5-1-2022

Applications of Algebraic Coding Theory to Cryptography

Kylie Schnoor
University of Northern Colorado

University of Northern Colorado

Greeley, Colorado

APPLICATIONS OF ALGEBRAIC CODING THEORY TO CRYPTOGRAPHY

A Thesis

Submitted in Partial

Fulfillment for Graduation with Honors Distinction and

the Degree of Bachelor's of Science

Kylie Schnoor

College of Natural and Health Sciences

MAY 2022

Abstract

Whether it is online commerce, international relations, or simply through email communication, the encryption and decryption of data is essential to the inner workings of everyday life. To encrypt and decrypt efficiently, it is important that there is some structure behind the process rather than just a random procedure. The purpose of this research is to analyze different encryption schemes and their structure, with a focus on schemes that apply algebraic coding theory to cryptography. Cryptosystems based in algebraic coding theory are particularly important to the future of cryptography, as they are resistant to attacks by quantum computers, unlike many currently employed cryptosystems.

Specifically, we examine the McEliece cryptosystem and its variations, in particular the use of Reed-Solomon codes. The goal is to understand the algebraic structure underlying the McEliece cryptosystem as well as to understand its shortcomings and variations that may strengthen it. The current results show that the original Goppa codes that are used in the McEliece systems are stronger and more secure than the proposed Reed-Solomon code alternative.

Acknowledgements

I would like to acknowledge and give the greatest thanks to my advisor Dr. Katherine Morrison who has helped me tirelessly throughout this whole project. Her guidance was extremely helpful throughout all steps of this process whether it was discussing a question and a topic to solve or in the final days helping with edits. Additionally, her guidance as an academic advisor is undoubtedly amazing. I would not be in the same place academically without her help and direction the past three years. I can attribute where I am today with her help.

I would also like to acknowledge and thank my family and friends who have been entirely supportive of this project and my academic career as a whole. Their support and encouragement means the world and I could not do it without them.

Contents

Chapter 1

Introduction

The encryption of data is essential to a number of applications including online commerce, international relations, and simply through email communication. To encrypt and decrypt efficiently, it is important that there is some structure rather than just a random process. This structure not only crucial to the security of the system, but also ensures that the cryptosystem can be widely deployed.

In this work, we explore a variety of mathematical structures underlying different encryption/decryption schemes. To begin, the basic cryptography framework will be described, including the analysis of the differences between private-key and public-key cryptosystems, as well as notable examples of each. While private-key cryptography is symmetric in the encryption and decryption process, public-key is asymmetric to ensure that knowledge of encryption does not reveal knowledge about decryption. This asymmetry creates interesting requirements for the encryption and decryption processes. The continued importance of cryptography and development of alternative cryptosystems will be discussed particularly in the context of the advent of quantum computers.

Next, we present an introduction to algebraic coding theory. Algebraic coding theory investigates how to encode and decode messages so that if noise corrupts part of the message during transmission, it will still be understood by the receiver. We will explore a variety of mechanisms for introducing redundancy and designing codes that are both efficient and effective at error correction. Reed-Solomon codes will particularly be focused on. These codes are constructed by evaluations of polynomials at the elements of certain finite fields, and they rely on polynomial interpolation for decoding.

After, we turn to an application of coding theory to cryptography in the McEliece cryptosystem, and analyze the encryption and decryption processes of this. Additionally, we examine variations on the McEliece cryptosystem, namely the Niederreiter variation, and use of alternative codes such as Reed-Solomon codes. It is proposed that Reed-Solomon codes can be implemented within the McEliece cryptosystem in order to reduce key size. Lastly, a vulnerability of the variant of the McEliece cryptosystem is analyzed. This vulnerability was exposed through a specific attack exploiting the structure of Reed-Solomon codes with the McEliece cryptosystem. This is ultimately problematic as it excludes the straightforward use of Reed-Solomon codes within the McEliece cryptosystem. This motivates the need for future research involving the security of the McEliece cryptosystem with other alternative codes or better disguise of the Reed-Solomon codes moving forward.

Chapter 2

Cryptography Background

Cryptography is the study of encrypting and decrypting messages, with the goal of sending a message so that only the people intended to read it will be able to, without any unwanted eyes. Cryptography is used in practically every aspect of modern day data transmission: *e.g.*, driving down toll roads, secure Internet transactions, and even sending emails. There are many goals that cryptographers have when creating a cryptosystem, with some of the most important being: (1) making the encryption process computationally easy and inexpensive; (2) making the decryption easy for those who have the key and computationally intractable for those who don't; (3) minimizing the space that the encryption and decryption algorithms take so that they can be implemented on a variety of devices (laptops, tablets, phones, etc.) [1]. Irrespective of the cryptosystem, the process of encryption/decryption is the same: given a plaintext message, somehow scramble it to a ciphertext, which is then transmitted to the intended receiver who can unscramble the ciphertext to recover the original plaintext. In this process, keys are needed to implement both encryption and decryption. In private-key cryptography the keys are the same for both encrypting and decrypting data, and thus must be kept completely private. This is problematic for encryption on a large scale since every arbitrary user would need a different key

and there would need to be mechanisms in place for privately communicating a new key for every user. This is the motivation for public-key cryptography. In public-key, any user will be able to encrypt, all using the same key. The encryption key is made public, and the cryptosystems is designed so that this public key doesn't give information about the private key, which is used to decrypt by the receiver.

This section describes and analyzes both of these systems as well as notable examples of each.

2.1 Private-Key Cryptography

Private-key cryptography makes use of only one key, which is used to both encrypt and decrypt messages. Thus, logically, the key must be kept private because anyone possessing the key can recover the message.

