Cryptography in a Post-Quantum World

Study Report

Sean Fox
University of North Carolina at Charlotte
Mint Hill, North Carolina
sfox33@uncc.edu

ABSTRACT

This paper discusses the effect of Quantum computing on the field of modern cryptography. Many of the most popular public key-based cryptographic protocols can be broken by the new computational abilities of quantum computers. For example, Shor's algorithm is a famous quantum-computing algorithm which can find the prime factorization of number in polynomial times. This alone breaks many famous and popular cryptography protocols such as RSA and Diffie-Hellman. This paper examines several categories of cryptographic protocols and analyzes which classes will be breakable by quantum computers. With many of the most popular cryptographic protocols broken by quantum computers, researchers are already looking for new quantum-safe protocols to start phasing into the public before quantum computers reach their full potential. The second part of this paper serves to create a survey of the many different areas of research that are currently being examined such as lattices, multivariate functions, hash functions, and code-based algorithms to find new cryptographic protocols that will still be secure with the arrival of quantum computers.

ACM Reference Format:
Sean Fox. 2018. Cryptography in a Post-Quantum World: Study Report. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The modern age of computers has its own space race: the race to build the first high-powered quantum computer. While the computer science community has been excited over the potential brought about by quantum computers, they have been a large source of stress for the security analysts. RSA, Diffie-Hellman, and many other popular public key protocols and cryptosystems can and will be broken be quantum computers. With how widespread these protocols are in application, there needs to be a large change in what protocols are used by the public as soon as possible.

However, even though the most popular protocols will be broken, there are other options. Throughout this paper, we will examine how RSA and Diffie-Hellman can be broken by quantum computers and what it takes to be a quantum-resistant cryptographic protocol. We will also examine the categories of algorithms that are being researched to protect against quantum computers.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND INFORMATION AND TERMINOLOGY

2.1 Mathematics

While the focus of this paper is not on mathematics, it would be very difficult to talk about quantum computing without first reviewing the symbols and terminology used to describe quantum spaces and states. Certain mathematical topics will be talked about during the sections they are introduced (such as with lattices), but we first define and review the common algebraic structures that will be used often during this paper. For a more thorough treatment of the construction and properties of these algebraic structures, please consult [6].

A group $(G, \star)$ is a set of elements G equipped with a binary operation $\star : G \to G$ such that

(1) G is closed under $\star$
(2) $\star$ is associative; that is, $g_1 \star (g_2 \star g_3) = (g_1 \star g_2) \star g_3$ for all $g_1, g_2, g_3 \in G$
(3) G contains an identity, e, under $\star$; that is, $g \star e = e \star g = g$ for all $g \in G$
(4) For any $g \in G$, there exists an element $g^{-1} \in G$ such that $g \star g^{-1} = g^{-1} \star g = e$

A group in which $\star$ is also commutative, $g_1 \star g_2 = g_2 \star g_1$ for all $g_1, g_2 \in G$, is called an abelian group. To simplify notation, the $\star$ operation is often represented with juxtaposition of the two elements.

A vector space is an additive abelian group $(V, +)$ with a field F (referred to as the field of scalars) with the properties that

(1) $cv \in V$ for all $c \in F$ and $c \in V$
(2) $1v = v$ for all $v \in V$ and where 1 is the multiplicative identity of F
(3) $c(v_1 + v_2) = cv_1 + cv_2$ for all $c \in F$ and $v_1, v_2 \in V$
(4) $(c_1 + c_2)v = c_1v + c_2v$ for all $c_1, c_2 \in F$ and $v \in V$

A set of vectors $v_1, \ldots, v_n$ is said to be linearly independent if the only solution to the equation $c_1v_1 + \cdots + c_nv_n = 0$ is $c_1 = \cdots = c_n = 0$, the trivial solution. Otherwise, the set of vectors is linearly dependent. Additionally, it has also been proven that every vector space contains at least one set of linearly independent vectors such that every vector in the space can be uniquely written as a linear combination of that set of vectors. This set of vectors is called a basis for the vector space, and the cardinality of that set is referred to as the dimension of the vector space.

Rather than trying to find unique, original vector spaces, there are several ways to construct a vector space by combining existing vector spaces. The first, and more well-known, method is by constructing a set of 2-tuples with each component containing an element from a specified vector space. This is called the direct sum of two vector spaces V and W. Symbolically, this is written as $V \oplus W = \{(v, w) : v \in V,\ w \in W\}$. Addition within the direct sum is handled component wise: $(v_1, w_1) + (v_2, w_2) = (v_1 + v_2, w_1 + w_2)$ where the addition on the left is performed in $V \oplus W$ and the addition on the right are performed in their respective vector spaces. Scalar multiplication is handled in a similar manner: $c(v, w) = (cv, cw)$ where c is a scaler of both vector spaces. The dimension of a direct sum, given that $\dim(V) = m$ and $\dim(W) = n$ is $m+n$. One common example of a direct sum is $\mathbb{R}^2$ which is shorthand for $\mathbb{R} \oplus \mathbb{R}$.

Another construction is the tensor product. With a direct sum, the combining of the vector space leaves the vector spaces as independent entities since all operations are handled component-wise. On the other hand, the tensor product of V and W, written $V \otimes W$ more effectively merges the two into a distinct entity. Elements of $V \otimes W$ consist of linear combinations of elements of the form $v \otimes w$ where $v \in V$ and $w \in W$. As an operator, $\otimes$ is abstract. There is no general method to compute $v \otimes w$ into a form that we recognize. However, $\otimes$ is bilinear which gives the tensor product different properties than the direct sum. The properties are as follows:

(1) $(v_1 + v_2) \otimes w = (v_1 \otimes w) + (v_2 \otimes w)$ for all $v_1, v_2 \in V$ and $w \in W$.
(2) $v \otimes (w_1 + w_2) = (v_1 \otimes w) + (v_2 \otimes w)$ for all $v_1, v_2 \in V$ and $w \in W$.
(3) $c(v \otimes w) = (cv) \otimes w = v \otimes (cw)$ for all $v \in V, w \in W$, and $c \in F$.

The last property shows that scalar multiplication is more analogous to traditional multiplication, where $a(bc) = (ab)c = b(ac)$ for real numbers, than the direct sum. Both the direct sum and tensor products can be performed on more than two vector spaces as long as the elements of each space are kept in the correct components.

2.2 Quantum Computing

In order to discuss the effects of quantum computers on modern cryptography, we need to have a basic understanding of quantum computing. With that in mind, we give a very brief overview of the basic terminology and ideas. For a more thorough introduction into the subject, see [9, 19].

Quantum computing is a study which takes advantage of quantum phenomena to perform computations. As classical computing uses the logical bit to encode information, quantum computing uses the quantum bit or qubit to encode information. A bit can take the form of a 1 or 0. Physically, this can be implemented through various methods such as a flip of a switch or the level of a supplied voltage.

A qubit is a bit more complex. Physical implementations of a qubit are performed by measuring the spin of an elementary particle, by measuring the energy of a photon, or through some other quantum measurement. Hence, many quantum properties are also inherently found within the qubit. A qubit can be represented as the superposition of two quantum states: $|0\rangle$ and $|1\rangle$. Symbolically, this can be written as $\alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. When measured, the states $|0\rangle$ and $|1\rangle$ represent the bits 0 and 1 respectively. The square of the modulus, or absolute value, of the coefficient associated with a state represents the probability of a measured qubit becoming the associated binary value.

