The McEliece Cryptosystem as a Solution to the Post-Quantum Cryptographic Problem

Isaac Hanna

A Senior Thesis submitted in partial fullfillment
of the requirements for graduation
in the Honors Program
Liberty University
Spring 2023

Acceptance of Senior Honors Thesis

This Senior Honors Thesis is accepted in partial
fullfillment of the requirements for graduation from the
Honors Program of Liberty University.

_____

Timothy E. Sprano, Ph.D.
Thesis Chair

_____

Robert Tucker, Ph.D.
Committee Member

_____

David E. Schweitzer, Ph.D.
Assistant Honors Director

_____

Date

Abstract

The ability to communicate securely across the internet is owing to the security of the RSA cryptosystem, among others. This cryptosystem relies on the difficulty of integer factorization to provide secure communication. Peter Shor's quantum integer factorization algorithm threatens to upend this. A special case of the hidden subgroup problem, the algorithm provides an exponential speedup in the integer factorization problem, destroying RSA's security. Robert McEliece's cryptosystem has been proposed as an alternative. Based upon binary Goppa codes instead of integer factorization, his cryptosystem uses code scrambling and error introduction to hinder decrypting a message without the private key. This cryptosystem cannot be reduced to the hidden subgroup problem and stands as a viable post-quantum alternative to RSA encryption.

The McEliece Cryptosystem as a Solution to the Post-Quantum Cryptographic Problem

Binary Goppa codes	Given a polynomial $G(x)$ of degree t with coefficients in F_{2^m} and a subset $(\alpha_0, ..., alpha_{n-1})$ where $G(\alpha_i) \neq 0 \forall i \in 0, 1, ..., n-1$, the code consisting of all vectors $C = (C_0, C_1, ..., C_{n-1}) \in V_n(F_2)$ satisfying $\sum_{i=0}^{n-1} \frac{C_i}{x-\alpha_i} \equiv 0 \pmod{G(x)}$ is a binary Goppa code
Superposition	The existence of a particle within multiple states. Example: $\|a\rangle = \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
Entanglement	Entanglement occurs when two particles are both in superposition and their states are inextricably linked. Example: $\frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$
Phase interference	When multiple phase shifts are applied to the same qubit the shifts can add or subtract as would occur with any other type of wave
Fourier basis	The Fourier basis is the basis in which values are encoded in the phase of the qubits instead of in the computational state.
Quantum Fourier transform	The Quantum Fourier Transform consists of Hadamard and *CNOT* gates and is used to convert from the computational basis to the Fourier basis
Hadamard gate	The Hadamard gate is used to put a qubit in a superposition
$CROT_k$ gate	The $CROT_k$ gate applies a rotation of $\frac{2\pi i}{2^k}$ radians to the second qubit if and only if the first is in state $\|1\rangle$
Hidden Subgroup Problem	Given a group G, a subgroup $H \leq G$, any set X, and a function $f : G \to X$ s.t. $f(a) = f(b) \iff aH = bH$, find H or a generating set of H.

In 1978, Ron Rivest, Adi Shamir, and Leonard Adleman proposed what has since become one of the premier cryptographic algorithms in use today. Pulling on the difficulty of factoring an integer, they created a system that allows for simple encryption with the public key and decryption with the private key, but an exponential-time decryption task without the private key. While still widely used today, the algorithm has begun to lose its hold because of a 1994 paper by Peter Shor in which he proposed an algorithm for factoring an integer in polynomial time on a quantum computer. In it he takes advantage of the peculiar nature of quantum mechanics–namely the wave nature of particles allowing for entanglement, superposition and phase interference. The McEliece cryptosystem, an alternative to RSA, utilizes binary Goppa codes instead of integer factorization. Taking advantage of error correction, the algorithm cannot be reduced to the same problem as RSA encryption and thus is more secure against a quantum attack.

Cryptography

The need to communicate information privately has sparked creativity for ages. If two people wanted to communicate without an intercepting third party being able to obtain the message, they needed some way of writing so that the interceptor couldn't understand it. Historically, this was done primarily by rearranging letters or substituting them for other letters or symbols. Most of these ciphers had a simple means of decoding the message, provided that both parties knew how it had been encoded. The security was in the fact that the enemy didn't know how it had been encoded. Such cryptosystems are referred to as symmetric or private key cryptosystems because the same technique is used to encrypt and to decrypt and either party would be able to do both—the same key is used for encrypting and decrypting. With the advent of

computers, much of our attention with regard to cryptography has focused on using mathematical functions to encrypt the bit-encoding of information that will be communicated (Singh, 2003). This poses a problem for traditional cryptosystems in that the two parties communicating may never have had a secure channel in which to decide on an encryption scheme. In 1976, Whitfield Diffie and Martin Hellman published a paper that has proven to be foundational in the development of modern cryptography. They defined the goal of a cryptosystem as "to make the enciphering and deciphering operations inexpensive, but to ensure that any successful cryptanalytic operation is too complex to be economical (p. 646) . They then proposed the idea of a public key cryptosystem, in which, instead of a single transformation $S_k : \{P\} \rightarrow \{C\}$ where $\{P\}$ is the set of possible unencoded messages and $\{C\}$ is the set of possible encoded messages, we have two transformations $E_k : M \rightarrow M$ and $D_k : M \rightarrow M$, where $\{M\} = \{P\} = \{C\}$ and the following properties hold given a key $k \in K$:

- $D_k = E_k^{-1}$
- E_k and D_k can be computed easily
- D_k cannot be feasibly derived from E_k, at least for most k
- E_k and D_k can be computed from k

(Diffie & Hellman, 1976).

RSA Encryption

The premier cryptosystem in use today was proposed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1978. Their algorithm made use of the difficulty of finding the prime

factorization of an integer. The encryption key k is made up of three integers e, n, and d. $E_k(M) \equiv M^e \pmod{n}$ and $D_k(C) = C^d \pmod{n}$. The key is calculated as follows. First, primes p and q are chosen and $n = pq$ is calulated. Next, we choose d, relatively prime to n and find e, the mutiplicative inverse of d modulo $\varphi(n)$ so that $ed \equiv 1 \pmod{\varphi(n)}$, where $\varphi(n)$ is Euler's totient funtion, giving the number of positive integers less than n that are relatively prime to n (Rivest et al., 1978). For a product of primes, $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$ can be calculated easily if p and q are known, but is as difficult as factoring n if they are not known (Miller, 1976; Stein, 2009). Euler showed that, for any M relatively prime to n, $M^{\varphi(n)} \equiv 1 \pmod{n}$ (Stein, 2009). Therefore $D_k(E_k(x)) \equiv (x^d)^e \pmod{n} \equiv x^{\varphi(n)+1}$. If $p \nmid x$ and $q \nmid x$, $\gcd(x, n) = 1$ and $x^{\varphi(n)+1} \equiv x \pmod{n}$. Thus knowing p and q allows the receiver of the message to quickly decrypt the message, while for a third party who intercepts the message, this task would be as difficult as factoring n into p and q–very difficult for a large n.

Quantum Mechanics

While RSA encryption is an excellent scheme in today's world, the advent of quantum computing threatens to upend this. Though quantum computing does not provide an exponential speedup across the board, the ability to put a quantum particle in a superposition, create an entanglement between values, and encode values in the phase of a particle allows us to create specific algorithms that provide an exponential speedup for certain problems–including that of factoring an integer. In order to understand how a quantum computer works, we first turn our attention to quantum mechanics.