One of the first examples of private-key cryptography was the Caesar Cipher [2]. The cipher first utilizes an alpha-numeric conversion (A=0, B=1, C=2 ...). Then the encryption follows the equation

$$y \equiv x + 3 \mod 26$$

while decryption is precisely the inverse functions

$$x \equiv y - 3 \mod 26.$$

Example 2.1.1. Suppose you want to encrypt the plaintext message "MATH IS

COOL". First the alpha-numeric conversion is applied:

$$12\,0\,19\,7\,8\,18\,2\,14\,14\,11$$

This is followed by the encryption algorithm for the Caeser Cipher:

$$15\,3\,22\,10\,11\,21\,5\,17\,17\,14$$

Finally, the alpha-numeric conversion is applied again with the message broken up into a strings of length 5:

$$\text{PDWKL VFRRO}$$

which are then transmitted. The inverse of this algorithm is used to decrypt the message. Starting with our ciphertext, we reapply the alpha-numeric conversion:

$$15\,3\,22\,10\,11\;\;21\,5\,17\,17\,14$$

This is followed by the decryption algorithm:

$$12\,0\,19\,7\,8\;\;18\,2\,14\,14\,11.$$

Applying the alpha-numeric conversion for the final time yields our original message

$$\text{MATHISCOOL}$$

Notice that knowledge of the encryption key (size of the numerical shift) immediately reveals how to decrypt any ciphertext message. Hence the need to keep the key private. The Caesar Cipher specifically uses a shift of size 3, however any shifts ranging from $1 - 25$ would also be appropriate for the shift cipher.

The most prominent example of private-key cryptography is Data Encryption Standard (DES). DES was created in the 1970s by members of the National Security Agency and the National Institute of Standards and Technology [1]. DES relies on block ciphers in which 64-bit blocks are encoded using a 56-bit key.

A block cipher is a type of algorithm that operates on string containing a fixed number of bits. Specifically, DES uses layers of block ciphers that separate the right and left hand sides of the plaintext to further scramble and encrypt the data [1]. An inverse function is used to decrypt the ciphertext message.

The full details of DES are beyond the scope of this work, in particular because private-key cryptography is no longer widely used. However, DES is still employed for some private communication within private companies. Due to the fact that private-key cryptography only has one key, it is not only impractical to use private-key cryptography for large-scale encryption but it also jeopardizes the security of the cryptosystem. This is a major shortcoming of private-key cryptography. Thus, public-key cryptography is typically used for large-scale encryption and decryption.

2.2 Public-Key Cryptography

Public-key cryptography is similar to private-key in that a function is used to encrypt data and an appropriate inverse function is used to decrypt. However, in public-key cryptography the function and key used for encryption do not reveal the inverse function required for decryption, which relies on a separate key. This means that the while the decryption key is still kept private, the encryption key can be made public. This also makes encryption more accessible; more people are able to use a public key.

At first glance, this may seem to indicate that public-key cryptosystems are weaker or more vulnerable than private-key ones. But fortunately, this is not the case as the decryption key is still extremely difficult to discover without the right information.

One simple, yet extremely powerful example of public-key cryptography is the Rivest-Shamir-Adleman (RSA) cryptosystem. RSA is one of the most common cryptosystems and is typically used to secure information online, particularly in e-commerce transactions [2]. This cryptosystem relies on a fundamental result from abstract algebra and number theory.

2.3 RSA cryptosystem

RSA is one of the most widely used cryptosystems today [2]. It is a public-key cryptosystem and its security relies on the computational intractability of factoring large numbers. The RSA cryptosystem's public keys include $n = pq$ as well as the encoding exponent E. Kept private are the primes p and q (chosen to be large, typically 300-digits long) and the decoding exponent D. The security of RSA lies in the Euler ϕ-function, which counts how many natural numbers less than n are relatively prime to n.

Since $n = pq$ then receiver knows that $\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$. The choice of E and D is informed by the value of $\phi(n)$. Specifically, we exploit the following result:

Theorem 2.3.1. If a, n, are relatively prime then $a^{\phi(n)} \equiv 1 \mod n$ [3]

From this we choose E such that $\gcd(E, \phi(n)) = 1$ and D such that $ED \equiv 1 \mod \phi(n)$. Note that $D = E^{-1} \mod \phi(n)$. For a plaintext message x and asso-

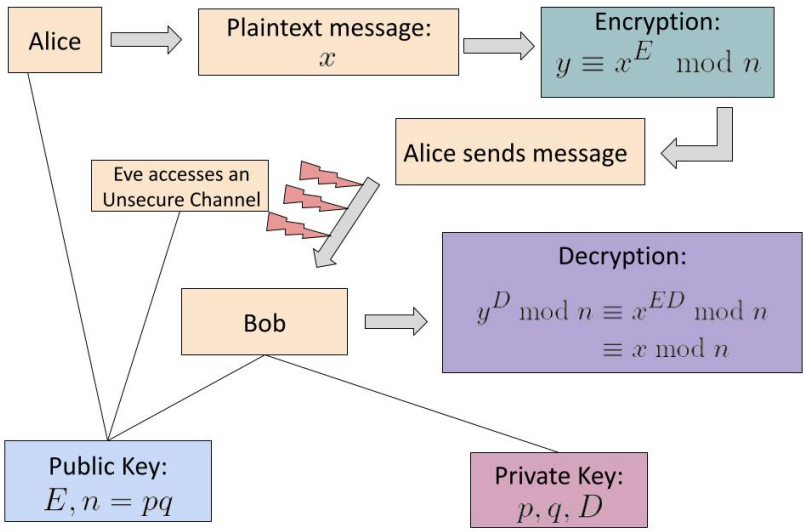

Figure 2.1: Flowchart describing the RSA cryptosystem

ciated ciphertext y, the encryption function is given by

$$y \equiv x^E \mod n$$

From this the derivation of the decryption function is quite simple:

$$\begin{aligned} y &\equiv x^E \mod n \\ y^D \mod n &\equiv (x^E)^D \mod n \\ y^D \mod n &\equiv (x^{ED}) \mod n \\ &\equiv x^1 \mod n \end{aligned}$$

Since it's necessary to know $\phi(n) = (p-1)(q-1)$ in order to discover D, which is necessary to the decryption process. Without the knowledge of p and q, the prime factorization of n must be known, hence the computational intractability because no

polynomial time algorithm exists.