Due to the nature of quantum mechanics, when a qubit is in a superposition of the two states, we do not know which state the qubit will produce when it is measured. As light acts as both a particle and a wave, a qubit, in a sense, can exhibit properties of both a 0 or a 1 since it has a probabilistic chance of producing either when measured. However, once it is measured, it we do know that the qubit will produce the same value every subsequent time it is measured. This act of nature is often compared to the way a polarizing lens acts upon light and forces it to travel at a certain angle.

To actually encode information, classical computers make use of strings of bits. Each bit works independently, but a sequence of bits can give meaningful information. To make a sequence of qubits, we have to take the tensor product of quantum states. If we abuse notation slightly and suppose that two quibits are expressed with the standard basis $|0\rangle,\ |1\rangle$, then

$$(a_0|0\rangle + a_1|1\rangle) \otimes (b_0|0\rangle + b_1|1\rangle) =$$
$$a_0b_0|0\rangle|0\rangle + a_0b_1|0\rangle|1\rangle + a_1b_0|1\rangle|0\rangle + a_1b_1|1\rangle|1\rangle$$

by the bilinear properties of the tensor product. More compactly, we write this as $a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$ where the a_{ij} are the normalized forms of the coefficients of the tensor product such that $\sum_i \sum_j |a_{ij}|^2 = 1$.

However, a system with multiple qubits is not always arranged this nicely. For example, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ cannot be separated into the tensor product of single qubits as in the previous example. A system of qubits whose representation cannot be separated into the tensor product of single qubits is said to be entangled. Physically speaking, this means that the quantum phenomena behind the qubits are inherently related. Though we only mention the use of entanglement at a high level, it is of extreme importance and usefulness in quantum information theory.

As with classical computing, quantum computing has its own set of logic gates, often also referred to as quantum transformations. In particular, the Hadamard transformation is a popular quantum transformation which can written as

$$H = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |1\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 1|)$$

Alternatively, H maps $|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. An extension of this transformation, called the Walsh-Hadamard transformation, uses the tensor product of n Hadamard transformations to produce a transformation for an n qubit system. What this allows us to do is apply the Walsh-Hadamard transformation

on n individual qubits, all in the state $|0\rangle$ to obtain the following:

$$\begin{aligned}
&(H \otimes H \otimes \cdots \otimes H)|00\ldots0\rangle \\
&= \frac{1}{\sqrt{2^n}}((|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle)) \\
&= \frac{1}{\sqrt{2^n}}(|0\ldots00\rangle + |0\ldots01\rangle + |0\ldots10\rangle + \cdots + |1\ldots11\rangle) \\
&= \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle
\end{aligned}$$

What this transformation allows us to do is take an easy to construct n-qubit system and transform it into a superposition of all non-negative numbers less than 2^n. From a mathematical perspective, if we were to put this superposition into a linear function or transformation, the function would act on each of the $2^n - 1$ numbers (represented as states). In a sense, we would be using the same function on $2^n - 1$ different numbers without having to use the function more than once. In quantum computing, this concept is referred to as quantum parallelism.

Now, it's important to realize that this concept is not as powerful as one might initially think. While, on paper, a function could act on $2^n - 1$ different numbers at the same time, when the system is measured, it will still only produce one of those values. The remaining values are still unknown to us, and a new system would need to be created with the Walsh-Hadamard transform in order to try again and get a different value. With that said, careful planning and manipulation of the coefficients of this superposition can result in powerful quantum algorithms that can accomplish a lot more than classical computers can.

Though the theory is deeper with a larger influence on the subject, one method of influencing the coefficients (also called the amplitudes) of a quantum state is with the quantum Fourier transform. In general, the amplitudes a_x of a n-dimensional quantum state $\sum_x a_x|x\rangle$ can be viewed as a function of x, denoted $a(x)$. To draw parallels between the quantum and discrete Fourier transforms, note that the domain of $a(x)$ is $\{0, 1, \ldots, 2^n-1\}$ when dealing with qubits. The quantum Fourier transform U_F acts linearly on a quantum state as

$$U_F\left(\sum_x a(x)|x\rangle\right) = \sum_x A(x)|x\rangle$$

where $A(x) = \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} a(k)\exp\left(2\pi i\frac{kx}{2^n}\right)$.

As with other Fourier transforms, the big interest is when the amplitudes $a(x)$ are periodic with a period r. If the period is a power of 2, it turns out that $A(x)$ is 0 except from when x is a multiple of $\frac{N}{r}$. When this is the case, we can identify the types of base states, those that are multiples of $\frac{N}{r}$, we will get from measuring the transformed quantum system. If the period is not a multiple of 2, or more explicitly does not divide 2^n, then the behavior described above is approximated. It is still possible to produce a base state that is not a multiple of $\frac{N}{r}$, but it will have a very low probability of being measured while base states that are close to multiples of $\frac{N}{r}$ will have a very large probability of being measured. While this may seem like a very specific property to take advantage of, it has become the crux of several quantum algorithms include Shor's algorithm.

3 EFFECT OF QUANTUM COMPUTERS ON CRYPTOGRAPHY

3.1 Popular Protocols

From a security perspective, quantum computing can be a terrifying subject. With the extra computational power of the qubit, algorithms can be designed to accomplish tasks that were not possible with the traditional bit. To demonstrate this, Shor's algorithm is discussed later. The main takeaway, however, is that Shor's algorithm can find the prime factorization of large numbers in polynomial time whereas this task cannot be done efficiently with classical computation.

Mathematical cryptography is based off of the idea of hard problems and trapdoors. Essentially, cryptography looks for problems that are easy to compute, but hard to reverse-engineer unless you have access to the secret back door which is normally done with a private key. Popular examples of these hard problems are prime factorization, the discrete logarithm, the quadratic residuocity problem, and the shortest vector problem. Of these, the first is used by RSA and the second is used by the Diffie-Hellman and ElGamal protocols. They are very popular algorithms and most people with even a high-level knowledge of security have heard of at least one of them.

Since RSA is secure if and only if the prime factorization of large numbers is hard, quantum computers will be able to break the most widely-used cryptosystems in use today. By the same computational abilities, quantum computers have also been proven to be able to break the discrete logarithm which suddenly makes any cryptosystem that uses elliptic curves also insecure.

Luckily, cryptographers and cryptanalysts are vehemently against having a single cryptographic primitive to accomplish a task. RSA, Diffie-Hellman, and ElGamal are some of the most popular because they are arguably simple, have keys of manageable size, and have a lot of research to support their security within classical computers. However, that does not mean that research has not been done to support the usage of other options for cryptosystems. There are several categories of research currently being conducted including lattices, multivariate cryptography, code-based cryptography, hashes, and many others. However, the categories explicitly mentioned are widely believed to be the best candidates to withstand attacks from quantum computers. Many of these categories had working cryptosystems well before the rush began to replace the modern broken cryptosystems, but they were not as popular due to reasons stated above. Before we move into examining the properties of these categories, we first try to answer the question of how to tell what cryptosystems are easily broken by quantum computers and which are not.

3.2 Shor's Algorithm

While better performing algorithms have been invented, Shor's algorithm is the first and possibly the most famous quantum solution to the prime factorization problems. Other more classical algorithms exist, but they are computationally infeasible for large numbers which is why prime factorization was considered a hard problem for cryptographers.

However, it is worth noting that Shor's algorithm does not directly factor numbers. Instead, his algorithm solves a different but related problem that, once solved, can help factor numbers. Shor's algorithm uses quantum capabilities to find the period of a number. Since it nicely makes use of quantum parallelism and the QFT to accomplish this goal, we will briefly go over the algorithm on a high level to help showcase how quantum computation can help solve problems and how ideas from Shor's algorithm can be used in more generalized cases.