Particle/Wave Duality

Prior to 1801, light was thought to consist of particles. In 1801, Thomas Young showed that it actually consists of waves (Feynman et al., 1964). He shone a light through a barrier with two slits in it onto another surface. If light consisted of particles, the pattern would be a smooth curve with peak at the middle of the surface. If light instead consisted of waves, the waves coming through the two slits would interfere, creating a pattern of alternating light and dark spots on the surface. Young found the second of these patterns, demonstrating that light does consist of waves.

In the late 1800s Heinrich Hertz observed that the ability of light to dislodge electrons from a metallic surface is dependent not on the intensity of light–which would be akin to the size of the wave if light is viewed as a wave– but on its frequency–the number of waves hitting it per second (Zubairy, 2020). This did not seem to fit with the wave nature of light, and so Einstein proposed a particle-based theory of light. This particle view, however, could not explain the wave-like nature that Young had observed .

Young's experiment has also been considered with electrons and the results are the same. Peculiarly, if we construct the experiment in such a way that we are unable to observe which path the electron or photon takes, we observe the wave pattern, even when firing them one at a time, suggesting that the electrons are passing through both holes and interfering with themselves, while if we construct it such that we can observe which hole the electron travels through, it only takes one path and the pattern we observe is the pattern we would expect with particles (Feynman et al., 1964). Observing the particles seems to impact how they behave (as a particle or a wave).

In 1929, De Broglie reconciled the two views of light by suggesting that the position of a

particle is modeled by a wave function with wavelength

$$\lambda = \frac{h}{p}$$

(De Broglie, 1929). In this equation, $h = 6.62607015 * 10^{-34}$ is Planck's constant and p is the particle's momentum. This wavelength only becomes significant on the quantum scale. With De Broglie's proposal and its subsequent verification came a new view of reality: every object has wave functions that describe the probability of observing its properties in particular states. When we observe a property, we collapse the wave function to a single value or smaller set of values but increase the uncertainty in another property. Given our classical understanding of the world, we would think that each of the properties already has a particular value, and we simply don't know what it is. John Bell proposed an inequality based upon this assumption and it has been shown experimentally that it does not hold (Aspect et al., 1982; Bell, 1964; Fry & Thompson, 1976). This means that the property actually is in multiple states at once, called a superposition of states.

Richard Feynman et. al (1964) described the behavior of electrons as being like particles which follow a wave function . This wave function is a superposition of locations and their intensities which collapses to a single location when we observe the electron. The intensities indicate the probability of observing the electron in that location. This explains the particle-like nature of an observed electron in the double-slit experiment—it really is shifting from being in multiple places at once to being in a single place. This concept of a superposition forms the foundation for quantum computing.

Particle Representation

Single Qubit Systems.

In the context of quantum computing, a particle is often referred to as a qubit–the quantum equivalent of a bit. A qubit's state can be represented as a vector of probability amplitudes

$$a = \begin{bmatrix} a_0 \\ \\ a_1 \end{bmatrix}$$

where a_0^2 is the probability amplitude of observing the qubit to be in state 0 and a_1^2 is the probability of observing it to be in state 1. The normalization property states that $a_0^2 + a_1^2 = 1$ because the sum of the probabilities has to be 1. As one might expect, we can also use different bases to measure a qubit. As a two-dimensional vector space, a basis can be defined by any two linearly independent vectors. A qubit that is in a definite state

$$\begin{bmatrix} 0 \\ \\ 1 \end{bmatrix} \; or \; \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}$$

in one basis will be in a superposition of states in any other basis. Three commonly used bases are the x, y, and z bases, listed below with their basis vectors, expressed in the z-basis (IBM, 2021).

$$z: \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} \; x: \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \\ 1 \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \\ -1 \end{bmatrix} \; y: \frac{1}{\sqrt{2}} \begin{bmatrix} i \\ \\ 1 \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} -i \\ \\ 1 \end{bmatrix}$$

Another commonly used notation is Dirac's bra-ket notation. It gives us a convenient notation for expressing the state of a qubit and its projection onto another state. First, the "ket" notation $|a\rangle = a_0|0\rangle + a_1|1\rangle$ gives the probability amplitudes of the basis vectors for a basis (Zubairy, 2020). The "bra" notation $\langle b|$is used in conjunction with the ket notation to represent the

projection of a onto b: $\langle b|a\rangle$. This is equivalent to an inner (dot) product $\langle b|a\rangle = \vec{a}\dot{b} = a_0b_0 + a_1b_1$ and gives the probability of finding a to be in state b when observed in b's basis. Clearly, if $b = a$, $\langle a|a\rangle = a_0^2 + a_1^2 = 1$ by the normalization property and if a is orthogonal to b, $\langle b|a\rangle = ba = 0$.

Multiple Qubits and Entanglement.

Though we have only considered single qubits, we can also work with multiple qubit systems. A pair of qubits have four discrete states that they can be in: 00, 01, 10, and 11, and of course can be in a superposition of these. The state is thus defined by the four probability amplitudes, often represented by a 4x1 vector as shown below.

$$a_{12} = \begin{bmatrix} p_{00} \\ p_{01} \\ p_{10} \\ p_{11} \end{bmatrix}$$

Similarly, an $n-$qubit system would be representd by 2^n complex densities.

Multi-qubit states can be classified as either separable or entangled. A state is separable if it can be expressed as the product of two distinct single-qubit states. For example, in

$$a_{12} = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)\text{: } a_1 = |0\rangle, a_2 = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

An entangled state, meanwhile, cannot be expressed as the product of two separable states.

$$\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$

is an example of such a state. The states of the two qubits are inextricably linked. If qubit a_1 is found to be a 1, we know that qubit a_2 will be a 0, while if qubit a_1 is found to be a 0, qubit a_2 will

be a 1. As discussed earlier, each individual qubit is in both of these states and there is nothing inherent to the qubit that dictates in which state it will be found. Thus the two particles are linked in some way that we do not understand.

Phase

In addition to a density for each of the states, a particle also has a relative phase shift between the two states. This is best understood with a unit sphere with $|0\rangle$ and $|1\rangle$ opposite one another. A unit semicircle would be sufficient to represent any combination of $|0\rangle$ and $|1\rangle$. The phase comes from rotating this semicircle 360 degrees. The phase can also be viewed akin to a phase shift in a sine wave. Two sine waves out of phase with one another would interfere constructively or destructively and so with a particle's phase shifts.

Quantum Computing

Now that we have laid the foundation of quantum mechanics, we will now consider the mechanisms that allow us to harness these properties for computational purposes.