Example 2.3.1. Suppose that Alice wants to send Bob an encrypted message. Bob posts his public key $n = 713$, $E = 13$. Suppose that Alice wants to send the message 420. They can now compute

$$\begin{aligned} y &\equiv 420^{13} \bmod 713 \\ &\equiv 420^8 \cdot 420^4 \cdot 420 \bmod 713 \\ &\equiv 18 \cdot 100 \cdot 420 \bmod 713 \\ &\equiv 220 \end{aligned}$$

and Alice sends Bob 220. Since Bob knows the factorization of n, he can compute

$$\phi(n) = (p-1)(q-1).$$

From this he uses the Extended Euclidean Algorithm to find D such that $ED + \phi(n)k = 1$ for some $k \in \mathbb{Z}$ which guarantees that $ED \equiv 1 \bmod \phi(n)$. Specifically, he can then compute

$$\begin{aligned} x &\equiv 220^{457} \bmod 713 \\ &\equiv 420 \end{aligned}$$

The value of D is the private key that Bob will use to decrypt. Thus, he knows that

$$x \equiv 360^{457} \bmod 660$$

and he can recover the original message 420.

2.4 Quantum Computing & Vulnerabilities of RSA

The security of RSA relies on the inability of a user to factor the public key n, in order to find $\phi(n)$ and D. Generally, this factoring is computationally intractable making the RSA cryptosystem resistant to attacks to crack it in a timely manner. However, that could all change with the widespread availability of quantum computers. Quantum computers take advantage of superposition and multiple states in order to greatly reduce processing time. Specifically, on a standard computer all the existing algorithm for factoring large numbers require non-deterministic polynomial (NP) time. However, on a quantum computer, due to the structure of representation, a polynomial time algorithm exists [4].While these computers currently exists in their earliest forms, they are expensive and impractical for everyday use [5].

One of the current goals in cryptography is to search for other NP-complete problems, whose structure can be exploited to create alternative public-key cryptosystems and cannot be solved efficiently in polynomial time regardless of the computational power or representation [4]. In other words, cryptographers want a problem that cannot be solved by any computer in polynomial time. Specifically, cryptographers seek an NP- complete problem that does not have a polynomial algorithm on a quantum computer. One such problem that has been proposed is the decoding of random linear codewords from an unknown code, which has shown to be NP-complete [4]. Such a problem can be the key computational bottleneck that provides security to a cryptosystem. This motivates the use of algebraic coding theory to develop new cryptosystems. To better understand these cryptosystems, we begin with background on algebraic coding theory.

Chapter 3

Coding Theory

The goal of coding theory is to ensure successful communication of a message even when noise may be present that corrupts parts of the message during transmission. To achieve this, structured redundancy is added onto the message to ensure that errors in transmission can be detected and corrected. One of the simplest encoding schemes is a repetition code, where the message is repeated n times. Through this redundancy, the receiver will be able to compare the messages and find errors. However, there are much more efficient ways to send messages that require significantly fewer redundant bits while still ensuring that the message will be understood by the receiver (see Figure 3.1)

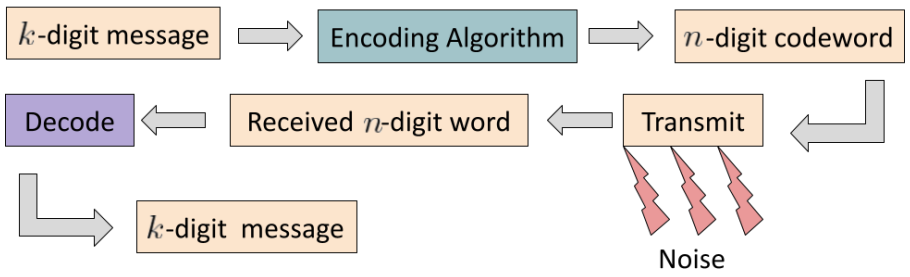

Figure 3.1: Flowchart for the standard coding theory paradigm

The basic structure of encoding is as follows. Uncoded messages (usually binary k-

tuples) are first encoded into codewords (binary n-tuples). The codeword is then transmitted. If the received word contains no errors, it is assumed that there were not enough errors to transform one codeword to another and so it is assumed that no errors occurred and the received word is decoded as itself. However, if errors occur due to noise in the transmission, then the received word will differ in at least one bit from all the codewords. The goal of coding theory is to construct the codes so that the decoding scheme will be able to detect, and then correct a guaranteed number of errors within the received word.

As stated previously, one of the simplest coding schemes is to encode the message by simply repeating it multiple times. This way, the received message can be compared to itself in order to find errors. However, this process is not only time consuming, but also requires a large codeword to be sent for a relatively small message. Thus repetition codes are not ideal, but the idea of building in redundancy in the code is still an important one.

Another coding scheme is to implement a parity check. This checks the number of 1's in the bit string. One bit is appended to the message and will be set to either 0 or 1 to make the total number of 1's even. Depending on the parity of the received message, the receiver will know if there is an error in the message based on the number of 1's, *i.e.* if the parity is odd, then you know that the message contains an error. While a parity check bit is certainly more efficient than just a repeated message, it is still not completely trustworthy for detecting errors, let alone correcting them. For example, a parity check would be useless if more than 1 error had occurred. If two different errors occurred in transmission then the parity would be even again, rendering the check useless. Additionally, a parity check doesn't indicate where the error occurred, just that there is one. Thus, a single parity check does not provide enough redundancy

for an effective code.

3.1 Block and Linear Codes

Block codes are a stronger mathematical tool that will not only allow for codes that detect multiple errors, but also codes that can actually correct errors as well. A block code is one where blocks of k binary digits are encoded to n binary digits [3]. In other words, an (n, k)-block code consists of both an encoding function that encodes k digits to n digits, and a decoding function that goes back again:

$$E : \mathbb{Z}_2^k \to \mathbb{Z}_2^n \qquad\qquad D : \mathbb{Z}_2^n \to \mathbb{Z}_2^k$$

E must be a one-to-one function as two different blocks will not be encoded to the same element. Elements in the image of E, $C = E(\mathbb{Z}_2^k)$, are called codewords [3].