First, we need to discuss what a period of a number is and how it can be used to factor numbers. For the sake of setting up our problem, let M be the number we wish to factor, and let $0 < a \leq M - 1$. The period of a number a modulo M is the smallest positive integer r that satisfies the equation $a^r \equiv 1 \pmod{M}$. In more algebraic terms, the period of a is the order of a within the multiplicative group $\mathbb{Z}_M^\times$. If such a solution does not exist, then r is set to equal infinity.

However, we do know a condition where r will be a finite value. It turns out that r is finite if and only if a is relatively prime to M - that is, the greatest common divisor or factor between them is 1. In this case where r is finite, then we see that $a^k \equiv a^k a^r \equiv a^{k+r} \pmod{M}$. Hence, if we define a function $f_a(k) = a^k \pmod{M}$, then we can say, in more familiar terms, that r is the period of our function f since $f(k + r) = f(k)$ for all values of k in the domain of f. Furthermore, if r is an even number, then we can write

$$a^r \equiv 1 \pmod{M} \Rightarrow (a^{r/2} + 1)(a^{r/2} - 1) \equiv 0 \pmod{M}.$$

The only reason for r being even is so that $\frac{r}{2}$ is still an integer. Assuming that neither term on the left is a multiple of M, and thus is not 0 modulo M, then it is very likely that $a^{r/2} + 1$ contains a non-trivial factor of M since the product of it and $a^{r/2} - 1$ is a multiple of M. From here, the famous Euclidean algorithm can be used to find the greatest common factor between $a^{r/2} + 1$ and M.

What we describe above is a general description to factor a number M when we can find the period of a positive number less than $|M|$. This can all be done with classical computational abilities. The brilliance of Shor's algorithm is in using quantum computing to find that period. After selecting a random integer a that is relatively prime to M as described above, we can use parallelism by performing $f(x) = a^x \pmod{M}$ on the superposition $\sum_{x=0}^{2^n-1} |x\rangle|0\rangle$. The values of the summation are based on an integer n such that $M^2 \leq 2^n \leq 2M^2$. This comes from analysis to make the algorithm more efficient. In the end, the summation is transformed to $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle$.

Measuring $|f(x)\rangle$, which is stored in a separate register from $|x\rangle$, produces a state $|u\rangle$ and we now get $C \sum_{x=0}^{2^n-1} g(x)|x\rangle|u\rangle$ where C is the appropriate scaling value and $g(x)$ is 1 if $f(x) = u$ and 0 otherwise. At this point, the value of u is irrelevant to the problem at hand and can be ignored. Since $f(x) = a^x \pmod{M}$, we know that $f(x) = f(y)$ if and only if x and y differ by a multiple of the period. This can be extended to show that f and g have the same period.

The trick to Shor's algorithm is to now manipulate the new superposition in such a way that, when measured, we have a high probability of getting one of the base states such that $g(x) = 1$. This is accomplished with the quantum Fourier transform. By applying the transform to the first register, we get

$$U_F \left(C \sum_{x=0}^{2^n-1} g(x)|x\rangle \right) = C' \sum_c G(c)|c\rangle$$

where $G(c) = \sum_x g(x) \exp\left(\frac{2\pi icx}{2^n}\right)$. As was described with the introduction of the quantum Fourier transform, $G(c)$ is large when c is either a multiple of $\frac{2^n}{r}$ or close to a multiple where r is the period of both g and f. Thus, by measuring the qubit system after the quantum Fourier transform, there is a high probability of getting a number that is at least close to having a formula of producing r. From here, the rest of Shor's algorithm uses classical means of obtaining r from the measured value. Since our focus is on the quantum implications and abilities, we will not discuss Shor's algorithm any further. However, a description of the full algorithm can be found in [9].

3.3 Hidden Subgroup Problem

As Shor's algorithm shows, problems that were previously thought to be hard for classical computers are not necessarily hard for quantum computer. With the power of parallelism, quantum algorithms can be designed to take computational shortcuts to achieve more in a shorter amount of time. When we look for new problems and methods to base our cryptosystems on, it helps to have an idea of what kind of problems can easily be solved by quantum computers and which ones cannot. Luckily, many problems that are being examined for cryptosystems can be generalized to have similarities to each other, and these generalizations lead to the hidden subgroup problem, or HSP. By knowing what form the examined problem can be generalized to, the hidden subgroup problem can help indicate if the cryptosystem would be insecure or if it requires more research to determine its level of security.

Suppose we are given a group G and implicitly define a subgroup $H < G$ by defining a function f on G. The codomain of f is determined by the specifics of the application. This function f is constant and distinct on every coset gH of H for every $g \in G$. The Hidden Subgroup Problem asks us to find the set of generators for this subgroup H.

The general form of this problem, as described above, is currently unsolved. However, there does exist a polylogarithmic bounded probability quantum algorithm for solving the problem in the case where G is abelian. This is partly due to the mathematical theorem stating that abelian groups have cyclic decompositions. Since, in computational applications, G is often a finite abelian group, this cyclic decomposition is simpler to find.

Because of the generality of the HSP, it is very easy to express cryptosystems within the confines of this problem. What this means for cryptanalysts is that if a cryptosystem can be expressed as a finite abelian group, there exists a quantum algorithm to solve the HSP for this cryptosystem which can result in a method of breaking the encryption. While we will not got over the algorithm, we will briefly demonstrate how converting both period finding and the discrete logarithm problems to subgroups and solving the HSP can lead to solving these problems.

Recall that cryptosystems based on the prime factorization of a number N can be reduced to finding the period of a factor of

N. With this in mind, let f be a periodic function on $G = \mathbb{Z}_N$ with a period r such that r divides N. In accordance with the HSP, we define H to be the subgroup of $\mathbb{Z}_N$ generated by r. Since $\mathbb{Z}_N$ is finite and abelian, H has a generator h, and r can be found by the relation $r = \gcd(h, N)$. Thus, we have a method of finding the period of N which allows us to break prime factorization problems. This reinforces the fact that RSA will be breakable by quantum computers.

Similarly, we can show that the HSP can help break the discrete logarithm problem. Recall that the discrete logarithm problem says that, for the multiplicative group $G = \mathbb{Z}_p^*$ where p is prime, given an element $b \in G$ and an positive integer y, find the solution x to the equation $b^x \equiv y \pmod{p}$. Consider a function $f : G \times G \to G$ where $f(g, h) = b^{-g} y^h$. We can define a subgroup $H < G \times G$ as the set of elements satisfying $f(g, h) = 1$. It turns out that the elements of H take the form (mx, m). Furthermore, from any generator of H, we can compute $(x, 1)$. As a result, solving the HSP for the discrete logarithm problem can produce the solution x to the equation we outlined above. This helps prove how ElGamal, Diffie-Hellman, and eliptic curve cryptography can all be broken by quantum computers.

4 KEY EXCHANGE PROTOCOLS

After considering the abilities of quantum information processing, cryptography has been split into two related but separate studies: quantum and post-quantum cryptography. Post-quantum cryptography is the study and development of cryptosystems, for both classical and quantum computers, that will resist quantum computing such as the NTRU cryptosystem. We will take a further examination of post-quantum cryptography in section 5.

Quantum cryptography, however, is the study of exploiting quantum mechanics to achieve secure communication between quantum computers. A large part of this field is studying how to exchange keys between quantum computers - for without keys, we would not be able to encrypt or decrypt any information. Due to the nature of quantum computers, most of these developed key distributions allow for the sharing of a single key which lets quantum computers use symmetric cryptographic algorithms.