Quantum Gates

Just as the quantum equivalent of a bit is a qubit, so too logical gates have a quantum equivalent that enables us to harness the power of superposition and entanglement for computing purposes. The simplest gates are single qubit gates that operate about a basis. These gates can be represented by a 2x2 matrix which can be multiplied by the qubit's state vector to find the resulting state of the qubit. For example, to rotate about the x, y, or z axes in the sphere defined

above we use the gates

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Applying these we find

$$Z|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} * \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \; X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \; Y|1\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} * \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -i \\ 0 \end{bmatrix}.$$

Notably, the basis vectors given earlier for each of these bases are the eigenvectors of the corresponding gate, meaning that they are unaffected by them. This makes sense because a vector falling along an axis would not be affected by a rotation about that axis (IBM, 2021). To create a superposition, we can use the Hadamard gate:

$$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H|1\rangle = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} * \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

We also can create gates for multiple qubits. As expected, these are 2^nx2^n gates. For example, the *CNOT* gate

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

performs an X gate on the second qubit if the first is a 1 and does nothing otherwise (IBM, 2021). Since this creates a dependency of the second qubit upon the value of the first, if the first is in a superposition, it creates an entanglement between them. Again, when a particle is in a

superposition, it is in both states, so the second particle will also be in both states until one or the other is observed.

Quantum Fourier Basis

Recall that a particle's state can be represented or measured in a variety of bases. One such basis, the Fourier basis, is a powerful tool for quantum computation and provides the foundation for Shor's algorithm. In the Fourier basis, every vector is an equally weighted superposition of every vector in the computational basis z, meaning that the probability of observing a given computational basis state is $\frac{1}{N}$ (Brun, 2017). A vector in the computational basis is encoded in the Fourier basis using a Quantum Fourier Transform (described below) which converts our values from being stored in a binary encoding to being stored in the phase of the particles. (Zhou et al., 2017). This is very powerful for performing computations and is the key to Shor's algorithm.

Shor's Algorithm

We now turn our attention to how quantum computing provides an exponential speedup for integer factorization, beginning with Shor's "trick"–the quantum Fourier transform.

Quantum Fourier Transform

As mentioned above, the Quantum Fourier Transform converts values that are encoded in binary in the computational basis to values that are encoded in the phase (equivalently in the Fourier basis) by rotating around the z-axis. Mathematically, we convert the state $|a\rangle$ to

$$\frac{1}{\sqrt{n}} \sum_{c=0}^{n-1} e^{\frac{2\pi i a c}{q}} |c\rangle$$

by applying the matrix with (a, c) entry

$$\frac{1}{\sqrt{q}} \sum_{c=0}^{n-1} e^{\frac{2\pi i a c}{q}}$$

(Shor, 1999). Practically, the Quantum Fourier Transform is performed by using a series of Hadamard and controlled rotation gates. The controlled rotation gate is represented by the following matrix:

$$CROT_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$$

Notably, this only performs the rotation on the state $|11\rangle$. If either qubit is in a superposition, this will cause an entangled state. Specifically, if the target qubit has already undergone a Hadamard gate, it is left in the state $\frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{2\pi i}{2^k}}|1\rangle$ if the control qubit is 1 and $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle$ if it is in the state 0 (IBM, 2021).

We apply a Hadamard gate to qubit 1, followed by $CROT_k$ gates from qubit j to qubit 1 with $k = j - 1$ for $j = 2...n$ (IBM, 2021). We then apply a Hadamard gate to qubit 2, followed by $CROT_k$ gates with $k = j - 2$ from qubits $j = 3...n$ to 2 . This pattern continues until we apply a Hadamard gate to qubit n. Finally, we must reverse the order of the qubits by swapping or reading them with reversed significance .

Finding the Period

Given an integer n to factor, we begin by checking whether it is even, a prime, or a power of a prime, because each of these cases is easy (Baumer et al., 2015; Shor, 1994). Assuming none

of the above checks yields a factor of n, we factor n by finding the period of a random x modulo n. This is the smallest r such that $x^r \equiv 1 \pmod{n}$, or equivalently $x^r = kn + 1$ for some $k \in \mathbb{Z}$. We select a random x and calculate $d = \gcd(x, n)$. If $d \neq 1$, d is a factor of n and we are done. If $d = 1$, we know that there exists an r such that $x^r \equiv 1 \pmod{n}$ (Stein, 2009). We then find q, the power of 2 with $n^2 < q < 2n^2$. Letting $l = \log_2(q)$, we will need 2 registers, each of at least l qubits for our state.(Baumer et al., 2015; Shor, 1999). We know that l qubits can encode up to $2^l = q$ numbers in binary so we can put our first register in a superposition of values from 0 to $q - 1$ by applying a Hadamard gate to each qubit. This leaves each qubit in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle$ and our system in the state

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0\rangle$$

since the second register is still in the state $|0\rangle$. Next, we create an entanglement between the two registers by performing the function $f(a) = x^a \pmod{n}$ (implemented by a series of gates) on the first register and placing the output in the second register. This leaves our system in the state

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|x^a \pmod{n}\rangle$$

After the registers have been entangled, a Quantum Fourier Transform is performed to encode our values of a in the Fourier basis. This leaves our system in the state

$$\frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} e^{\frac{2\pi i a c}{q}} |c\rangle|x^a \pmod{n}\rangle$$

Finally, the state is observed, yielding a value $(c, x^k \pmod{n})$. When the second register is observed and takes on a definite value, it collapses the state of the first to a superposition of those

values of a such that $f(x) = x^k \pmod{n}$. Our system is then in the state

$$\frac{1}{q}\sum_{a:x^a\equiv x^k}\sum_{c=0}^{q-1} e^{\frac{2\pi iac}{q}}|c\rangle|x^k \pmod{n}\rangle$$

which allows constructive interference to yield a higher probability of observing a desirable c (Shor, 1999). Once we have c and q, can use them to find r with good probability. There is at most one fraction $\frac{d}{t}$ with $k < n$ such that

$$\left|\frac{c}{q} - \frac{d}{t}\right| \leq \frac{1}{2q}$$

and it can be found using continued fraction expansion (Shor, 1994). If $\gcd(d, r) = 1$, this denominator will be r. This occurs $\varphi(r)$ times for each value of k in $x^a \equiv x^k \pmod{n}$, yielding $r\varphi(r)$ states that would provide us with the value of r. Shor showed that each of these states occurs with probability at least $\frac{1}{3r^2}$ so that we find r with probability $\frac{\varphi(r)}{3r}$. It is known that $\frac{\varphi(r)}{r} > \frac{\delta}{\log\log r}$ for some δ so the probability of finding r is $O(\frac{1}{\log\log r})$

Finishing the Factoring

Once we have r, we can use it to find a factor of n. We first note that $(x^{r/2} - 1)(x^{r/2} + 1) = x^r - 1 \equiv 0 \pmod{n}$ since $x^r \equiv 1 \pmod{n}$. By the fundamental theorem of arithmetic, we know that every prime factor of n must be a factor of either $x^{r/2} - 1$ or $x^{r/2} + 1$, except in the case that r is odd or that $x^{r/2} \equiv -1 \pmod{n}$. These cases occur only if the largest power of 2 in the orders of x with respect to each of n's prime factors are the same (Shor, 1999). This has probability $\frac{1}{2^{k-1}}$ so we will obtain a good value of x with probability $1 - \frac{1}{2^{k-1}}$, where k is the number of distinct prime factors of n . Since we have already confirmed that n is not a prime, nor a power of a prime, this occurs with a worst case probability of .5. Assuming that our x was a good choice, computing $\gcd(x^{r/2} - 1, n)$ and $\gcd(x^{r/2} + 1, n)$ will yield a factor of n

Shor's algorithm reduces a problem that is currently $O(e^{c(\log n)^{1/3}(\log\log n)^{2/3}})$ on a classical computer to $O((\log n)^2(\log\log n)(\log\log\log n))$ on a quantum computer. When expressed in terms of $k = \log n$, the length of the binary representation of n, this reduces $O(e^{k^{1/3}\log k^{2/3}})$ to $O(k^2(\log k)(\log\log k))$. In either case, this is an exponential speedup and renders RSA encryption insecure against a quantum attack. This failing motivates our evaluation of another algorithm that could avoid this pitfall.