The Hamming distance between two n-tuples is defined to be the number of bits in which the two n-tuples differ. For example, the 4-tuples 0100 and 0001 have a Hamming distance of 2. The *minimum distance* of a code, is the minimum of all the distances between distinct codewords. The Hamming distance is important as it gives a measurement of the possible error detection and correction capabilities a code. The larger the Hamming distance, the more errors that a code will be able to detect and correct. However, if the Hamming distance is too large, then there are fewer n-tuples available as possible codewords and fewer messages can be transmitted. Ideally, codewords are chosen from a sphere-packing arrangement. Figure 3.2 gives a cartoon representation of such a sphere-packing arrangement; in actuality, the possible codewords are points in the Boolean lattice $\mathbb{F}_2^n$, and spheres with respect to Hamming distance are centered around such points. Here we see a two dimensional cartoon of

this sphere packing to understand the error-correcting capabilities of a given code (see cartoon in Figure 3.2).

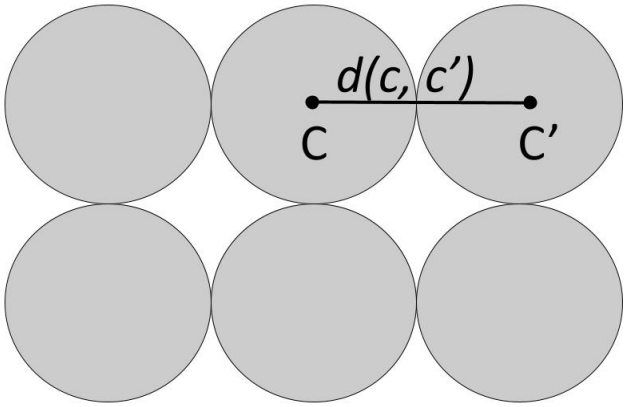

Figure 3.2: Choosing codewords from a sphere-packing arrangement is guaranteed a specified minimum distance and error-correcting capabilities

To see how Hamming distance determines error-detecting/correcting capability, observe that if the minimum distance between codewords is d then it would take d errors to turn one codeword into another. The receiver knows errors have occurred whenever the received words is not a codeword and thus, $d-1$ errors can be detected. However, $\frac{d-1}{2}$ errors can be corrected, since the errors need to be corrected to the correct codeword. As shown in Figure 3.2, disjoint spheres of radius $\frac{d}{2}$ surround the codewords. If more than $\frac{d-1}{2}$ error occur, then it would be impossible to tell the correct codeword in which the message should be decoded into or that a message will be decoding to a wrong codeword.

Theorem 3.1.1. *If a code has a minimum distance of* $2m+1$*, then the code can correct* m *or fewer errors and detect* $2m$ *errors [2].*

Proof. Suppose that a code has minimum distance of $2m+1$. Consider a codeword $\mathbf{x}$ that is transmitted and let $\mathbf{z}$ be the received word, with at most m errors. Thus,

$d(\mathbf{x}, \mathbf{z}) \leq m$. Let $\mathbf{y}$ be another codeword such that $\mathbf{x} \neq \mathbf{y}$. By the triangle inequality:

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z})$$

This means that:

$$m + 1 = 2m + 1 - m \leq d(\mathbf{x}, \mathbf{y}) - d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{y}, \mathbf{z})$$

This shows that since $d(\mathbf{y}, \mathbf{z}) \geq m + 1$, $\mathbf{z}$ will be decoded as the correct codeword $\mathbf{x}$ instead of another codeword, $\mathbf{y}$. Additionally, for any received word $\hat{\mathbf{z}}$ with $d(\mathbf{x}, \hat{\mathbf{z}}) \leq 2m$, $\mathbf{y}$, we see that $\hat{\mathbf{z}}$ cannot be a codeword and $2m$ errors will be detected. □

A particularly important type of block codes are *linear code* which are subsets of $\mathbb{Z}_2^n$ that have group structure. Specifically, $C \subseteq \mathbb{Z}_2^n$ is linear if $\mathbf{0} \in C$ and for every $c_1, c_2 \in C$, we have $c_1 + c_2 \in C$, where addition is defined as adding the entries together, respectively. Linear codes have an important property that the minimum distance between any two code words is the same as the minimum weight of all nonzero codewords [3]. *Weight* is the amount of 1's in a given code. This follows because

$$d(c_1, c_2) = d(c_1 - c_2, 0) = minwt(c_1, c_2).$$

Note that $c_1 - c_2 = c_1 + c_2 \in C$.

Linear codes have the additional property that they are the null space of an $k \times n$ matrix over $\mathbb{Z}_2$. In other words, a linear code is the set of all $\mathbf{x} \in \mathbb{Z}_2^n$ that satisfy $H\mathbf{x}^T = \mathbf{0}$ where $H \in \mathbb{M}_{k \times n}(\mathbb{Z}_2)$. We call H a *parity-check matrix* of the corresponding linear code. Each row of H prescribes a parity check a codeword, i.e. a subset of entries that must have an even number of 1's.

Given the structure of linear codes, there must a efficient way to generate all of the codewords in the code. This is not efficiently done with the parity-check matrix. Instead, we turn to a *generator matrix*, G, of the code. Aptly named, G, generates all the codewords for a given code. Specifically, the rows of the generator matrix are basis vectors for a linear subspace C. Generator matrices and parity-check matrices are closely related. A generator matrix is called *systematic* if it has the form

$$G = [I_k|A]$$

for some $A \in M(\mathbb{Z}_2)$ (the set of all matrices with entries in $\mathbb{Z}_2$) [3]. In this case, the encoded message is visible in the first k bits of the codeword.The parity check matrix for the code generated by G then has the form

$$H = \left[A^T|I_{n-k}\right]$$

so that $HG^T = [0]$.

3.2 Reed-Solomon Codes

There are other codes that utilize algebraic structure. Polynomial codes are one such example. These codes still employ generator and parity-check matrices, however they capitalize on the fact that messages and/or codewords can be represented as polynomials, not just as vectors. Specifically, an $m-$tuple gives rise to a polynomial of degree $m-1$ by prescribing the coefficients of the polynomial. For example, the 5-tuple 11001 is represented by the polynomial $1 + x + 0x^2 + 0x^3 + x^4$ or $1 + x + x^4$.

At the fundamental level, Reed-Solomon codes are derived from evaluating poly-

nomials at specific points. Consider a finite field K, of degree m over $\mathbb{Z}_p$ that is generated by powers of α, the root of an irreducible polynomial in $\mathbb{Z}_p[x]$. That is to say $K = \mathbb{Z}_p(\alpha)$. Then, K contains p^n elements and if the minimum polynomial is in fact primitive, then α is a primitive element of K, meaning $K^* = \langle x \rangle$.