The first published Quantum key distribution, BB84, was designed by Charles Bennett and Gilles Brassard in 1984 [11]. In this scheme, we have two parties, Alice and Bob, who wish to share a secret key. As with many other key distributions, Alice wishes to secretly send Bob a string of bits that will be used as a key. However, since we are assuming that Alice and Bob both have access to quantum computers, we suppose that they have two viable public channels between them: a classical channel which can send bits and a quantum channel which can send qubits.

First Alice randomly generates a string of random bits and encodes each bit into a separate qubit by polarizing photons. For each bit, this is done by randomly choosing either the standard basis, in which we have the following bit to qubit mapping:

$$0 \mapsto |\uparrow\rangle = |0\rangle$$
$$1 \mapsto |\rightarrow\rangle = |1\rangle$$

or the Hadamard basis, giving the following mapping:

$$0 \mapsto |\nearrow\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle + |\rightarrow\rangle)$$
$$1 \mapsto |\nwarrow\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle - |\rightarrow\rangle)$$

The arrows in the above notation represent the quantum state of a qubit after polarizing its photon in the respective direction. Alice then send these qubits, one at a time, to Bob. After using the classical channel to verify that Bob has received all of the qubits, Bob randomly measures each qubit in the order they were sent with either the standard basis or the Hadamard basis.

By the nature of quantum measurement, if Alice and Bob's choice of basis agree, then Bob will produce the same bit value. However, if the choice of basis differs for a given qubit, Bob has a 50 percent chance of getting the same bit value. Since sending the bits over the classical channel to verify which ones are the same would be insecure, Alice and Bob instead use the classical channel to verify for which qubits the basis were the same. Bob and Alice discard any bits for which a differing basis was chosen. This way, they can guarantee that they have the same string of bits without sending any of them over a classical channel for eavesdropping. Before they are finished, Bob and Alice also agree on a small set of the remaining bits to send over the classical channel. If these bits match, then they are removed from the string, and the remaining string becomes the shared secret key. If any of these bits do not match, the two parties are alerted that there may be an eavesdropper manipulating the qubits, and the process is started again.

The quantum nature of the photons being sent between Alice and Bob offer some natural security to this process. In a traditional interception setup, suppose there is a third party, Oscar, who is eavesdropping on the quantum channel when Alice is sending out her photons. When Oscar receives Alice's photons in-transit, he has two options, to just send the qubits along to Bob or to measure the qubit before doing so. Doing the former does not benefit Oscar at all, but the second option does not help him either. This is because Oscar would have to randomly choose which basis to measure the photon in since he has no knowledge on which basis Alice chose for her encoding. Measuring each photon means that each qubit suddenly becomes fixed to one of the basis elements chosen for the measurement. And since Oscar is randomly choosing basis for measuring, an average of 50 percent of them will be accurate to Alice's choice. When Bob receives them, he too will randomly choose bases resulting in about 25 percent of them being correct. With an accuracy so low, Alice and Bob will be made aware of a third presence interfering with their communication.

Some readers may wonder about a third option for Oscar. Would it be possible to recreate or copy the qubits before sending them to Oscar? This way, he has access to the unmeasured quantum states when Bob and Alice start verifying basis choices over the classical channel. Oscar could then follow what bases were being used by Bob to measure the qubits. Unfortunately, there is no way to generally clone an unknown quantum state (a principle known as the no-cloning theorem).

With that said, BB84 is not immune to all attacks. For example, it lacks authentication mechanisms, though this can be fixed with the addition of hash-based signature schemes. There is also the case of a

photon number splitting attack. Due to the physical implementation of sending photons, Alice may sometimes send two photons at the time with the same characteristics to Bob. Oscar could keep the second photon, send the first photon to Bob, and measure the photon with the correct basis when it is revealed between Alice and Bob. Oscar would not retrieve the entire key through this method, but he could use this information to guess the remainder of the key.

As one would guess since BB84 was proposed in 1984, research has been done into several other protocols and options to improve key distribution options. The E91 protocol [8] is based off of BB84 but measures quantum entangled photons to retrieve the shared key. Thought it has other potential attacks, the SARG04 protocol [24] improves the BB84 protocol so that, under certain parameters, the protocol can resist the photon number splitting attack. There is also the decoy state protocol which allows Bob and Alice to detect if the photon number splitting attack is occurring [17].

5 LATTICE-BASED CRYPTOGRAPHY

5.1 Background Information

Let $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$ be a set of linearly independent vectors of $\mathbb{R}^m$. A lattice $\mathcal{L}$ generated by $\mathcal{B}$ is the set $\{\sum_{i=1}^{n} c_i b_i : \ c_i \in \mathbb{Z}\}$. In other words, given a linearly independent subset $\mathcal{B}$ of $\mathbb{R}^n$, a lattice is the set of integral linear combinations of $\mathcal{B}$.

There are several variations of cryptosystems that rely on lattices. To break any of these cryptosystems is equivalent to solving either the closest vector problem or CVP, which is $\mathcal{NP}$-hard, or the shortest vector problem or SVP, which is $\mathcal{NP}$-hard under certain assumptions. Given a lattice $\mathcal{L}$ and a vector $w \in \mathcal{L}$, the CVP problem aims to find a vector $v \in \mathcal{L}$ such that $\|w - v\|$ is minimal. Similarly, the SVP strives to find a vector $v \in \mathcal{L}$ such that $\|v\|$ is minimal. While both problems allow for $\|\cdot\|$ to be any L_p norm, the Euclidean norm is most commonly used in applications. Furthermore, it turns out that CVP can be reduced to finding a solution to SVP in a lattice of a higher dimension.

Suffice to say, both problems are very hard to solve. While some algorithms do exist, like with the factorization problem, they become computationally expensive if the dimension of the lattice is large enough. This makes them good candidates to base cryptosystems on. Moreover, there is no currently known quantum solution to either the SVP or the CVP.

While there have been several lattice-based cryptosystems proposed over the years, two of the more prominent ones have been the GGH cryptosystem, proposed by Oded Goldreich, Shafi Goldwasser, and Shai Halevi in 1997 [10], and the NTRU cryptosystem, whose first encryption algorithm was proposed by Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman [12]. Since its inception, GGH has been shown to have a few weaknesses based on the choice of its parameters. If they are poorly chosen, the system becomes vulnerable to attacks based on lattice basis reduction algorithms such as the famous Lenstra-Lenstra-Lovász, or LLL, lattice basis reduction algorithm and to attacks based on Babai's algorithm. However, there have been efforts to improve GGH. In comparison, NTRU has had much fewer problems and even more analysis. The cryptosystem is already being implemented in open source software such as open quantum safe and wolfSSL, and as a result, will be the primary focus of this section.

5.2 NTRUEncrypt

It should be noted that, even though the NTRU algorithms are considered to be lattice-based, they do not actually use lattices in their computation like the GGH cryptosystem does. Instead, as will be explained later, the structure of the keys can be viewed within a lattice and, if the SVP can be broken, NTRU can be broken as well. Instead of a lattice, NTRU directly relies on factoring polynomials in convolution polynomial rings to encrypt and decrypt messages.

A ring R is a set of elements equipped with two binary operations, usually denoted as addition ($+$) and multiplication ($*$), with the following properties:

(1) R is an abelian group under addition
(2) R is closed under $*$
(3) $*$ is associative: $(a * b) * c = a * (b * c)$ for all $a, b, c \in R$
(4) The distributive properties hold in R: $a * (b + c) = a * b + a * c$ and $(b + c) * a = b * a + c * a$ for all $a, b, c \in R$

Higher classes of rings can be made by adding more properties such as requiring that $*$ be commutative (called a commutative ring), R contain multiplicative identities (called a ring with identity), R have no zero divisors (called an integral domain), or that R be an integral domain with a multiplicative inverse for every nonzero element (called a field).