McEliece Cryptosystem

In 1977 Robert McEliece proposed a cryptosystem based upon algebraic coding theory that can be shown not to be susceptible to the same attacks as RSA encryption. In order to understand the McEleice cryptosystem, we must first lay the algebraic background it rests upon.

Basic Algebra

We begin by giving a brief overview of fields and vector spaces.

Fields.

Definition 1 (Field). A field F is composed of a set of elements A together with two operations, 'addition' '$+$' and 'multiplication' '$*$' in which the following properties are satisfied.

a) $\exists 0 \in A$ s.t. $a + 0 = a \; \forall a \in A$

b) $\exists 1 \in A$ s.t. $a * 1 = a \; \forall a \in A$

c) $\forall a, b, c \in A$

i) $(a + b) \in A$

ii) $a + (b + c) = (a + b) + c$

iii) $a + b = b + a$

iv) $\exists -a \in A$ s.t. $a + (-a) = 0$

v) $(a * b) \in A$ **vii)** $a * b = b * a$

vi) $a * (b * c) = (a * b) * c$ **viii)** $\exists a^{-1} \in A$ s.t. $a * a^{-1} = 1$

For those familiar with algebra, a field is a commutative ring in which every nonzero element has a multiplicative inverse. We note here that every finite field will have cardinality equal to some power of a prime. Further, there is only one field of each cardinality, termed $GF(p^k)$ or F_{p^k} where p is prime and k a positive integer (Lidl & Niederreiter, 1996).

Definition 2 (Extension Field and Subfield)**.** Let F be a field, $K \subseteq F$, with K a field. Then K is called a subfield of F and F an extension field of K

Finally, we define the concept of a derivative in a general field.

Definition 3. In a field the formal derivative of a polynomial $f(x) = f_0 + f_1 x + ... + f_n x^n$ is a polynomial $f'(x) = f_1 + f_2 x + ... + n f_n x^{n-1}$.

s The following familiar properties of derivatives of functions of real numbers are easily verified for a general field.

Lemma 1. *Let f, g be polynomials over a field. The following relationships hold.*

a) $(f+g)' = f' + g'$

b) $(fg)' = fg' + f'g$

c) $(f^m)' = mf^{m-1}f'$

From these properties we can prove the following result

Theorem 1. *If* $f(x) = \prod_{i=1}^{r}(x-\beta_i)$, *then* $f'(x) = \sum_{i=1}^{r}\prod_{j=1,j\neq i}^{r}(x-\beta_j)$

The proof of this and all other theorems can be found in Appendix A.

Vector Spaces.

Given a field F, we construct a vector space of dimension n, labeled V_n by selecting n-tuples of elements from F. The operation '+' is extended to mean element-wise addition and we obtain the following definition.

Definition 4 (Vector Space)**.** A vector Space V_n of dimension n over a field F is the set of n-tuples $\mathbf{v} = (v_1, v_2, ..., v_n)$ where $v_1, v_2, ..., v_n \in F$, satisfying the following properties.

a) $\exists\ \mathbf{0} \in V$ s.t. $\mathbf{v} + \mathbf{0} = \mathbf{v}\ \forall \mathbf{v} \in V$

b) $\forall \mathbf{v}, \mathbf{u}, \mathbf{w} \in V, a, b \in F$

i) $(\mathbf{u} + \mathbf{v}) \in V$

ii) $(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$

iii) $\mathbf{v} + \mathbf{0} = \mathbf{v}$

iv) $\exists$ **-v** $\in V$ s.t. $\mathbf{v} + (\text{-}\mathbf{v}) = 0$

v) $a(b\mathbf{v}) = (ab)\mathbf{v}$

vi) $1\mathbf{v} = \mathbf{v}$

vii) $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$

viii) $(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$

Letting F_{p^k} be an extension field of F_p, F_{p^k} is also a vector space over F_p, namely $F_{p^k} = V_k(F_p)$. Again, the curious reader is referred to the appendix for the proof.

Theorem 2. *Let G be an extension field of K. G is a vector space over K via* $F_{p^k} = V_k(F_p)$

Algebraic Coding Theory

Having now established an algebraic foundation, we will investigate algebraic coding theory, starting with linear codes and then narrowing to binary Goppa codes–the codes upon which McEliece built his cryptosystem.

Linear Codes.

Definition 5 (Linear Code)**.** An (n, k) linear code is a k-dimensional subspace of an $n-$dimensional vector space $V_n(F_p) = (x_1, ..., x_n) : x_i \in F_p$. n is called the length of the code and k the dimension.

As a $k-$dimensional subspace, this code can be represented by k linearly independent vectors (a set of vectors in which no vector is a combination of the others).

Example 1 (Linear Code)**.** In $(Z_2)^8$, the vectors $(1,0,0,0,0,0,0,0)$, $(0,1,0,0,0,0,0,0)$, $(0,0,1,0,0,0,0,0)$, $(0,0,0,1,0,0,0,0)$ form a linearly independent basis for a 4-dimensional linear code of length 8 with codewords:

$(0,0,0,0,0,0,0,0)$ $(1,0,0,0,0,0,0,0)$ $(0,1,0,0,0,0,0,0)$ $(1,1,0,0,0,0,0,0)$

$(0,0,1,0,0,0,0,0)$ $(1,0,1,0,0,0,0,0)$ $(0,1,1,0,0,0,0,0)$ $(1,1,1,0,0,0,0,0)$

$(0,0,0,1,0,0,0,0)$ $(1,0,0,1,0,0,0,0)$ $(0,1,0,1,0,0,0,0)$ $(1,1,0,1,0,0,0,0)$

$(0,0,1,1,0,0,0,0)$ $(1,0,1,1,0,0,0,0)$ $(0,1,1,1,0,0,0,0)$ $(1,1,1,1,0,0,0,0)$

Combining these vectors as the rows of a matrix, we can define a generator matrix for a linear code.

Definition 6 (Generator Matrix)**.** A Generator Matrix G for an (n, k) linear code $\mathcal{C}$ over a field F is an $k \times n$ matrix whose rowspace is the code $\mathcal{C}$. In other words, the rows of G are the basis

vectors of the subspace C of $V_n(F)$. Conversely, if G is an $k \times n$ matrix of elements from F, G generates a k-dimensional code of length n over F.

Example 1 (cont.)**.** In the above example, our generator matrix would be

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

the 8 x 4 matrix whose rowspace is the code $\mathcal{C}$.

Binary Goppa Codes.

We now turn our attention to a subclass of linear codes, binary Goppa codes. First, we state the following lemma without proof.