Now consider a polynomial $P(x) = a_0 + a_1x^1 + \ldots a_{k-1}x^{k-1}$ where each of the $a_i \in K$ and $k < p^m$. In other words, the coefficients of the polynomial are in K. We consider a Reed-Solomon code to be the mapping from the code generated by G $k-$tuple of the coefficients of $P(x)$ to the evaluation of $P(x)$ at the powers of α. That is to say: $(a_0, a_1, \ldots a_{k-1})$ maps to $(P(0), P(\alpha), P(\alpha^2) \ldots P(1))$, where $P(x) = a_0 + a_1x^1 + \ldots a_{k-1}x^{k-1}$. The $k-$tuple is encoded into a p^m-tuple which is then transmitted. Given a finite field $\mathbb{F}$, for each $k < n$ there is a Reed-Solomon code of dimension k and length $n = p^m$ that is defined by evaluation of polynomials of degree of at most $k-1$ at n points of the field $\mathbb{F}$ [6].

In order to decode, we must recover the original polynomial $P(x)$. Since $P(x)$ has degree $k-1$, it can be characterized by k points. This means that a system of k equations need to be solved out of the $n = p^m$ possible choices of equations:

$$\begin{aligned}
P(0) &= a_0 \\
P(\alpha) &= a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_{k-1}\alpha^{k-1} \\
P(\alpha^2) &= a_0 + a_1\alpha^2 + a_2\alpha^4 + \cdots + a_{k-1}\alpha^{2k-2} \\
&\cdots \\
P(1) &= a_0 + a_1 + a_2 + \cdots + a_{k-1}
\end{aligned}$$

The existence of all these additional equations provides redundancy so that errors in

transmission can be corrected. Observe that there are

$$\binom{p^m}{k}$$

choices of equations to solve and each give a prediction for $P(x)$.

An equivalent representation of Reed-Solomon codes is as a linear codes generated by a matrix of the form:

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & \alpha_1 & \alpha_1^2 & \dots & 1 \\ 0 & \alpha_2 & \alpha_2^2 & \dots & 1 \\ \vdots & & \vdots & & \vdots \\ 0 & \alpha_k & \alpha_k^2 & \dots & 1 \end{bmatrix}$$

Matrices of this form are known as Vandermonde matrices which have special proprieties such that each k columns are linearly independent [2]. Specifically, Reed-Solomon codes have the property that the Vandermonde determinant is nonzero [6]. This further ensures that any k equations will be linearly independent.

In the unlikely case that the transmitted message has no errors, we could have computed our original message, $(a_0, a_1, \dots a_{k-1})$,in $\binom{p^m}{k}$ different ways. However, any errors that occur within the transmitted p^m-tuple will inevitably affect the values of the a_i's when decoding. This is why it will be useful to have the redundancy of multiple systems of equations to solve.

If no errors have occurred, then any choice of k equations gives the correct value of $PX(x)$. With errors in transmission, different choices of equations may have different

predictions for $P(x)$ and thus it is necessary to perform multiple computations to determine the correct transmitted polynomial. We will see that that for s errors, the wrong $k-$tuple (the wrong prediction of a polynomial) can be calculated

$$\binom{s+k-1}{k}$$

ways. This will be useful in determining the number of errors a code can correct.

Lemma 1. *Given s errors in transmission, there are $\binom{s+k-1}{k}$ sets of equations that give incorrect predictions of $P(x)$ [6].*

Proof. Each equation of the form $P(\alpha^i)$ defines a hyper plane in K^k. Thus any choice of k equations yields of intersection of k hyperplanes, which corresponds to a $k-$tuple. By the linear independence of the equations $P(\alpha^i)$, it follows that these hyperplanes will meet at exactly one point [6]. This $k-$tuple is a prediction for the polynomial $P(x)$.

Since we have established linear independence from above, it is the case that these hyper planes will meet at only one point. For every choice of k equations that gives the correct polynomial and thus the correct $k-$tuple, all the hyperplanes must intersect at that same point.

For a wrong $k-$tuple, there are at most $s+k-1$ hyperplanes intersecting at a single wrong point, where s is the number of wrong equations coming from errors in the codeword, and $k-1$ equations are chosen from the remaining p^m-s correct equations. Note: if there were more than $k-1$ correct equations, the correct solution would be determined from their point of intersection. Thus, there can be at most

$$\binom{s+k-1}{k}$$

choices of equations from which an incorrect $k-$tuple would be determined. □

From the proof above, we see that any choice of k equations from the $p^m - s$ correct equations yields the correct $k-$ tuple. Thus, there are $\binom{p^m-s}{k}$ ways to calculate a correct $k-$tuple, and whenever

$$p^m - s > s + m - 1,$$

there will be more correct determinations than incorrect ones. Thus for correct decoding, we must have

$$s < \frac{p^m - k}{2}.$$

In other words, Reed-Solomon codes can correct up to $\frac{p^m-k}{2}$ errors.

This computation for error-correcting ability matches the fact that $s \leq \frac{d-1}{2}$ since the minimum distance, d, of Reed-Solomon codes is $p^m - k + 1$. Since each polynomial will have $k - 1$ roots, the minimum distance between any two Reed-Solomon codes is exactly $p^m - k + 1$. Since Reed-Solomon codes are linear codes, we know that the minimum distance between any two codewords is equal to the minimum weight of all the codewords. Note that the weight of a codeword is the number of nonzero entries. Also note that relatively small degree polynomials have few roots. Specifically, the number of roots in a degree $k-1$ polynomial is at most $k-1$. This means that there are at most $k-1$ zeros and thus makes the minimum weight $p^m-(k-1)$ or p^m-k+1.

The following is an example of a Reed-Solomon code of dimension $k = 3$ and length 16 over the field $K = \mathbb{Z}_2(\alpha)$ with 16 elements.

Example 3.2.1. Consider $K = \mathbb{Z}_2(\alpha)$ where α is a root of the polynomial $x^4+x+1 = 0$. Since this polynomial doesn't have any linear or quadratic factors, it is irreducible.