NTRUencrypt uses convolution polynomial rings for its calculations. A convolution polynomial ring is a quotient ring of the form $\mathbb{Z}[x]/(x^n - 1)$ or $\mathbb{Z}_q[x]/(x^n - 1)$ where q, n are positive integers and $\mathbb{Z}[x], \mathbb{Z}_q[x]$ are rings of polynomials with coefficients of integers and integers modulo q respectively. For readers unfamiliar with ring theory, see [6] for more details on quotient rings. For the sake of brevity, we summarize the key points needed for our discussion. The elements of a convolution ring take the form $a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$ where a_i are either in $\mathbb{Z}$ or $\mathbb{Z}_q$ depending on the ring. Furthermore, in both rings, we have the relation $x^n - 1 = 0 \Rightarrow x^n = 1$. If we encounter a monomial of the form x^m where $m > n$ during our calculations, this can be reduced modulo $x^n - 1$ as $x^m = x^{n+(m-n)} = x^n x^{m-n} = x^{m-n}$.

Addition within these rings is the normal polynomial addition, that is, it happens component-wise followed by a modular reduction if needed. Multiplication in these rings is also the traditional polynomial multiplication. Since polynomials can be stored by keeping track of their coefficients, we can express the multiplication by writing multiplication of their coordinate vectors as

$$a \star b = c \text{ where } c_k = \sum_{k=i+j} a_i b_j$$

For the NTRU cryptosystem, we will define the three following polynomial rings:

$$R = \frac{\mathbb{Z}[x]}{(x^N - 1)}, \quad R_p = \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{(x^N - 1)}, \quad R_q = \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{(x^N - 1)}$$

where N is a predetermined integer and p and q are two integers chosen for modular arithmetic.

There is one more topic of discussion before breaking down NTRUEncrypt, and that is the homomorphic relation between R and R_p or R_q. In ring theory, a homomorphism Φ is a mapping from ring R to Q such that $\Phi(r_1 + r_2) = \Phi(r_1) + \Phi(r_2)$ and $\Phi(r_1 r_2) = \Phi(r_1)\Phi(r_2)$ for all $r_1, r_2 \in R$. In other words, a homomorphism preserves the

relations defined by the arithmetical operations (addition and multiplication) of the two rings.

It turns out that the mapping created by reducing an element of R by modulo q is a homomorphism. So we have a natural homomorphism from R to R_p and from R to R_q which gives us very nice algebraic relations. However, we will find that it is also useful to go in the other direction (from R_q or R_p to R), but this is not a homomorphism. In fact, to create such a mapping, extra restrictions would need to be in place for it to even be well-defined.

With this in mind, let $a(x) \in R_q$. The center-lift of $a(x)$ from R_q to R is the unique polynomial $A(x) \in R$ such that

$$A(x) \equiv a(x) \pmod{q}$$

and coefficients of $A(x)$ satisfy the relation

$$-\frac{q}{2} < A_i \leq \frac{q}{2}$$

While the center-lift is not generally a homomorphism, it does allow us to use a well-defined reverse mapping for our induced homomorphisms.

The NTRUEncrypt process begins by fixing some public parameters. It first chooses a prime integer N and a positive integer d. Then two positive integers p and q are selected satisfying $\gcd(N, q) = \gcd(p, q) = 1$ and $q > (6d+1)p$. Now suppose Alice and Bob wish to communicate. Alice first chooses a ternary polynomial.

A ternary polynomial is an element of the set

$$\mathcal{T}(d_1, d_2) = \left\{ a(x) \in R : \begin{array}{r} a(x) \text{ has } d_1 \text{ coefficients equal to } 1 \\ a(x) \text{ has } d_2 \text{ coefficients equal to } -1 \\ a(x) \text{ has all other coefficients equal to } 0 \end{array} \right\}$$

for two positive integers d_1, d_2. Alice chooses ternary polynomials $f(x) \in \mathcal{T}(d+1, d)$ and $g(x) \in \mathcal{T}(d, d)$. Alice then computes the inverses of f in R_q and R_p, denoted as $F_q(x) = f(x)^{-1} \in R_q$ and $F_p(x) = f(x)^{-1} \in R_p$. If either of the inverses fails to exists in either ring, $f(x)$ is discarded and a new random ternary polynomial is selected.

Next, Alice computes the polynomial $h(x) = F_q(x) \star g(x)$ in R_q. At this point, all the set up for the encryption scheme is complete. Alice's public key will be $h(x)$, and the four parameters (N, p, q, d) will also be publicly known. Alice's private key will be the tuple $\big(f(x), F_p(x)\big)$. If storage is an issue, Alice's private key may consist of only $f(x)$ allowing $F_p(x)$ to be recomputed as needed.

With the keys decided, Bob can now send Alice a message. Bob's plaintext must be converted to a polynomial with coefficients satisfying the relation $-\frac{1}{2}p < m_i \leq \frac{1}{2}p$. Using our terminology from earlier, this mean that $m(x)$ is the center-lift for some polynomial in R_p. After choosing a random polynomial $r(x) \in \mathcal{T}(d, d)$, Bob computes his ciphertext $e(x) \equiv ph(x) \star r(x) + m(x) \pmod{q}$ in R_q.

Alice receives Bob's ciphertext $e(x)$ and decrypts it by first computing $a(x) \equiv f(x) \star e(x) \pmod{q}$. She then center-lifts $a(x)$ to a polynomial $A(x)$ in R and computes $b(x) \equiv F_p(x) \star A(x) \pmod{p}$. Because of the restrictions set by the public parameters, $b(x) = m(x)$. We can verify this by first observing that

$$\begin{aligned} a(x) &\equiv f(x) \star e(x) \pmod{q} \\ &\equiv f(x) \star ph(x) \star r(x) + f(x) \star m(x) \pmod{q} \\ &\equiv pf(x) \star F_q(x) \star g(x) \star r(x) + f(x) \star m(x) \pmod{q} \\ &\equiv pg(x) \star r(x) + f(x) \star m(x) \pmod{q} \end{aligned}$$

Because $g(x), r(x) \in \mathcal{T}(d, d)$, they will have d 1's and d -1's as coefficients. Hence, the largest possible coefficient of their product will be $2d$. By similar reasoning since $f(x) \in \mathcal{T}(d+1, d)$ and $m(x)$ is a center-lift, the largest coefficient of their product can be $(2d+1)\frac{1}{2}p$. This results in $a(x)$ having a largest possible coefficient of $p2d + (2d+1)\frac{1}{2}p = (3d + \frac{1}{2})p < (6d+1)p < q$. Since every coefficient of $a(x)$ is bounded by q, no coefficient is reduced and we can say that $a(x) = pg(x) \star r(x) + f(x) \star m(x)$ rather than just being equivalent modulo q.

As a result, we also have the relation $A(x) = a(x)$. By substitution, this means that

$$\begin{aligned} b(x) &\equiv F_p(x) \star a(x) \pmod{p} \\ &\equiv F_p(x) \star (pg(x) \star r(x) + f(x) \star m(x)) \pmod{p} \\ &\equiv F_p(x) \star f(x) \star m(x) \pmod{p} \\ &\equiv m(x) \pmod{p} \end{aligned}$$

From here, it is easy to convert retrieve $m(x)$ from $m(x) \pmod{p}$ since the coefficients of $m(x)$ were already restricted to satisfy $-\frac{1}{2}p < m_i \leq \frac{1}{2}p$.

