpha^{13})^{-1} = \alpha^2$$

$$\frac{1}{g(\alpha_{11})} = \frac{1}{(\alpha^{12})^2 + \alpha^{12} + \alpha^3} = \frac{1}{\alpha^9 + \alpha^{12} + \alpha^3} = [(0,1,0,1)^T + (1,1,1,1)^T + (0,0,0,1)^T]^{-1}$$

$$= [(1,0,1,1)^T]^{-1} = (\alpha^{13})^{-1} = \alpha^2$$

$$\frac{1}{g(\alpha_{12})} = \frac{1}{(\alpha^{13})^2 + \alpha^{13} + \alpha^3} = \frac{1}{\alpha^{11} + \alpha^{13} + \alpha^3} = [(0,1,1,1)^T + (1,0,1,1)^T + (0,0,0,1)^T]^{-1}$$

$$= [(1,1,0,1)^T]^{-1} = (\alpha^7)^{-1} = \alpha^8$$

We can then compute $XYZ = H$.

$$H = \begin{pmatrix} g_2 \cdot g(\alpha_1)^{-1} & g_2 \cdot g(\alpha_2)^{-1} & \cdots & g_2 \cdot g(\alpha_{12})^{-1} \\ (g_1 + g_2 \cdot \alpha_1) \cdot g(\alpha_1)^{-1} & (g_1 + g_2\alpha_2) \cdot g(\alpha_2)^{-1} & \cdots & (g_1 + g_2\alpha_{12}) \cdot g(\alpha_{12})^{-1} \end{pmatrix}$$

$$= \begin{pmatrix} \alpha^3 & \alpha^9 & \alpha^4 & \alpha & \alpha^8 & \alpha^6 & \alpha^3 & \alpha^6 & \alpha & \alpha^2 & \alpha^2 & \alpha^8 \\ 1 & \alpha^{13} & \alpha^7 & \alpha^{14} & \alpha^3 & 0 & \alpha^{14} & \alpha^6 & \alpha^7 & \alpha^{10} & \alpha^4 & \alpha^{13} \end{pmatrix}$$

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

Recall that $GH^T = 0$. Therefore, we can compute the rows of G to be the vectors of the nullspace of $H \ mod \ 2$. And so $G = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$. Since the dimensions of this matrix is 4×12, we can conclude that the dimension of $\Gamma(L, g(x))$ is 4. And so this Goppa code has the parameters $[12, 4, \geq 5]$.

2.7 Error Correction

Let y be the received codeword with $r \leq t$ errors. Then

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

where there are r places where $e_i \neq 0$. In order to correct the codeword back into the original codeword, we must first find the error vector. As such, we need to find the set of

error locations, $E = \{\, i \; such \; that \; e_i \neq 0\}$, and the corresponding error values e_i for all $i \in E$.

Definition 2.7.1 The *error locating polynomial* $\sigma(x)$ is defined as

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

Since we are using binary Goppa codes, it is sufficient to simply locate the errors since there is only one other possible element. If we were to use regular Goppa codes, we would need to calculate a separate error correction polynomial as well.

In order to error correct a codeword, we must apply Patterson's algorithm [H]. His algorithm for correcting $r \leq t$ errors for $g(x)$ irreducible over $GF(2^m)$ is as follows:

1. Let $y = (y_1, \dots, y_n)$ be a received codeword. Compute the syndrome
$$s(x) = \sum_{i=1}^{n} \frac{y_i}{x - \alpha_i} \; mod \; g(x).$$
2. Calculate $\sigma(x)$ by following the next four steps:
    a. Find $h(x)$ such that $s(x)h(x) \equiv 1 \left(mod \; g(x)\right)$. If $h(x) = x$, then we are finished and the solution is $\sigma(x) = x$.
    b. Calculate $d(x)$ such that $d^2(x) \equiv h(x) + x \; (mod \; g(x))$.
    c. Find $a(x)$ and $b(x)$, with $b(x)$ of least degree, such that $d(x)b(x) \equiv a(x)\left(mod \; g(x)\right)$.
    d. Set $\sigma(x) = a^2(x) + b^2(x)x$.
3. Use $\sigma(x)$ to determine the set of error locations $E = \{\, i \; such \; that \; \sigma(\alpha_i) = 0\}$.
4. Define the error vector $e = (e_1, \dots, e_n)$ by $e_i = 1$ for $i \in E$ and $e_i = 0$ elsewhere.
5. Define the codeword $c = y - e$.