Lemma 2. *For any polynomial $f(x) \in F_{2^k}[x]$ there exists $g(x) \in F_{2^k}[x]$ s.t. $f'(x) = g(x)^2$ (McEliece, 1977).*

To create a Goppa code, we first select a polynomial $G(x)$ with coefficients in F_{2^m} and a subset $(\alpha_0, \alpha_1, ..., \alpha_{n-1}) \subseteq F_{2^m}$ where $G(\alpha_i) \neq 0 \ \ \forall i \in 0, 1, ..., n-1$. Let t be the degree of G. Our code $\mathcal{C}$ consists of all vectors $c \in V_n(F_2)$ s.t.

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \equiv 0 \pmod{G(x)} \tag{1}$$

Theorem 3. *Let G(x) be an irreducible polynomial in F_{2^m} of degree $s > 1$. The Binary Irreducible Goppa code consisting of all vectors $C = (C_0, C_1, ..., C_{n-1} \in V_n(F_2)$ satisfying (1) is a linear code with $d_{min} \geq 2s + 1$.*

The mimimum distance is at least $2s + 1$ because were it not, we could construct a polynomial from the nonzero terms that is not identically zero and yet has degree less than that of $G(x)$ and is divisible by $G(x)$. This is a contradiction and so the result holds. See the appendix for the full proof.

It is important to note that the polynomial $G(x)$ has coefficients that are in F_{2^m} while the codewords are in $V_n(F_2) = F_{2^n}$. Here, n is an arbitrary integer equal to the cardinality of the set $\{\alpha_0, \alpha_1, ..., \alpha_{n-1}\}$, with $n \leq 2^m - 1$. For example, if $m = 3$, $n = 4$, we can let $G(x) = (1,0,0)x^3 + (0,1,0)x^2, (1,1,1)x + (0,1,1)$. Our specific codewords would depend upon how multiplication is defined, but an example would be $(0,1,1,0)$.

Theorem 4. *A linear code with minimum distance $d + 1$ can correct up to $\frac{d}{2}$ errors*

Theorem 5. *The Binary Irreducible Goppa code defined by (1) has dimension $k \geq n - ms$*

As this theorem is not critical to understanding the workings of the cryptosystem, we state it without proof. The curious reader can reference (McEliece, 1977) for the proof.

Definition 7. Given a Goppa code defined by an irreducible polynomial $G(x)$ of degree $s > 1$ in which codeword C is transmitted and R is received, we define the following

- $E = R - C$ is the error sequence
- $S(x) = \sum_{i=0}^{n-1} \frac{R_i}{x-\alpha_i} = \sum_{i=0}^{n-1} \frac{E_i+C_i}{x-\alpha_i} = \sum_{i=0}^{n-1} \frac{E_i}{x-\alpha_i}$
- $\beta = \alpha_i : E_i \neq 0$ is called the syndrome of the received message
- $\sigma(x) = \prod_{B \in \beta} \frac{E_b}{x-\beta}$ is the error-locator polynomial

- $e = |\beta|$ is the number of errors in the transmitted message

Here, β contains the locations in which there are errors.. Also $\deg(\sigma(x)) = e$ and

$$\sigma(x) = \begin{cases} 1 & \text{if } x \in \alpha_0, \alpha_1, ..., \alpha_{n-1} \\ 0 & \text{otherwise} \end{cases}$$

This allows $\sigma(x)$ to determine where there are errors in our received codeword. Clearly, in F_{2^n}, $E_i \in \{0, 1\}$ so once the error locations are found, we can recover C from R. This, coupled with the following theorem forms the foundation of our decoding algorithm.

Theorem 6. $\sigma(x)S(x) \equiv \sigma'(x) \pmod{G(x)}$

Because the Goppa code is a linear code, it can also be represented by a generator matrix G as defined above, consisting of k linearly independent codewords as its rows. This will become important in the following sections.

Encoding

We are finally ready to discuss the process of sending a message. Following convention, we will call our sender Alice, our receiver Bob and our interceptor Eve. First, Bob chooses an irreducible polynomial $G(x)$ of degree $s > 1$ with coefficients in F_{2^m} and finds the generator matrix G for the corresponding Goppa code using k linearly independent codewords from $V_n(F_2)$ as the row vectors. G has dimensions $k \times n$. Next, Bob chooses an n x n permutation matrix P having a singular 1 in each row and column and a "scrambler" matrix S such that S is invertible (equivalently $|S| \neq 0$). Bob computes $G' = SGP$ and publishes G' as his public key. Alice begins with a message $x \in V_k(F_2)$ and computes $m = xG' + z = xSGP + z$ where $z \in V_n(F_2)$ has weight s (s nonzero elements). Alice then transmits m to Bob.

Decoding

In order for this to be an effective cryptosystem, we need decoding to be simple if the private key is known, and difficult if it is not.

With a Private Key.

Our decoding algorithm is based upon knowing our generator polynomial $G(x)$. Once this is known we can use a pair of algorithms put forth by Euclid and Patterson to correct the errors in the message.

Euclid's Algorithm. In this section we put forth Euclid's algorithm for finding the greatest common divisor of two polynomials.

Algorithm 1 (Euclid's Algorithm)**.** We use the equation $R(x)t(x) + G(x)k(x) = r(x)$ where $\deg(G(x)) > deg(R(x))$ and start with $R(x)(0) + G(x)(1) = G(x)$ and $R(x)(1) + G(x)(0) = R(x)$ so that $t_{-1} = 0, t_0 = 1, k_{-1} = 1, k_0 = 0$.

1. $q_i(x) = \lfloor \frac{r_{i-2}(x)}{r_{i-1}(x)} \rfloor$
2. $r_i(x) = r_{i-2}(x) - q_i(x) * r_{i-1}(x)$
3. $t_i(x) = t_{i-2}(x) - q_i(x)t_{i-1}(x)$
4. $k_i(x) = k_{i-2}(x) - q_i(x) * k_{i-1}(x)$

Repeat until $r_i(x) = 0$ and go back one step to find the greatest common divisor of $G(x)$ and $R(x)$. (McEliece, 1977)

The appendix provides an analogous algorithm for integer factorization to aid in understanding this algorithm. While Euclid's algorithm is designed to find the greatest common

divisor of two polynomials, it can also be used to find $\alpha(x)$ and $\beta(x)$ in $R(x)\beta(x) + G(x)k(x) = \alpha(x)$. We will use the equation $G(x)k_i(x) + R(x)t_i(x) = r_i(x)$. One can observe that $t_i(x)$ is increasing in degree with each iteration while $r_i(x)$ is decreasing. The following theorem takes advantage of this.

Theorem 7. *Given two nonnegative integers μ and v with $v \geq deg[\gcd(a,b)]$, and $\mu + v = deg(a) - 1$, there exists a unique index j, $0 \leq j \leq n$ such that $deg(t_j) \leq \mu$ and $deg(r_j) \leq v$ (McEliece, 1977).*

This result guarantees a unique index j such that $\deg(t_j) \leq \frac{s-1}{2}$ and $\deg(r_j) \leq \frac{s}{2}$, provided $\frac{s-1}{2} \geq \deg(\gcd(R,G))$. Once we have found this index, let

$$\beta(x) = t_j(x), \qquad \alpha(x) = r_j(x)$$

so that

$$\beta(x)R(x) + k_j(x)G(x) = \alpha(x) \implies \beta(x)R(x) \equiv \alpha(x) \pmod{G(x)}$$

We solve this for $\beta(x)$ and $\alpha(x)$ and calculate $\sigma(x)$ using

$$\sigma(x) = \alpha^2(x) + x * \beta^2(x)$$

In the next section we show how to find $R(x)$ and combine these results to create a decoder algorithm.