5.3 Breaking NTRUEncrypt

When describing NTRUEncrypt, we have not mentioned anything about a lattice at all. While the computation involves polynomial ring, the act of breaking the encryption can be written in a way to involve lattices. It also turns out that, when written in terms of lattices, trying to break the encryption is equivalent to solving the SVP.

Recall that the public key of NTRUEncrypt is a polynomial $h(x)$ in R_q of degree $N-1$ with coefficients h_i. The NTRU lattice, denoted as $L_{\mathbf{h}}^{NTRU}$ associated with $h(x)$ is the lattice spanned by the rows of the $2N$-dimensional block matrix

$$M_{\mathbf{h}}^{NTRU} = \begin{pmatrix} I_N & \mathbf{h} \\ 0 & qI_N \end{pmatrix}, \text{ where } \mathbf{h} = \begin{pmatrix} h_0 & h_1 & \dots & h_{N-1} \\ h_{N-1} & h_0 & \dots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \dots & h_0 \end{pmatrix}$$

Of the remaining blocks, I_N is the identity matrix of order N, qI_N is the matrix of order N with q's along the diagonal, and 0 is the matrix of order N composed entirely of zeros. Additionally, we can represent pairs of polynomials $a(x), b(x) \in R$ as a $2N$ dimensional vector by keeping track of their coefficients in the form $(a, b) = (a_0, a_1, \dots, a_{N-1}, b_0, \dots, b_{N-1})$.

With this notation in mind, we can identify a relationship between the public key $h(x)$ and the matrix $M_{\mathbf{h}}^{NTRU}$. Recall that $h(x) \equiv F_q \star g(x) \pmod{q}$. Then $f(x) \star h(x) \equiv g(x) \pmod{q}$. Then by the definition of congruence modulo q, there exists a polynomial $u(x)$ such that $f(x) \star h(x) = g(x) + qu(x)$. Then we have the equality

$$(f, -u)M_{\mathbf{h}}^{NTRU} = (f, -u)\begin{pmatrix} I_N & \mathbf{h} \\ 0 & qI_N \end{pmatrix} = (f, f \star h - qu) = (f, g).$$

This is important because $f(x)$ is the private key of the NTRUEncrypt algorithm. $g(x)$, while also private, is thus shown to have a relationship with Alice's private key and her public key, represented by the matrix in the above equation. Breaking NTRUEncrypt is equivalent to finding polynomials $f(x)$ and $g(x)$ in R_q that satisfy our equation. Furthermore, we can represent this problem as a system of equations allowing cryptanalysts to use tools from linear algebra and lattice theory, such as the LLL basis reduction algorithm, to attack the problem.

6 MULTIVARIATE CRYPTOGRAPHY

Cryptosytems that fall under the multivariate category make use of sets of nonlinear, usually quadratic, polynomials. Generally speaking, these cryptosystems make use of a set $\mathcal{P} = \{p_1, \ldots, p_m\}$ of polynomials where each polynomial takes the form

$$p_k = \sum_i a_{ik} x_i^2 + \sum_{i>j} a_{ijk} x_i x_j + \sum_i a_{ik} x_i$$

where $X = \{x_1, \ldots, x_n\}$ are the indeterminates of the polynomials and the coefficients are elements of some finite field $\mathbb{F}_q$ for some prime number q.

The security of these cryptosystems is based off of the assumed hardness of the $\mathcal{MQ}$ problem which is considered to be $\mathcal{NP}$-hard. The $\mathcal{MQ}$ problem asks us to find a solution to the system of equations $p_1(X) = \cdots = p_m(X) = 0$ where each polynomial $p_i(X)$ is quadratic, as described above, in $X = \{x_1, \ldots, x_n\}$. The problem also assumes that the coefficients of each polynomial are elements of a fixed finite field. Finding solutions to these types of problems is the very foundation of the study of algebraic geometry which is where most of the machinery for these cryptosystems comes from.

While the different algorithms can differ in their details, many multivariate systems follow similar approaches to encryption and decryption. The most famous of the categories is the Bipolar construction. Using the notation that was previously established, the public key is the set of polynomials $\mathcal{P}$. For storage purposes, this is usually represented by somehow only keeping track of the coefficients in some predetermined order and by assuming that the constant coefficient is 0.

The private key takes a bit more work to describe. Since the underlying problem asks us to find solutions in $\mathbb{F}_q^n$, each of the polynomials in $\mathcal{P}$ can be thought of as a map from $\mathbb{F}_q^n$ to $\mathbb{F}_q^m$. To find the private key, $\mathcal{P}$ is constructed in such a way that it can be decomposed as such: $\mathcal{P} = \mathcal{S} \circ \mathcal{M} \circ \mathcal{T}$ where $\mathcal{S}$ and $\mathcal{T}$ are affine maps and $\mathcal{M}$ contains invertible maps of polynomials. The set $\mathcal{M}$ along with certain parameters of $\mathcal{S}$ and $\mathcal{T}$ serve as the private key.

This showcases one of the larger problems of implementing a multivariate cryptosystem: the key size. The modern RSA implementation uses 2048-bit keys, and it is a common belief that the algorithm is currently slow, even though that term is relative, because of the large key size. For example, a modern implementation of the Rainbow signature scheme in $GF(2^8)$ with 24 equations and 42 indeterminates requires a public key or size 22,680 bytes [15]. [3] also contains more examples of large key sizes for various multivariate cryptosystems. While this does not make it impossible for some embedded systems and other smaller devices to use, it is certainly an obstacle that needs to be considered in this field.

Other categories include the Mixed or Implicit category and the Isomorphism of Polynomials category. Information on the general constructions of both types of multivariate cryptosystems can be found in [5] and [15]. The first multivariate cryptosystem is often attributed to H. Ong, C. P. Schnorr, and A. Shamir in 1984 [21]. However, it was broken by Pollard and Schnorr only a couple of years later [23]. A few more attempts would be made and quickly broken in a similar fashion until T. Matsumoto and H. Imai published the C^* scheme in 1988 [18]. Although it too would become broken, it popularized the study of multivariate functions for cryptographic purposes and inspired the Hidden Fields Equations which would become the foundation for the Quartz, Flash, and SFlash signature schemes. One major attack against the C^* cryptosystem was a linearization attack. By studying the effect of this attack, Patarin developed the balanced Oil-Vinegar signature scheme in 1997 [22]. After being proven insecure, the Oil-Vinegar schemes soon came to include two other families of signature schemes: the unbalanced oil-vinegar scheme (originally made by Kipnis et al in 1999 [16]), and the Rainbow scheme (made by Ding and Schmidt in 2005 [7]). To date, both signature schemes are still believed to be secure against both classical and quantum computers.

Multivariate cryptographic methods have a come a long way since their inception and have gone through many iterations. Unlike with lattice-based cryptography which has had one cryptosytem dominate the field, many different multivariate schemes are currently being studied - even if they can all have their origins traced back to the same system. Though each scheme may have its own attacks, there are three types of attacks the multivariate cryptosystems are generally weak against due to their reliance of sets of polynomials over finite fields: the Gröbner basis method, the XL method, and the Zhuang Zi method. To discuss these attacks would require learning tools from ring theory and algebraic geometry, so they will not be discussed any further. However, information on the attacks can be found in [15].