2.8 Decoding

Once all possible errors in a codeword are corrected, the receiver can easily recover the original message. Recall that $(m_1, m_2, \dots, m_k) * G = (c_1, \dots, c_n)$. Think of this function as a map from $F^k \to F^n$ such that $m \to mG$. Since m_k has rank k and G has rank k and F^n has rank n, this map from $F^k \to F^n$ is injective. Therefore, we can rearrange the equation $(m_1, m_2, \dots, m_k) * G = (c_1, \dots, c_n)$ so that $G^T \cdot \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$.

This is simply a system of n equations with k unknowns, which we can use row reduction to solve,

$$\left(G^T \middle| \begin{matrix} c_1 \\ \vdots \\ c_n \end{matrix}\right) \sim \cdots \sim \begin{pmatrix} 1 & 0 & \cdots & 0 & m_1 \\ 0 & 1 & \cdots & 0 & m_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & m_k \\ - & - & - & - & - \\ & & X & & \end{pmatrix},$$

such that X is a $(n-k) \times (k+1)$ matrix.

Part III: The McEliece Cryptosystem

3.1 Overview

The McEliece Cryptosystem is a type of Public Key cryptosystem that uses a linear, error-correcting code in order to create a public key and a private key. The original error-correcting code used in this cryptosystem is the binary Goppa code. A public key, as one would assume, is public; anyone and everyone can find it. The public key is based on the

private key, but in such a way that makes it unfeasible to recover the private key. In order to do this, the private key is only held by the receiver of the message. Traditionally, we use the example of two friends, Alice and Bob, to explain cryptography.

Suppose Alice wants to send a private message to Bob. Bob must first publish his public key, which is based on his private key. Then, Alice takes Bob's public key and encrypts her message with it. The message then becomes a codeword. She sends her encrypted message to Bob. Bob then uses his private key to decrypt the codeword and read the message.

In order to construct the public and private keys, Bob must first choose an arbitrary Goppa polynomial $g(z)$ with a degree t over $GF\ (2^m)$. The Goppa code defined by this polynomial and by L has parameters $[n, \geq n - mt, \geq 2t + 1]$. Using this, Bob would then compute the $k\ x\ n$ generator matrix G of the Goppa code. Then, Bob randomly chooses a $k\ x\ k$ invertible matrix S and a $n\ x\ n$ permutation matrix P, which means that P has exactly one 1 in every row and column, with all other entries being zero. Then he computes $G' = SGP$. G' is his encoding matrix. This results in his public key consisting of G' and t only.

The private key consists of the polynomial $g(z)$, the original matrix G, along with matrices S and P such that $G' = SGP$.

Once Bob publishes his public key, Alice generates a random binary vector e of length k that has a weight $wt(e) \leq t$. Then, Alice can encode her message $m = (m_1, m_2, \dots, m_k)$ by computing $y = mG' + e$. Then, Alice sends her ciphertext y.

Bob receives Alice's codeword and uses his permutation matrix P to compute

$$y' = yP^{-1} = mG'P^{-1} + eP^{-1} = mSGPP^{-1} + e' = (mS)G + e'.$$

Bob can then decode y' into the message $m' = mS$ by finding e', which is done by Bob applying Patterson's algorithm. Once this is done, Bob can is calculate $y - e'$=mSG and since Bob knows what S is, he can calculate S^{-1}, and then recover the original message $m = m'S^{-1}$.

3.2 Example

We will use the same Goppa code from our previous examples. Recall our generator matrix

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

Now we need to choose our random matrices S and P and modify matrix G. G is a 4×12 matrix, and so S must be a 4×4 matrix and P must be a 12×12 matrix. I have chosen our random matrices as the following:

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

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

Using these matrices, we compute the public encryption matrix G' such that

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

Bob then publishes this matrix G' along with $t = 2$. Notice that anyone can encrypt a message with this information. But let us suppose Alice wants to send a message $m = (1,0,1,0)$.