With this, we know K is a degree 4 extension over $\mathbb{Z}_2$ so $n = 4$ and $|K| = 16$. Moreover, the root α is in fact primitive, so that $K^* = \langle\alpha\rangle$.

Let $k = 3$ so each information vector has the form $(a_0, a_1, a_2) \in K^3$. Every such vector gives rise to the polynomial $P(x) = a_0 + a_1x + a_2x^2$. Then the encoding map is given by

$$(a_0, a_1, a_2) \to (P(0), P(\alpha), P(\alpha^2), \ldots, P(\alpha^{15}))$$

Since there are 16^3 information vectors, the Reed-Solomon code contains 16^3 codewords. The minimum distance between codewords is:

$$p^m - k + 1 = 16 - 3 + 1 = 14$$

This code can correct any pattern of 6 error or less since

$$s < \frac{p^m - k + 1}{2}$$

Chapter 4

The McEliece Cryptosystem

In this section, we analyze the original form the the McEliece cryptosystem first proposed in 1978 by R.J McEliece. This public-key cryptosystem exploits structure from algebraic coding theory, specifically introducing intentional errors in the encryption stage in order to ensure security. Specifically, the security of this cryptosystem relies on the fact recovering a random linear codeword from an unknown code is proven to be an NP- complete problem [4].

The McEliece system was originally built using Goppa codes. These codes are similar to Reed-Solomon codes as they are formed by evaluating functions at specific points. Specifically, Goppa codes are formed by evaluations of functions from the function field of an algebraic curve at points on that curve. These are a special class of algebraic geometry codes, which generalize Reed-Solomon codes. Note that Goppa codes have length $n = 2^m$ and dimension $k \geq n - sm$ where s is the maximum number of errors that the code can correct [7].

The first step in the construction of the McEliece cryptosystem is choosing a Goppa code of length $n = 2^m$, with dimension $k < n$ and error-correcting capability s such

that $k \geq n - sm$. This Goppa code is specified by a particular $k \times n$ generator matrix G.

Consider a $k \times n$ matrix G. For encryption, this matrix is scrambled using two matrices S and P. This scrambling effectively disguises G. S is a non-singular $k \times k$ matrix (the determinant of this matrix is nonzero) and P, is a permutation matrix of size $n \times n$. Note that S is non-singular; for the decryption algorithm to work, S needs to be invertible. A permutation matrix is a matrix that only has entries 0 and 1, with exactly one 1 in any given row and column. The purpose of P is to permute the entries of a given vector. The matrix P is chosen at random from among all permutation matrices. P is also invertible. G, S, and P must be kept private as they are the decryption key, while the public key is the generator matrix G' given by:

$$G' = SGP$$

together with the error correcting capability s. Note that the rows of G are permuted

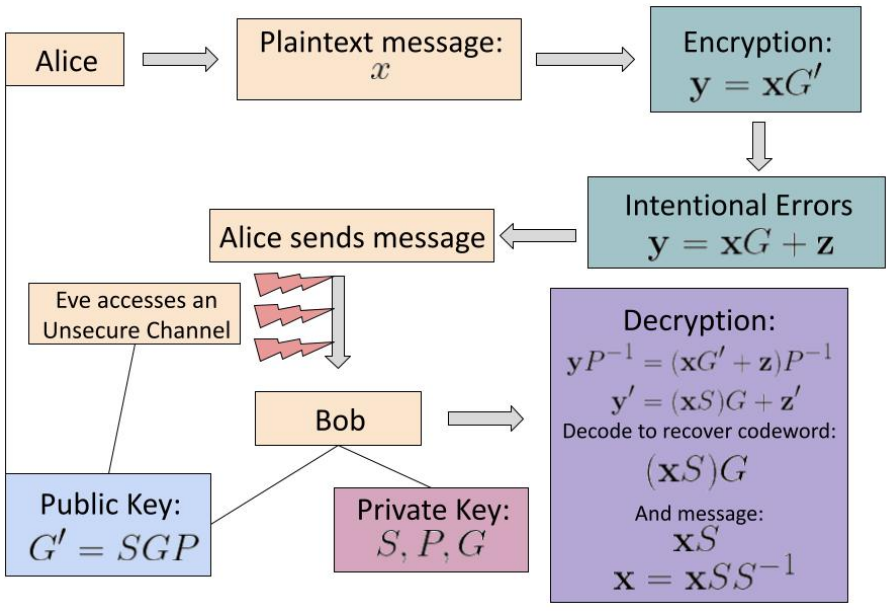

Figure 4.1: Flowchart detailing the McEliece cryptosystem

by P and the columns are scrambled by S. Thus, G is adequately scrambled to keep its identity hidden even when G' is known.

The encryption algorithm begins with a message to be transmitted that is broken up into k-bit blocks. Let $\mathbf{x}$ be such a block. The encryption of $\mathbf{x}$ is as follows.

$$\mathbf{y} = \mathbf{x}G' + \mathbf{z}$$

for some randomly generated vector $\mathbf{z} \in \mathbb{Z}_2^n$, with length n and weight s; $\mathbf{z}$ represents intentional errors begin added. Note that $\mathbf{x}G'$ gives a disguised codeword from the Goppa code that has been permuted and $\mathbf{y}$ is a noisy version of this vector which is then transmitted.

Once $\mathbf{y}$ is received, $\mathbf{x}$ can be recovered using the decryption algorithm. The first step is to compute $\mathbf{y}' = \mathbf{y}P^{-1}$. Observe that $\mathbf{y}'$ is a corrupted version of a codeword in the Goppa code, since:

$$\begin{aligned}
\mathbf{y}' &= (\mathbf{x}G' + \mathbf{z})P^{-1} \\
&= (\mathbf{x}G')P^{-1} + \mathbf{z}P^{-1} \\
&= \mathbf{x}SG + \mathbf{z}P^{-1} \\
&= (\mathbf{x}S)G + \mathbf{z}'
\end{aligned}$$

Note that $\mathbf{z}'$ is just a permutation of $\mathbf{z}$ since the inverse of a permutation matrix is also a permutation matrix and thus $\mathbf{z}'$ also has weight s. Therefore, $\mathbf{y}'$ is a corrupted codeword of the Goppa code, containing at most t errors. Since the code is known by the receiver, any decoding algorithm for the code, e.g. Patterson's algorithm can

be used to recover $\mathbf{x}S$ [7]. Then, since S is invertible and is part of the private key, $\mathbf{x}$ can be immediately recovered and the original message is obtained.