7 CODE-BASED CRYPTOGRAPHY

Code-based cryptosytems are designed to use error-correcting codes as their one-way function. The first, and arguably most famous, cryptosystem in this category was conceived by Robert J. McEliece in 1978 [13]. Another well-known code-based cryptosystem is the Niederreiter cryptosystem [20], and the studies in this category also have implications in random number generation, hash functions, and signature schemes. Even better, in the 40 years since the invention of the McEliece cryptosystem, there have been no discovered quantum attacks against this category of systems - though some attacks and concerns still exist for individual cryptosystems. However, these algorithms are known to have significantly large keys which makes implementation difficult due to a lack of available memory.

The McElice cryptosystem is based around the usage of a binary Goppa code - though later derivations loosen the requirement of the type of code that is needed. Regardless, to properly describe any code-based cryptosystem, we need to introduce basic concepts in coding and information theory. For a more detailed introduction to coding theory, consult [3] for coding theory as applied to cryptography and [25] for a general introduction to the subject.

Let A be a nonempty set. We denote by A_n the set of all sequences of length n composed entirely of elements of A. A code C is a nonempty subset of A_n. Any element of C is referred to as a code vector, legal codeword, or codeword. Any element of A_n not in C is called an illegal codeword. In coding theory, it is often the case that $A = \mathbb{Z}_2$. Thus, any subset is then a set of bit strings.

The entire motivation behind error correcting codes is to reduce and fix errors in binary strings. To do that, we first need to be able to detect when there is an error. The Hamming distance is measure of the difference between two codewords c_1 and c_2. Symbolically written as $H(c_1, c_2)$, the hamming distance is the number of differing bits in each codeword. For example, suppose $c_1 = (1, 1, 0)$ and $c_2 = (0, 1, 1)$. Then $H(c_1, c_2) = 2$ since the bits in the first and third positions differ. The minimum distance between any two legal codewords of a code C is called the minimum Hamming Distance of C and is written as $d(C)$.

A linear code of dimension k and length n over a field $\mathbb{F}$ is a subspace of $\mathbb{F}^n$ of dimension k. This can also be referred to as a $[n, k]$ code. When the minimum Hamming distance d of the code needs to be specified, it can be referred to as a $[n, k, d]$ code. In other words, a linear code is a vector space spanned by k linearly independent vectors of $\mathbb{F}^n$. In most applications, $\mathbb{F} = \mathbb{Z}_2$.

With the groundwork laid out, we can define a Goppa code. A Goppa polynomial is a polynomial over the finite field of size p^m, denoted as $GF(p^m)$, where p is a prime number. Symbolically, a Goppa polynomial is a polynomial of the form

$$g(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_t x^t, \qquad g_i \in GF(p^m)$$

Let $L = \{a_1, \ldots, a_n\}$ be a finite subset of $GF(p^m)$ such that no element of L is a root of $g(x)$. For some codeword $c = (c_1, \ldots, c_n)$ of length n over a finite field $\mathbb{F}_q$ where q is prime, we can define a function

$$R_c(w) = \sum_{i=1}^{n} \frac{c_i}{x - a_i}$$

For mathematical subtleties, the function $f_i(x) = \frac{1}{x - a_i}$ represented above can be defined as the function satisfying the equation $f_i(x) * (x - a_i) \equiv 1 \pmod{g(x)}$ with degree less than the degree of $g(x)$.

A Goppa code $\Gamma(L, g(x))$ is the set of code vectors satisfying $R_c(x) \equiv 0 \pmod{g(x)}$. A binary Goppa code is a Goppa code where $g(x)$ is defined over $GF(2^m)$. Furthermore, if $g(x)$ is irreducible - it cannot be factored over $GF(p^m)$ - then it is a known theorem that the Goppa code based on $g(x)$ will have a minimum Hamming distance of $2t - 1$ where t is the degree of $g(x)$.

The McEliece encoding scheme makes use of three different types of matrices. As related to Goppa codes, a generator matrix is a $k \times n$ matrix G such that the rows of G form a basis of the Goppa code $\Gamma(L, g(x))$. A binary nonsingular matrix is a matrix over $\mathbb{F}_2$ with a nonzero determinant. Finally, a permutation matrix is a matrix over $\mathbb{F}_2$ such that each column and row will only contain a single 1.

The McEliece cryptosystem goes as follows. First, we select two natural numbers n and t where t is significantly less than n. We then create an irreducible, binary Goppa polynomial $g(x)$ of degree t over $GF(2^n)$. The resulting Goppa code $\Gamma(L, g(x))$ will have a minimum Hamming distance of $2t - 1$. We then determine a $k \times n$ generating matrix for $\Gamma(L, g(x))$, randomly select a square binary nonsingular matrix S of order k, and randomly select a square permutation matrix P of order n. We denote the product of these three matrices as $G_{pub} = SGP$ which has dimension $k \times n$.

The public key of this cryptosystem is the tuple (G_{pub}, t). The private key is the tuple (S, G, P). If Bob wants to send Alice a message, then he first needs to convert his plaintext to a vector $m \in \mathbb{F}_k$ and randomly choose a vector $z \in \mathbb{F}_n$ of Hamming weight t. The ciphertext is computed as $c = mG^{pub} \oplus z$ where $\oplus$ is the exclusive or operator. The decryption process requires a few more results from coding theory which we will not discuss here but can be found in [5].

The McEliece cryptosystem is by no means the only code-based cryptosystem. It is, however, one of the more famous ones and shows how results from studying error-correcting codes can create an encryption scheme. Another famous variant is the Niederreiter cryptosystem, but we will not discuss that here. More details about both cryptosystems can be found in [3], [5], and [2]. In general, research has also been done into encryption schemes based on low-density parsity checks (LDPC) and quasi-cyclic low-density parsity checks (QC-LDPC), but most encryption schemes in these categories were proven to be insecure.

8 HASH FUNCTIONS

Unlike the previously mentioned categories, hash functions are commonly used in modern cryptography in one form or another. For example, the SHA family of algorithms are commonly used for storing passwords so that they are not humanly readable and for digital signatures. Hash functions are designed to be easy to compute since they tend to be used often. They also need to highly sensitive to input change. In other words, if the two very similar messages are input to a hash function, say that the two messages differ by a single character, then the output - known as a hash message, digest, or fingerprint - will be vastly different. To make computation easier, the digest of a hash function needs to also be the same size regardless of the input size. In applications, this is usually between 128 to 512 bits.

From a security standpoint, a hash function $h(x)$ needs to satisfy three different requirements:

(1) $h(x)$ is a one-way function (Pre-image resistance)
(2) $h(x)$ has weak collision resistance
(3) $h(x)$ has strong collision resistance

Suppose that we have messages x_1, x_2 such that $y_1 = h(x_1)$ and $y_2 = h(x_2)$. Property 1 implies that, if we know the value of y_1, it is computationally infeasible to find x_1 even if we know the hash function used. Property 2 states that if $y_1 = y_2$ and we know the value of x_1, then it is difficult to find the value x_2. Finally, property 3 is a stronger version of property 2. If we have a digest value y, is it computationally infeasible to find two different input values that both produce y under h.

Hash functions partially stand out from the rest of the cryptographic primitives and algorithms because they are not necessarily designed around a hard mathematical problem. At most, many can be thought to use basic modular arithmetic to disguise their input. The SHA family algorithms are designed around a sequence of cycles in which the input message is split into blocks whose positions are swapped and have their bits shifted and occasionally put into

an exclusive or operation (which is equivalent to the modular addition of $\mathbb{Z}_2$). Because of their nature of shifting and adding, there is seemingly no number theoretical problem that needs to be solved to break every hash function. As a result, it is believed that many popular hash functions are quantum resistant. However, this does not mean that they are completely safe as SHA-1 has been found to be susceptible to collision attacks over the past several years. Both new and current hash functions will need to be researched to make sure they are not weak to attacks, but it is not believed at this current time that quantum computation will lend any significant benefit to breaking hash functions.