First, we compute the matrix

$$mG' = (1,1,1,1,1,1,0,1,1,1,1,0),$$

and then we add a random error vector $e = (1,1,0,0,0,0,0,0,0,0,0,0)$ so that,

$$y = mG' + e = (0,0,1,1,1,1,0,1,1,1,1,0).$$

This results in the ciphertext that Alice then sends to Bob. Bob then wants to retrieve the original message m from the ciphertext y. In order to do so, he first computes yP^{-1} by using his secret permutation matrix P.

$$yP^{-1} = mG'P^{-1} + eP^{-1}$$

$$= m(SGP)P^{-1} + e'$$

$$= (mS)G + e'$$

$$= (0,0,1,1,1,1,0,1,1,1,1,0) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= (0,1,1,1,1,0,1,0,1,1,1,0).$$

With this permutation, the errors have been moved to the first and sixth columns. Bob corrects these by using the error-correcting algorithm and finds

$$mSG = (1,1,1,1,1,1,1,0,1,1,1,0).$$

We know from section 2.8 that mS can be found by row reducing $[G^T|(mSG)^T]$. Thus we find $mS = (1,1,0,1)$. Finally, we compute $m = (1,1,0,1) \cdot \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = (1,0,1,0)$.

3.3 Attacking

In cryptology, there are various types of attacks. A brute force attack is when the enemy tries every possible key and determines which key results in a meaningful message.

For this attack, the longer the length of a key, the longer it will take to try every possible key. A direct, or per-message, attack is an attack that tries to decode a given message, but does not necessarily solve the entire cryptosystem. A structural attack occurs when Eve would try to recover the structure, or at least part of the structure, of the original message from the public key.

The most effective attack against the McEliece cryptosystem is an attack called "information-set decoding." Many cryptologists have published variants of this attack from McEliece in his original paper [K] to Stern [M] in his subsequent paper. Most of the variants of the information-set decoding are based off of Stern's attack.

Stern's attack refers to Stern's method on finding codewords with a low Hamming weight. In order to do so, we must select an integer $w \geq 0$ and a $(n-k) \times n$ parity check matrix K for an $[n, k]$ code over the binary field. Our goal is to find a solution to the equation $Kz = 0$ with the weight $|z| = w$. Then we complete the following steps:

1. Randomly select a $n \times k$ column from K. Select $n - k - 1$ additional columns out of the remaining columns of K such that each column is chosen due to pivots in previous columns by using Gaussian elimination.
2. Randomly select a subset Z from the $n - k$ columns consisting of l elements.
3. Partition the leftover columns into two subsets X and Y. Each column is added to either X or Y with the probability of $\frac{1}{2}$.

4. Search for codewords with exactly p nonzero bits in X, exactly p nonzero bits in Y, no nonzero bits in Z, and exactly $w - 2p$ nonzero bits in the remaining columns by completing the following steps.
    a. Apply elementary row operations to the matrix K so that the randomly selected $n - k$ columns become the identity matrix.
    b. For every subset A of X with p elements, add the columns of the matrix $modulo$ 2 for each row l in order to compute the l-bit vector $\pi(A)$. Do the same for every subset B of Y with p elements.
    c. Find all pairs A, B such that $\pi(A) = \pi(B)$. For every such pair, compute the sum of all members of $A \cup B$. Said sum is a $(n - k)$-bit vector. If the vector does not have a weight of $w - 2p$, then the attack has failed. If the vector does have a weight of $w - 2p$, then the solution can be found by adding the corresponding columns and those columns, together with A and B, form a codeword with weight w. If there are no such codewords, the attack fails and Stern starts over with s new selection of columns.

Say we have a code C over the binary field. Let y have a distance w from a codeword $c \in C$, then $y - c$ is a element with a weight w of the code $C + \{0, y\}$. Eve knows the McEliece public key G', t. thus, Eve can add y to the list of generators in order to form a generator matrix $C + \{0, y\}$. Note that the only codeword with weight t is $y - c$, which is precisely the codeword we found with Stern's attack. Therefore, Eve can use this codeword to find c and solve for the original message.

In the paper by Bernstein, Lange, and Peters [C], they present an improved attack based on Stern's method. They prove that their attack is more effective than any previous attack. They demonstrated their attack on a [1024, 524] Goppa code and prove that they can successfully break the code in about 1400 days with the use of a 2.4 GHz Intel Core 2 Quad Q6600 CPU. By using 200 of these computers, their attack takes approximately one week. They note that the previously most effective attack by Canteaut, Chabaud, [D] and Sendrier [E] would need 7400000 days on one 433MHz DEC Alpha CPU and would need 220000 days if the improvements in hardware were taken into account.