There are two potential source of attack on the McEliece cryptosystem. The first is to attempt to of recover S, P, and G from G'. The second is the brute force approach of constructing the original vector $\mathbf{x}$ from the transmitted vector $\mathbf{y}$.

It has been shown that both attacks are computationally intractable, ensuring the security of the cryptosystem. Specifically, by choosing n and s large enough, the first attack can be circumvented since there are too many possibilities for the matrices S and P to recover G by brute-force. Furthermore, since P is randomly generated, this increases the security and ensures that G' cannot be deconstructed. Since the rows of G are permuted by P and the columns are scrambled by S, G is adequately disguised.

The second attack, deconstructing $\mathbf{y}$ into the original message $\mathbf{x}$, is harder to account for. However, security against it is also enhanced by the size of n and s. This is because in order to effectively decode an intercepted vector $\mathbf{y}$, it will need to be compared to every codeword from a particular code, and the code is unknown to the eavesdropper. Since decoding a random codeword from an unknown linear code is NP-complete problem, a brute-force attack on the McEliece cryptosystem by attempting to decode $\mathbf{y}$ is computationally infeasible [7].

How do the two public-key cryptosystems, RSA and McEliece compare? It is clear that the foundation for these cryptosystems is different. RSA relies on number theory to form the foundation of its encryption and decryption schemes; using sufficiently large prime numbers as well as Euler's Theorem and the Extended Euclidean algorithm in order to encrypt and decrypt. In contrast, the McEliece cryptosystem pulls

from algebraic coding theory by using a specific linear code and its generator matrix in order to encrypt and decrypt. Despite these foundational differences, both systems are secure against both strategic attacks and brute-forces ones (as long as the decryption keys are kept secure).

But, there are two key computational differences between RSA and the McEliece cryptosystem that influence their ease in use: number of computations and key size. The McEliece cryptosystem has the advantage when it comes to computational speed because it doesn't require a lengthy encryption and decryption process. The computation needed to encrypt and decrypt is relatively short when compared to RSA. However, RSA has a smaller key size than McEliece; RSA requires that just a two numbers be published for a public key, with only one other number kept as the private key. On the other hand, the matrix G' for the McEliece cryptosystem is relatively large, due to the Goppa codes employed in the cryptosystem. This large key size is the primary reason that the McEliece cryptosystem is not more widely used despite it enhanced security against quantum attacks.

In order to overcome the large key-size of McEliece, it is important to explore other linear codes can be used in the implementation of the McEliece cryptosystem. Towards this end, a number of researchers have been exploring variations on the McEliece cryptosystem including the following variation.

4.1 Niederreiter Variation

One such variation of the McEliece cryptosystem is the Niederreiter variation. The key difference between the McEliece cryptosystem and the Niederreiter variation is

the use of a parity-check matrix rather than a generator matrix that is used in the original cryptosystem. Note that a parity-check matrix is a matrix H such that for every codeword $\mathbf{c} \in C$, H satisfies

$$H\mathbf{c}^T = \mathbf{0}.$$

One appeal of using the parity-check matrix over a generator matrix is the decoding capabilities. Rather than using an algorithm to solve a system of equations, syndrome decoding is used to decode from the parity-check matrix, which only requires a simple precomputed look-up table. In this setting, a plaintext message $\mathbf{x}$ is viewed as an error vector obtained from transmitting $\mathbf{0}$ and $H\mathbf{x}^T$ is the *syndrome* of $\mathbf{x}$ with respect to C. Then $\mathbf{x}$ can be recovered via syndrome decoding [8].

The encryption algorithm of the Niederreiter variation is similar to the McEliece cryptosystem, except that the plaintext message will serve as added noise rather than a random vector. To create the public key, two matrices, M and P are multiplied by H:

$$H' = MHP,$$

where M is a non-singular $(n-k) \times (n-k)$ matrix and P is a permutation matrix. Note that both M and P are invertible. Similar to the McEliece cryptosystem, H' is the public-key, while the individual matrices, M, H, and P are kept as the private-key.

A given plaintext message $\mathbf{x}$, will be encoded as

$$H'\mathbf{x}^T.$$

Note that the Niederreiter variation does not add intentional errors as part of the

encryption algorithm. This is, once again, due to the use of H as part of the structure of the Niederreiter variation. In this context, $\mathbf{x}$ is already considered an error-vector with respect to transmission of the zero vector, $\mathbf{0}$. Once the ciphertext is transmitted, the decryption algorithm is as follows:

$$\begin{aligned} M^{-1}(H'\mathbf{x}^T) &= M^{-1}MHP\mathbf{y}^T \\ &= H(\mathbf{x}P^T)^T \end{aligned}$$

Since $P\mathbf{x}^T = (\mathbf{x}P^T)^T$. Additionally since P is just a permutation matrix, $(\mathbf{x}P^T)^T$ still has the same weight as $\mathbf{x}$. Thus, we can use syndrome decoding on $H((\mathbf{x}P^T)^T)$,to obtain $\mathbf{x}P^T$. Finally, multiplying by $(P^T)^{-1}$ allows one to recover the original plaintext message $\mathbf{x}$.

The same attacks on the McEliece cryptosystem can be applied to the Niederreiter variation; namely attempting to deconstruct H' to the parts M, H, and P. These attacks are circumvented in similar ways, similarly to the McEliece cryptosystem. By choosing, q, n, and k sufficiently large enough, there are simply too many possibilities to brute-force recover M, H, and P in a reasonable amount of time. Additionally, the NP-problem of decoding a random linear codeword takes NP time, which still provides the needed security.

The Niederreiter variation enables more efficient decryption, but it still does not solve the issue of large key size. Thus, there are still a desire to use alternative code that can be represented more compactly, such as Reed-Solomon codes [8]. Reed-Solomon codes are still desirable to utilize within the Niederreiter variation due to the fact that they meet the singleton bound, meaning they have an optimal trade-off between their

rate and minimum distance, and thus between their efficiency and error-correcting capabilities [2]. However, in the next section, the use of Reed-Solomon codes in both the original McEliece cryptosystem and the Niederreiter variation have been susceptible to attacks.