Patterson's Algorithm. The following algorithm was originally presented by N.J. Patterson (1975) and is used by McEliece to find $R(x)$ and ultimately $\sigma(x)$.

Algorithm 2 (Compute $\sigma(x)$)**.**

1. Start with S(x) and G(x)

2. Compute $T(x)$ s.t. $T(x)S(x) \equiv 1 \pmod{G(x)}$

3. Calculate R(x) s.t. $[R(x)]^2 \equiv T(x) + x$

4. Use Euclid's algorithm to solve for $\beta(x), \alpha(x)$ in $\beta(x)R(x) \equiv \alpha(x) \pmod{G(x)}$

5. Compute $\sigma(x) = \alpha^2(x) + x\beta^2(x)$

Again, once we have obtained $\sigma(x)$, we can use $\sigma(\alpha)$ for each $\alpha \in \alpha_0, \alpha_1, ..., \alpha_{n-1}$ to determine the positions in C which have errors.

Pulling it Together. Once Bob has received the transmitted message m, he computes $m' = mP^{-1} = (xSGP + z)P^{-1} = xSGPP^{-1} + zP^{-1} = xSG + z'$ where $z = zP^{-1} \in V_n(F_2)$ is still of weight s. As shown above, Patterson's algorithm can detect up to s errors in a codeword. This can be used to find $u = xSG$. Bob decodes this to $x' = xS$ and uses $x = x'S^{-1}$ to find x.

Without the Private Key.

Now let's consider the case of our interceptor Eve. Eve will see G' sent to Alice and m sent back to Bob. In order to determine x, Eve would have to somehow correct the errors, depermute the remaining vector, decode it to a $k-$dimensional vector, and finally unscramble x'. The difficulty of these tasks makes the algorithm secure (McEliece, 1978).

Hidden Subgroup Problem

To show that the McEliece cryptosystem is not susceptible to quantum attack would require showing that no possible attack using any algorithm would reduce the problem to polynomial time. Even for classical attacks this is essentially impossible to show. What can be

shown is that currently developed quantum algorithms–for example the one that leads to the downfall of RSA–are not capable of taking down the McEliece system. The approach used by Shor's algorithm to crack the problem of integer factorization is known more generally as the quantum Fourier sampling algorithm for solving the hidden subgroup problem.

Definition 8 (Hidden Subgroup Problem)**.** Given a group G, a subgroup $H \leq G$, any set X, and a function $f : G \to X$ s.t. $f(a) = f(b) \iff aH = bH$, find H or a generating set of H.

The function f essentially splits H over X into cosets (subsets of a group that are formed by "multiplying" the group by one of its elements) by mapping elements to the same element in X if and only if the cosets they generate are equal (Wang, 2010).

Reduction of Shor's Algorithm to the Hidden Subgroup Problem

We can reduce Shor's algorithm to the hidden subgroup problem as follows. Let G be the set of positive integers modulo q, the smallest power of 2 s.t. $q \geq n^2$. Let $X = H$ be the set of powers of r modulo q and $f : G \to X$ by $f(a) = x^a \pmod{n}$. The quantum Fourier transform then allows us to observe an element that will be useful in finding H. One may object that the second register is not observed until after the QFT has been performed but this distinction does not matter because of the concept of delayed choice–an observation can be made later and previous operations performed on entangled particles will result as though the observation had been performed prior to them (Hallgren et al., 2003; Kim et al., 2000). Thus the core of Shor's algorithm is actually the hidden subgroup problem and the corresponding quantum sampling algorithm.

Irreducibility of the McEliece Cryptosystem to the Hidden Subgroup Problem

The hidden subgroup problem and the quantum Fourier sampling algorithm for solving it make up one of the premier approaches to solving NP problems in polynomial time on a quantum computer. Dinh, Moore, and Russell (2018) studied the application of this algorithm to breaking the McEliece cryptosystem. While the details of their analysis are beyond the scope of this paper, they proved that it would not be possible to break McEliece encryption using the hidden subgroup problem because the McEliece system uses a subgroup that is indistinguishable, meaning that it cannot be distinguished from other subgroups of the group G.

Other Attacks on the McEliece Cryptosystem and Adaptations

While the McEliece cryptosystem is not susceptible to a hidden subgroup attack, other attacks have been proposed, leading to the development of variations on the cryptosystem to protect against these attacks. Most attacks take advantage of a weakness in the private key, often resulting from an attempt at increasing efficiency. For example, using a generator polynomial with coefficients from F_2 has been shown to be insecure because one can use a modified brute force attack that reduces the problem to polynomial time (Loidreau & Sendrier, 2001). Similarly, if Reed-Solomon codes are used instead of Goppa codes, the problem can again be reduced to polynomial time (Engelbert et al., 2007). Still, the McEliece cryptosystem is a viable alternative to the RSA and Diffie-Helman protocols with the advent of quantum computing.

Conclusion

RSA encryption–today's premier encryption system–is based on the difficulty of factoring an integer, a problem that Peter Shor reduced to polynomial time on a quantum computer by

taking advantage of the ability to put a particle in a superposition, entangle it with other particles, and encode values in the phase of the particle. Pulling on his background in information theory and coding, Robert McEliece developed a cryptosystem that ultilizes error correction by intentionally introducing errors to a message. If the errors were not introduced, the eavesdropper could simply undo the encoding and obtain the message, but by adding errors alongside scrambling, permutation, and encoding, he created a system in which the eavesdropper would either have to crack a linear code with errors or decode a matrix into three matrices. This allows Bob to generate a public key G' and broadcast this to Alice, who encrypts her message and sends it back to Bob. Bob can correct the errors and decode the message. The algorithm is built upon a substantial foundation of algebra and provides a fascinating application of what can seem an abstract, impractical subject. Most significantly, the problem of cracking McEliece encryption cannot be reduced to the hidden subgroup problem, meaning that it cannot fall to the same trick to which RSA encryption fell. Further research can explore ways to make this more secure or reduce the cost of transmitting a message with this system.

References

Aspect, A., Dalibard, J., & Roger, G. (1982). Experimental test of Bell's inequalities using time-varying analyzers. *Physical Review Letters*, *49*(25), 1804–1807. https://doi.org/10.1103/PhysRevLett.49.1804

Baumer, E., Sobez, J.-G., & Tessarini, S. (2015, May 15). *Shor's algorithm* [PowerPoint slides]. https://qudev.phys.ethz.ch/static/content/QSIT15/Shors%5C%20Algorithm.pdf

Bell, J. (1964). On the Einstein Podolsky Rosen paradox. *Physics*, *1*(3), 195–200. https://cds.cern.ch/record/111654/files/vol1p195-200_001.pdf

Brun, T. (2017). *Lecture 13* [PowerPoint slides]. University of Southern California. https://viterbi-web.usc.edu/~tbrun/Course/lecture13.pdf

De Broglie, L. (1929, December 12). *The wave nature of the electron* [Speech transcript]. The Nobel Prize. https://www.nobelprize.org/prizes/physics/1929/broglie/lecture/

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. https://doi.org/10.1109/TIT.1976.1055638