3.4 Security

As stated at the beginning of the paper, we assume that Eve knows what cryptosystem is being used. This assumption is used to assess the security of a cryptosystem and is referred to as *Kerckhoffs's Principle:* When determining how secure a cryptosystem is, one should always assume the enemy knows the method that is being used [N, page 4].

The security of this cryptosystem depends on how difficult it is to decode y' in order to obtain m'. Eve will have a hard time trying to separate G from G' because in order to do so, she would need to know the inverse of matrix S, which was not published. Also, Eve does not know what P is either, which means Eve cannot find y' in order to find m'.

Multiplying SG together scrambles the message into another matrix. Then when we multiply the result by the permutation matrix P, we further scramble the matrix by randomizing the order of the columns. This is done in order to make the resulting matrix look random and ensure the difficulty of obtaining the decoding method from the encoding

matrix. Because of this scrambling of the matrices, Eve cannot decipher the private key from the public key.

In order to make this system as secure as possible, the chosen code needs to be very large in order to successfully hide the Goppa code within the scrambled generator matrix G'. This is a major drawback because using a code with such a large key size makes it very difficult to use this system practically. But as technology continues to advance and the potential key size increases, this cryptosystem will become more and more useful.

In his original paper on the McEliece cryptosystem [K], McEliece suggested the use of a [1024, 524] Goppa code. Such a code can correct for up to 50 errors. Ever since this introduction, there have been a variety of modifications made to the cryptosystem, most of them using different codes. However, most of these modifications have proved lacking and less secure than the original proposal.

Repeatedly sending the same message will also make the cryptosystem more susceptible to attack. This is because sending the same message twice will generate two different ciphertext and since the location of the errors are unlikely to be the same, Eve could then compare and potentially uncover the original message.

References

[A] Artin, Michael. *Algebra.* Englewood Cliffs, NJ: Prentice Hall, 1991. Print.

[B] Berlekamp, E. "Goppa Codes." IEEE Transactions on Information Theory 19.5 (1973): 590-92. Web.

[C] Bernstein, Daniel J., Tanja Lange, and Christiane Peters. "Attacking and Defending the McEliece Cryptosystem." Post-quantum Cryptography: Second International Workshop, PQCrypto 2008 Cincinnati, OH, USA, October 17-19, 2008 Proceedings. Ed. Johannes Buchmann and Jintai Ding. Berlin: Springer, 2008. 31-46. Print.

[D] Canteaut, Anne, and Chabaud, Florent. "A New Algorithm for Finding Minimum-Weight Words In a Linear Code: Application to the McEliece Cryptosystem and to narrow-sense BCH codes of length 511. IEEE Transactions on Information Theory, 44(1):367-378, 1998. Print.

[E] Canteaut, Anne, and Sendrier Nicolas. "Cryptanalysis of the Original McEliece Cryptosystem." Ad*vances in Cryptology—ASIACRYPT '98*, volume 1514 of Lecture Notes in Computer Science, pages 187-199. Springer, Berlin, 1998.

[F] Engelbert, Daniela, Raphael Overbeck, and Arthur Schmidt. "A summary of McEliece-type cryptosystems and their security." Journal of Mathematical Cryptology 1, 151–199. MR 2008h:94056. Print.

[G] Goppa, V. D. *Geometry and Codes.* Dordrecht: Kluwer Academic, 1988. Print.

[H] Jiang, Yuan. *A Practical Guide to Error-Control Coding Using MATLAB.* Boston: Artech House, 2010. Print.

[I] Jochemsz, Ellen. "Goppa Codes & the McEliece Cryptosystem." Amsterdam: Vrije Universiteit Amsterdam, 2002. Print.

[J] Lay, David C. *Linear Algebra and Its Applications.* Boston: Pearson/Addison-Wesley, 2012. Print.

[K] McEliece, R. J. "A Public-Key Cryptosystem Based on Algebraic Coding Theory." Jet Propulsion Laboratory DSM Progress Report 42-44. N.p., n.d. Web. 12 Jan. 2015.

[L] Patterson, N. "The Algebraic Decoding of Goppa Codes." *IEEE Transactions on Information Theory* 21.2 (1975): 203-07. Web.

[M] Stern, Jaques. "A Method For Finding Codewords of Small Weight." *Coding Theory and Applications,* 388: 106-133, 1989.

[N] Trappe, Wade, and Lawrence C. Washington*. Introduction to Cryptography: With Coding Theory.* Upper Saddle River, NJ: Pearson Prentice Hall, 2006. Print.