Since we believe hash functions will be useful in resisting quantum computers, the question now become how to use them. Because they are based on one-way functions without a trapdoor, they do not lend themselves well to encryption because they lack a decryption process. Instead, hash functions are useful in signing messages to provide an authentication process in cryptosystems. This allows for receiving parties to check to see if the message was tampered with in any way. We make use of signature schemes in modern public key cryptography, but they are usually associated with a major cryptosystem like RSA or ElGamal. Regardless of the cryptosytem used in the future, one option is to use one-time signature schemes, those that only rely on hash functions, to remain secure around quantum computers. Examples of one-time signature schemes include the Lamport-Diffie signature scheme and the Winternitz signature scheme, both of which are discussed in [5]. However, one downfall of these schemes is that each key pair generated in the process can only be used for one message. This can be fixed with the Merkle signature scheme which uses a binary hash tree to use a single key for a finite number of messages. It should be noted that the Merkle signature scheme still make use of one-time signature schemes to accomplish this task, so they should not be immediately discarded with regard to research.

9 CONCLUSION

A large change in technology will almost always have unwanted consequences. With quantum computing, cryptography gets the short end of the stick. The new computational abilities of quantum computers allow them to solve the number theoretic foundations of the field's most prominent cryptosystems. However, there are still plenty of viable options, we just need more research into both the security of these systems and the abilities of quantum computers. Lattice-based cryptography is probably one of the most realistic replacements for RSA with the speed and relatively small key size of NTRU. Additionally, it has the most research supporting it out of any quantum-resistant cryptosystem on the table. Code-based cryptosystems and multivariate signature schemes and cryptosystems show a lot of promise in certain areas for alternatives, but need more research in how to reduce key sizes or fix other problems before they can be implemented for any computational system. And finally, hash functions seem to be sticking around allowing for us to keep using something familiar for our digital signatures and other purposes. Quantum computing does not spell the end for cryptography, it just means the field needs to adapt to the new challenges. And once more research is done to test the security of these different primitives, the challenge then becomes how to securely phase out the last generation's cryptosystems like RSA and implement the new ones instead.

REFERENCES

[1] Scott A. Vanstone Alfred J. Menezes, Paul C. van Oorschot. 2001. *Handbook of Applied Cryptography* (5th. ed.). CRC Press.

[2] Marco Baldi. 2014. *QC-LDPC Code-Based Cryptography* (1st. ed.). Springer, Cham.

[3] Paulo S. L. M. Barreto, Felipe Piazza Biasi, Ricardo Dahab, Julio Cesar Lopez-Hernandez, Eduardo M. de Morais, Ana D. Salina de Oliveira, Geovandro C. C. F. Pereira, and Jefferson E. Ricardini. 2014. *A Panorama of Post-quantum Cryptography*. Springer International Publishing, Cham, 387–439. https://doi.org/10.1007/978-3-319-10683-0_16

[4] Jan Pelzl Christoff Paar. 2010. *Understanding Cryptography: A Textbook for Students and Practitioners* (2nd. ed.). Springer-Verlag Berlin Heidelberg.

[5] Erik Dahmen Daniel J. Bernstein, Johannes A. Buchmann. 2009. *Post-Quantum Cryptography*. Springer-Verlag Berlin Heidelberg.

[6] Richard M. Foote David S. Dummit. 2004. *Abstract Algebra* (3rd. ed.). John Wiley and Sons, Inc., Danvers, MA.

[7] Jintai Ding and Dieter Schmidt. 2005. Rainbow, a New Multivariable Polynomial Signature Scheme. In *Applied Cryptography and Network Security*, John Ioannidis, Angelos Keromytis, and Moti Yung (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 164–175.

[8] Artur K. Ekert. 1991. Quantum cryptography based on Bell's theorem. *Phys. Rev. Lett.* 67 (Aug 1991), 661–663. Issue 6. https://doi.org/10.1103/PhysRevLett.67.661

[9] Wolfgang Polak Eleanor Rieffel. 2011. *Quantum Computing: A Gentle Introduction* (1st. ed.). The MIT Press, Cambridge, MA.

[10] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. 1997. Public-Key Cryptosystems from Lattice Reduction Problems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '97)*. Springer-Verlag, London, UK, UK, 112–131. http://dl.acm.org/citation.cfm?id=646762.706185

[11] Charles H. Bennett and Gilles Brassard. 1984. WITHDRAWN: Quantum cryptography: Public key distribution and coin tossing. (01 1984), 175-179 pages.

[12] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 1998. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory*, Joe P. Buhler (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 267–288.

[13] R J. Mceliece. 1978. A Public-Key Cryptosystem Based on Algebraic Coding Theory. 44 (05 1978).

[14] Joseph H. Silverman Jeffrey Hoffstein, Jill Pipher. 2014. *An Introduction to Mathematical Cryptography* (2nd. ed.). Springer-Verlag Berlin Heidelberg.

[15] Dieter S. Schmidt Jintai Ding, Jason E. Gower. 2006. *Multivariate Public Key Cryptosystems* (2nd. ed.). Number 25 in Advances in Information Security. Springer-Verlag Berlin Heidelberg.

[16] Aviad Kipnis, Jacques Patarin, and Louis Goubin. 1999. Unbalanced Oil and Vinegar Signature Schemes. In *Advances in Cryptology — EUROCRYPT '99*, Jacques Stern (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 206–222.

[17] Hoi-Kwong Lo, Xiongfeng Ma, and Kai Chen. 2005. Decoy State Quantum Key Distribution. *Phys. Rev. Lett.* 94 (Jun 2005), 230504. Issue 23. https://doi.org/10.1103/PhysRevLett.94.230504

[18] T. Matsumoto and H. Imai. 1988. Public Quadratic Polynomial-tuples for Efficient Signature-verification and Message-encryption. In *Lecture Notes in Computer Science on Advances in Cryptology-EUROCRYPT'88*. Springer-Verlag New York, Inc., New York, NY, USA, 419–453. http://dl.acm.org/citation.cfm?id=55554.55593

[19] Isaac L. Chuang Michael A. Nielson. 2010. *Quantum Computation and Quantum Information* (7th. ed.). Cambridge University Press, New York, NY.

[20] H Niederreiter. 1986. Knapsack Type Cryptosystems and Algebraic Coding Theory. 15 (01 1986).

[21] H. Ong, C. P. Schnorr, and A. Shamir. 1984. An Efficient Signature Scheme Based on Quadratic Equations. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (STOC '84)*. ACM, New York, NY, USA, 208–216. https://doi.org/10.1145/800057.808683

[22] J. PATARIN. 1997. The oil and vinegar signature scheme. *Presented at the Dagstuhl Workshop on Cryptography September 1997* (1997). https://ci.nii.ac.jp/naid/10027921256/en/

[23] JM Pollard and Claus-Peter Schnorr. 1985. Solution of x2+ ky2 m (mod n), with application to digital signatures. *preprint* (1985).

[24] Valerio Scarani, Antonio Acín, Grégoire Ribordy, and Nicolas Gisin. 2004. Quantum Cryptography Protocols Robust against Photon Number Splitting Attacks for Weak Laser Pulse Implementations. *Phys. Rev. Lett.* 92 (Feb 2004), 057901. Issue 5. https://doi.org/10.1103/PhysRevLett.92.057901

[25] J.H. van Lint. 1992. *Introduction to Coding Theory* (2nd. ed.). Number 86 in Graduate Texts in Mathematics. Springer-Verlag Berlin Heidelberg.