4.2 Attacks on the McEliece cryptosystem and Variations with Reed-Solomon codes

Sidelnikov and Shestakov propose an attack on the McEliece cryptosystem when Reed-Solomon codes are used [9]. This attack is focused on reconstructing the individual matrices from the public-key; specifically the parity-check matrix will be recovered by solving multiple systems of equations. The attack was developed for the Niederreiter variation, however similar methods are used in order to attack and deconstruct the original McEliece as well [9].

Consider a finite field $\mathbb{F}_q$ and let $F = \mathbb{F}_q \cup \infty$. Solving for the matrices, M, H, and P is equivalent to solving

$$B = HU(x_1, x_2, x_3, \ldots, x_n; z_1, z_2, z_3, \ldots z_n)$$

where the matrix B is known and the matrix U has the form

$$U(x_1, x_2, x_3, \ldots, x_n; z_1, z_2, z_3, \ldots z_n) = \begin{bmatrix} z_1 & z_2 & z_3 & \ldots & z_n \\ z_1x_1 & z_2x_2 & z_3x_3 & \ldots & z_nx_n \\ z_1x_1^2 & z_2x_2^2 & z_3x_3^3 & \ldots & z_nx_n^3 \\ \vdots & & \vdots & & \\ z_1x_1^s & z_2x_2^s & z_3x_3^s & \ldots & z_nx_n^s \end{bmatrix}$$

for $x_i \in \mathbb{F}_q$ and $z_i \in \mathbb{F}_q \setminus 0$. Since the matrix B is known, there are two main steps to solve for the individual matrices, the first being solving for the x_i's [9]. This algorithm requires solving multiple systems of equations in order to derive the x_i's from $c_j \in \mathbb{F}_q$.

Once those values are found, now we can find the z_i's. For $z_1 = 1$ solve the equality

$$\sum_{j}^{s+2} = 1 c_j z_j x_j^i = 0$$

for $0 \leq i \leq s$, such that $c_j \in \mathbb{F}_q$ (not necessarily the ones used to solve the x_i's) and x_j^i are the found x_i's from the algorithm in [9]. In order to find the matrix H, one can solve

$$\sum_{k=0}^{s} h_{ik} x_j^k = z_j^{-1} b_{ij}.$$

This algorithm provides a method for recovering the parity-check matrix H, which means that the eavesdropper will be able to decrypt messages [9]. The original motivation for using Reed-Solomon codes was that they have sufficient structure such that it is possible to represent them more compactly and thus obtain a smaller key size. But as the attack demonstrates, this added structure can be exploited by the eavesdropper in order to recover the parity-check matrix H of the private key and crack the encryption. Thus, Reed-Solomon codes cannot be used within the Niederreiter variation and by extension the original McEliece cryptosystem (at least not directly) because they will introduce serious security risks.

Chapter 5

Conclusion

The mathematical structure underlying different encryption/decryption schemes is vitally important to understand as cryptography is essential to a number of modern day applications, *e.g.* online commerce and email communication. Public-key cryptography is particularly of interest because it enables wide-scale use for encryption. Compared to private-key cryptography, public-key employs an asymmetric algorithm to the encryption and decryption process so that the knowledge of the public-key does not give information away about the private-key.

The RSA cryptosystem is a widely used public-key cryptosystem. It's security is based in the computational intractability of factoring large numbers. However, the RSA cryptosystem will soon become obsolete with the wide-scale usage of quantum computers. Thus, a proposed alternative is the McEliece cryptosystem. As we have seen, this cryptosystem relies on an NP-complete problem from algebraic coding theory; specifically, the intractability of decoding random linear codes. The original proposal employs Goppa codes, which have been shown to be highly secure, but require a large public key making them impractical for everyday use. Thus, Reed-Solomon have been proposed as an alternative because their structure allows for a smaller pub-

lic key, while still maintaining high error-correcting capabilities. Reed-Solomon codes are constructed by evaluations of polynomials at the elements of certain finite fields, and they rely on polynomial interpolation for decoding. These codes can correct up to $\frac{d-1}{2}$ errors, where the minimum distance $d = n - k + 1$ and the code length n is the size of some extension field.

Unfortunately, a specific attack exploiting the structure of Reed-Solomon codes with the McEliece cryptosystem has shown the security risks of switching to these codes [9]. This is ultimately problematic as it excludes the straightforward use of Reed-Solomon codes within the McEliece cryptosystem. This motivates the need for future research involving the security of the McEliece cryptosystem with other alternative codes or better disguise of the Reed-Solomon codes moving forward.

Bibliography

[1] S. Landau. "Polynomials in the Nation's Service". In: *The American Mathematical Monthly* 111,2 (2004).

[2] D.W Hardy and C.L Walker. *Applied Algebra: Codes, Ciphers, and Discrete Algorithms*. Pearson Education, Inc., 2003.

[3] T.W Judson. *Abstract Algebra: Theory and Applications, Remixed!* Thomas W. Judson, 1997.

[4] E.R. Berlekamp, R.J. McEliece, and H.C.A Tilborg. "On the Inherent Intractability of Certain Coding Problems". In: *IEEE Transactions on Information Theory* 23,3 (1978).

[5] P.S. Menon and M. Ritwik. "A Comprehensive but not Complicated Survey on Quantum Computing". In: *IERI Procedia* 10 (144-52) (2014).

[6] I.S Reed and G. Solomon. "Polynomials over Certain Finite Fields". In: *Soc. Indust. Appl. Math* 8 (1960).

[7] R.J. McEliece. "A public-key Cryptosystem Based on Algebraic Cryptography". In: *DSN Progress Report* 42-44 (1978).

[8] H. Niederreiter. "Knapsack Type Cryptosystems and Algebraic Coding Theory". In: *Problems of Control and Information Theory* 15,2 (159-166) (1986).

[9] V.M. Sidelnikov and S.O. Shestakov. "On insecurity of cryptosystems based on generalized Reed-Solomon Codes". In: *Discrete Math Appl.* 2,4 (1992).