Engelbert, D., Overbeck, R., & Schmidt, A. (2007). A summary of McEliece-type cryptosystems and their security. *Journal of Mathematical Cryptology*, *1*(2), 151–199. https://doi.org/10.1515/JMC.2007.009

Feynman, R., Leighton, R., & Sands, M. (1964). *Feynman lectures on Physics Vol 3*. Addison Wesley. https://www.academia.edu/30257311/Feynman_Lectures_on_Physics_Vol_3_Feynman_Leighton_and_Sands_1964_

Fry, E. S., & Thompson, R. C. (1976). Experimental test of local hidden-variable theories. *Physical Review Letters*, *37*(8), 465–468. https://doi.org/10.1103/PhysRevLett.37.465

Hallgren, S., Russell, A., & Ta-Shma, A. (2003). The hidden subgroup problem and quantum computation using group representations. *SIAM Journal on Computing*, *32*(4), 19. https://doi.org/10.1137/S009753970139450X

IBM. (2021). *IBM Quantum*. https://quantum-computing.ibm.com/

Kim, Y., Yu, R., Kulik, S., Shih, Y., & Scully, M. (2000). Delayed “choice” quantum eraser. *Physical Review Letters*, *84*, 1–5. https://doi.org/10.1103/PhysRevLett.84.1

Lidl, R., & Niederreiter, H. (1996). *Finite Fields* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511525926

Loidreau, P., & Sendrier, N. (2001). Weak keys in the McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, *47*(3), 1207–1211. https://doi.org/10.1109/18.915687

McEliece, R. (1977). *Theory of information and coding*. Addison Wesley. https://archive.org/details/theoryofinformat03mcel/page/162/mode/2up

McEliece, R. (1978). A public-key cryptosystem based on algebraic coding theory. *The Deep Space Network Progress Report*, *42*(44), 114–116. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF

Miller, G. L. (1976). Riemann’s hypothesis and tests for primality. *Journal of Computer and System Sciences*, *13*(3), 300–317. https://www.sciencedirect.com/science/article/pii/S0022000076800438?ref=pdf_download&fr=RR-2&rr=7b7e742b58e381af

Patterson, N. (1975). The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory*, *21*(2), 203–207. https://doi.org/10.1109/TIT.1975.1055350

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120–126. https://dl.acm.org/doi/pdf/10.1145/359340.359342

Shor, P. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, *41*(2), 303–332. https://www.jstor.org/stable/2653075

Shor, P. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134. https://doi.org/10.1109/SFCS.1994.365700

Singh, S. (2003). The history of cryptography: How the history of codebreaking can be used in the mathematics classroom. *Mathematics in School*, *32*(1), 2–6.

Stein, W. (2009). *Elementary number theory: Primes, congruences, and secrets*. Springer New York. https://doi.org/10.1007/b13279

Wang, F. (2010, July). *The Hidden Subgroup Problem* [Master's Project, Aarhas Universitet]. Arxiv. https://arxiv.org/ftp/arxiv/papers/1008/1008.0010.pdf

Zhou, S. S., Loke, T., Izaac, J. A., & Wang, J. B. (2017). Quantum Fourier transform in computational basis. *Quantum Information Processing*, *16*(3), 82. https://doi.org/10.1007/s11128-017-1515-0

Zubairy, M. S. (2020). *Quantum mechanics for beginners: With applications to quantum communication and quantum computing*. Oxford University Press. https://doi.org/10.1093/oso/9780198854227.001.0001

Appendix A: Proofs and Examples

In this appendix we prove several of the theorems used throughout the paper. Recall the definition of the formal derivative of a polynomial:

Definition 3. In a field the formal derivative of a polynomial $f(x) = f_0 + f_1x + ... + f_nx^n$ is a polynomial $f'(x) = f_1 + f_2x + ... + nf_nx^{n-1}$.

It is easy to verify the following.

Lemma 1. *Let f, g be polynomials over a field. The following relationships hold.*

a) $(f+g)' = f' + g'$

b) $(fg)' = fg' + f'g$

c) $(f^m)' = mf^{m-1}f'$

We now use these relationships to prove the following result.

Theorem 1. *If $f(x) = \prod_{i=1}^{r}(x-\beta_i)$, then $f'(x) = \sum_{i=1}^{r}\prod_{j=1,j\neq i}^{r}(x-\beta_j)$*

Proof. Letting $f_i(x) = (x-\beta_i), f(x) = \prod_{i=1}^{r} f_i(x)$. Note that $f_i'(x) = 1$. Then by repeated application of b) above,

$f'(x) = \sum_{i=1}^{r} f_i'(x)\prod_{j=1,j\neq i}^{r} f_j(x) = \sum_{i=1}^{r}\prod_{j=1,j\neq i}^{r} f_j(x)$. □

Theorem 2. *Let G be an extension field of K. G is a vector space over K via $F_{p^k} = V_k(F_p)$*

Proof. $G = F_{p^k}$ for some prime p and positive integer k and likewise $K = F_{q^j}$ for some prime q and positive integer j. Because $[G:K]$ is a positive integer, call it n, and $p^k = (q^j)^{[G:K]}$, with p and

q prime, it follows that $p = q$ and $k \mid j$ so that $\frac{j}{k} = [G : K] = n$. Then we can represent the elements of G as $g = (g_0, g_1, g_2, ..., g_{n-1})$, where $g_i \in K$ and $g = g_0 + p * g_1 + p^2 * g_2 + ... + p^{n-1} * g_{n-1}$. Defining addition on G to be element-wise, we see that closure, associativity, and inverses all hold, and the element containing all 0's is the identity. As for the other properties,

- $1\mathbf{g} = 1(g_0, g_1, g_2, ..., g_{n-1}) = (1g_0, 1g_1, 1g_2, ..., 1g_{n-1}) = (g_0, g_1, g_2, ..., g_{n-1}) = \mathbf{g}$
- $a\mathbf{(g+h)} = a(g_0+h_0, g_1+g_1, ..., g_{n-1}+h_{n-1}) = (a(g_0+h_0), a(g_1+h_1), ..., a(g_{n-1}+h_{n-1})) = (ag_0 + ah_0, ag_1 + ah_1, ..., ag_{n-1} + ah_{n-1}) = a\mathbf{g} + a\mathbf{h}$
- $(a+b)\mathbf{g} = ((a+b)g_0, (a+b)g_1, ..., (a+b)g_{n-1}) = (ag_0 + bg_0, ag_1 + bg_1, ..., ag_{n-1} + bg_{n-1}) = a\mathbf{g} + b\mathbf{g}$

Therefore G is a vector space over K and if $K = F_{p^k}$, $G = F_{p^{k*[G:K]}}$ □

Theorem 3. *Let G(x) be an irreducible polynomial in F_{2^m} of degree $s > 1$. The Binary Irreducible Goppa code consisting of all vectors $C = (C_0, C_1, ..., C_{n-1} \in V_n(F_2)$ satisfying (1) is a linear code with $d_{min} \geq 2s + 1$.*

The following proof is based on an outline from exercise 8.17 of (McEliece, 1977)

Proof. First, note that the minimum distance between codewords will always be equal to the weight of the smallest codeword since the sum (or equivalently difference) of any two codewords is also a codeword and the zero vector is a codeword. This simplifies the proof to showing that noe element with weight $r \leq 2s$ exists. We proceed by contradiction. Assume the code has a

codeword of weight $r \leq 2s$. Let $\beta_0, \beta_1, ..., \beta_{r-1}$ be the α_i corresponding to the r positions of the codeword containing 1s. Then $\sum_{i=0}^{r-1} \frac{1}{x-\beta_i} \equiv 0$. Letting $p(x) = \sum_{i=0}^{r-1} \prod_{j \neq i}(x - \beta_j)$ and $q(x) = \prod_{i=0}^{r}(x - \beta_i)$, $\frac{p(x)}{q(x)} = \sum_{i=0}^{r-1} \frac{1}{x-\beta_i} \equiv 0 \pmod{G(x)}$. But we know that $G(\beta_j) \neq 0 \ \forall j$ so that $G(x)$ and $q(x)$ are relatively prime and thus $p(x) \equiv 0 \pmod{G(x)}$. By Lemma 2 $p(x) = g(x)^2$ for some $g(x) \in F_{2^m}[x]$. Since $G(x)$ is irreducible, $p(x) \equiv 0 \implies g(x) \equiv 0 \pmod{G(x)}$. But $p(\beta_i) = \sum_{i=0}^{r} \prod_{j \neq 1}(\beta_i - \beta_j) \neq 0$ so that $p(x)$ is not identically 0. $g(x) \neq 0$ because $0 * 0 = 0$. But $\deg(g(x) \leq s) = \deg(G(x))$, so this is a contradiction. Therefore $d_{min} \geq 2s + 1$. □

Theorem 4. *A linear code with minimum distance $d + 1$ can correct up to $\frac{d}{2}$ errors*

Proof. Let x be a codeword and e be the error vector with weight $w \leq \frac{d}{2}$. $r = x + e$ is the received codeword. Letting x' be the nearest codeword other than x, $r - x'$ has weight $\geq \frac{d}{2} + 1$. Therefore x is the nearest codeword. □

Theorem 6. $\sigma(x)S(x) \equiv \sigma'(x) \pmod{G(x)}$

Proof. $\sigma'(x) \equiv \sum_{B \in \beta} \prod_{\gamma \in \beta, \gamma \neq B} (x - \gamma) \equiv \sum_{B \in \beta} [\frac{\prod_{\gamma \in \beta}(x-\gamma)}{x-B}] \equiv \prod_{\gamma \in \beta}(x - \gamma) \sum_{B \in \beta} \frac{1}{x-B} \equiv \sigma(x)S(x) \pmod{G(x)}$ □

Theorem 7. *Given two nonnegative integers μ and v with v with $v \geq deg[\gcd(a, b)]$, and $\mu + v = deg(a) - 1$, there exists a unique index j, $0 \leq j \leq n$ such that $deg(t_j) \leq \mu$ and $deg(r_j) \leq v$.*

Proof. $\deg(t_i) + \deg(r_{i-1}) = \deg(G)$ because this result holds for the first pair and $\deg(t_i)$ increases by the same amount that $\deg(r_{i-1})$ decreases iteration. Since r_{i-1} is a decreasing sequence and t_i increasing, we can find j such that $\deg(r_{j-1}) \geq v + 1$, $deg(r_j) \leq nu$, with $\deg(t_j) \leq \mu$, and $\deg(t_{j+1}) \geq \mu + 1$ following from $\deg(t_i) + \deg(r_{i-1}) = \deg(G) - 1$. □

Example 2. To aid in understanding this algorithm, we present an example of Euclid's greatest common divisor algorithm for integers–a special case of this algorithm. Find $\gcd(10, 24)$:

$t_{-1} = 0, k_{-1} = 1, t_0 = 1, k_0 = 0$

so that $10(0) + 24(1) = 24$ and $10(1) + 24(0) = 10$

$q_1 = \frac{24}{10} = 2, r_1 = 24 - 2 * 10 = 4, t_1 = 0 - 2 * 1 = -2, k_1 = 1 - 2 * 0 = 1.$

Now we have $24(1) + t(-2) = 4$. Repeating this process,

$q_2 = \frac{10}{4} = 2, r_2 = 10 - 2 * 4 = 2, t_2 = 1 - 2 * (-2) = 5, k_2 = 0 - 2 * 1 = -2$

so that $24(-2) + 5(10) = 2$.

Finally, $q_3 = \frac{4}{2} = 2, r_3 = 4 - 2 * 2 = 0$ so we are finished.

Therefore $\gcd(10, 24) = 2$.

The polynomial algorithm functions similarly, except that each integer is replaced by polynomials.

Algorithm 2 (Compute $\sigma(x)$)**.** 1. Start with $S(x)$ and $G(x)$

2. Compute $T(x)$ s.t. $T(x)S(x) \equiv 1 \pmod{G(x)}$

3. Calculate $R(x)$ s.t. $[R(x)]^2 \equiv T(x) + x$

4. Use Euclid's algorithm to solve for $\beta(x), \alpha(x)$ in $\beta(x)R(x) \equiv \alpha(x) \pmod{G(x)}$

5. Compute $\sigma(x) = \alpha^2(x) + x\beta^2(x)$

The following proof is based on an outline from (McEliece, 1977)

Proof. We have already shown how Euclid's algorithm can find $\alpha(x)$ and $\beta(x)$. It remains to show that the $R(x)$ computed as such is the correct $R(x)$ to find $\alpha(x)$ and $\beta(x)$. Let $G(x)$ be an irreducible polynomial of degree $s > 1$ in $F_{2^k}[x]$ defining a Goppa code. Assume we have just received a codeword $R \in F_{2^k}$. Compute the syndrome $S(x) = \sum_{i=0}^{n-1} \frac{E_i}{x-\alpha_i}$. $\sigma'(x) = \beta^2(x)$ for some $\beta(x) \in F_{2^m}[x]$ by Lemma 1. Therefore $\sigma(x) = \alpha^2(x) + x * \beta^2(x)$. Since $\deg(\sigma(x)) \leq s$, $\deg(\alpha(x))$ $\leq \frac{s}{2}$ and $\deg(\beta(x)) \leq \frac{s-1}{2}$. Let $T(x)$ be the unique polynomial of degree $< k$ s.t. $S(x)T(x) \equiv 1 \pmod{G(x)}$. $\sigma(x)S(x) = \sigma'(x)$

$$\implies \alpha^2(x)S(x) + x\beta^2(x)S(x) \equiv \beta^2(x) \pmod{G(x)}$$

$$\implies \alpha^2(x)S(x)T(x) + x\beta^2(x)S(x)T(x) \equiv \beta^2(x)T(x) \pmod{G(x)}$$

$$\implies \alpha^2(x) + x * \beta^2(x) \equiv \beta^2(x)T(x) \pmod{G(x)}$$

$$\implies [T(x) - x]\beta^2(x) \equiv \alpha^2(x) \pmod{G(x)}.$$

Because we are in $F_{2^m}[x]$, $-x = x$ so $[T(x) + x]\beta^2(x) \equiv \alpha^2(x)$. $\exists$ unique $R(x) \in F_{2^m}[x]$ s.t. $R^2(x) \equiv T(x) + x$. Therefore $R(x)\beta(x) \equiv \alpha(x)$. □