CRACKING THE CODE:

UNDERSTANDING THE SHIFT TOWARDS

POST-QUANTUM CRYPTOGRAPHY

_____

A Thesis

Presented to

The Honors Tutorial College

Ohio University

_____

In Partial Fulfillment

of the Requirements for Graduation

from the Honors Tutorial College

with the degree of

Bachelor of Science in Mathematics

_____

by

Hannah Culver

April 2024

This thesis has been approved by

The Honors Tutorial College and the Department of Mathematics

_____

Dr. Sergio Lopez-Permouth

Professor, Mathematics

Thesis Advisor

_____

Dr. Alexei Davydov

Director of Studies, Mathematics

_____

Beth Novak

Interim Dean, Honors Tutorial

College

Abstract

The existence of post-quantum cryptography alludes to the existence of its predecessor: pre-quantum cryptography. In this thesis, we walk the reader through this progression from classical to quantum cryptography. We cover the two most promising forms of PQC and make it a point to emphasize the strength of lattices. Above all, we stress the gravity of encouraging the general public to be aware of this emerging field.

In pursuing this goal, we are joining the consensus in academic circles about the importance of bringing this information to contemporary students. It is worth noting the efforts being made at Ohio University to introduce these topics into the curriculum.

Acknowledgments

I would like to extend my sincerest gratitude to my thesis advisor, Dr. Sergio Lopez-Permouth, both for introducing me to the study of quantum computing and for really kick-starting my passion for this project.

I would also like to thank my director of studies, Dr. Alexei Davydov, as well as the Honors Tutorial College for selecting me to be in such a prestigious program. The trust that has been placed on me to drive my academia is not something I take lightly.

Tackling this project has by no means been an easy feat but a rewarding one nonetheless. As a proportionate reflection of the culmination of my time as an undergrad here at Ohio University, I am proud to share it with the community.

Thank you.

Table of Contents

List of Abbreviations/Acronyms*

***page number marks the first instance or definition**

List of Tables

List of Figures*

***all sourced from Wikipedia and Wikimedia Commons**

I. Introduction

Quantum computing has become a hot topic in recent years. Since the early 1980s, talk of building so-called “supercomputers” has slowly bubbled to the surface and pushed its way to the forefront of news, suggesting quicker and more powerful calculations than ever. This exciting feat–seemingly growing closer to soon becoming a reality–could not have come at a better time, too, as we are living in the golden age of technology. Nearly everyone has some sort of technological device and relies on it for means such as communication, browsing or searching the Internet, and keeping personal records. As a result, data is constantly being generated and needs to be kept safe and secure from outside influences.

Current methods set in place to ensure that data remains private between separate parties can be attributed to a field of research known as *cryptography*. This act of encrypting and decrypting code and coming up with a unique key made known only to the two parties at hand ensures that a third party is unable to listen in on or tamper with the messages wished to be kept private. You may be familiar with the classical example of Alice, Bob, and Eve. Alice and Bob want to exchange information across a secure line, while Eve attempts to eavesdrop on their conversation. This clever wordplay on the names can be extended to include an additional character, Mallory. Her malicious intentions are often described as wanting to possibly corrupt the data being shared between Alice and Bob.

Great lengths have been taken to deal with this problem, as evident by having an entire study dedicated to the matter. Normal computers themselves already have an intelligent built-in standard, depending on the extreme difficulty that comes with factoring unfathomably large integers. In some cases, this is believed to be impossible to achieve within a set timeframe if the integer is large enough, yet, as quantum computers and algorithms are approaching the existence of a practical reality and of a higher caliber, this seemingly foolproof measure is edging dangerously close to being cracked. Hence the need for post-quantum cryptography.

II. Background

Research on post-quantum cryptography is pertinent for a number of reasons and must be treated with the utmost care and precision. Precious data is at stake and is threatened by quantum cyber-security attacks. Many argue that this pressing issue needs to be addressed and dealt with immediately, yet others remain skeptical. In either case, because the research is still relatively new, there is room for investigation. This monograph aims to expand current knowledge on the subject by shedding light on an otherwise hypothetical topic and proposes a call to action to combat these threats.

To begin, we will start by considering some background information about cryptography. From there we will transition to what algorithms already exist in the realm of quantum computing that have helped immensely in this area of protecting data from potential breaches. Mitigating these threats posed by quantum computers is what post-quantum cryptography is set out to achieve. Lastly, we will look at and discuss the competition that was held by NIST (National Institute of Standards and Technology) in their search for adequate algorithms designed to tackle this problem.

A. Cryptography and Public Key Encryption

Present cryptography can be broken down into two main types: *symmetric* and *asymmetric* [28]. In the symmetric case, both the sender and the receiver use the same key and algorithm to manipulate their data. For example, Alice can encrypt a plaintext message using her shared secret key and Bob can decrypt the message using the same cryptographic algorithm Alice used and the same shared secret key. The key needs to be kept secret, meaning that only Alice and Bob should know it; therefore, an efficient way of exchanging secret keys over public networks is necessary. It is for this very reason that asymmetric cryptography was introduced.

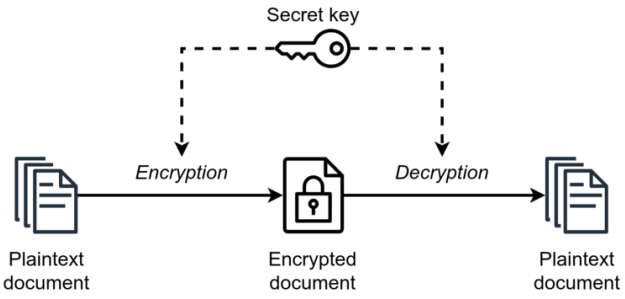

Figure 1. *Symmetric (AES): faster, but how to distribute secret key?* [24]

Unlike symmetric cryptography, *asymmetric* cryptography, or *public key cryptography* (PKC), is a form of encryption where the keys come in pairs. Each party should have its own private and public keys. For instance, if Bob wanted to encrypt a message, Alice would send her public key to Bob and then Bob could encrypt the message with Alice's public key. Next, Bob would transmit the encrypted message to

Alice who is then able to decrypt the message with her private key. Thus, the message is encrypted with a public key and only the person owning the private key can decrypt it. Mavroeidis et al. [28] further elaborate on asymmetric cryptography with an additional and common usage, that of digital signatures.

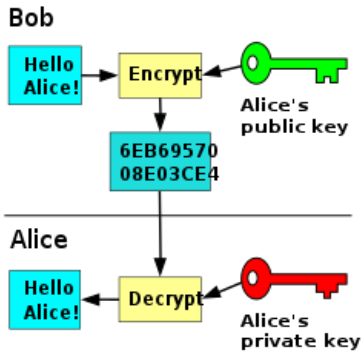

Figure 2. *Asymmetric public/private key (RSA): slow, but easy to share keys* [24]

Another form of cryptography, *RSA* (named after its founders Ron Rivest, Adi Shamir, and Leonard Adleman [35]) is a popular style of cryptography that allows for the secure transfer of information via the Internet [17]. The core conjecture of RSA cryptography is that multiplying two large prime numbers is a trapdoor function. In other words, multiplying the two numbers is easy, yet finding the two factors after the multiplication has occurred is hard. This one-way function can be likened to a hash function. More specifically, we want to design a way for an algorithm to move easily in one direction, but that is nearly impossible to reverse [14]. This crucial idea has been the crux of modern computers, making it challenging to find the inverse of a function given its output. If we were to express this mathematically, it would be likened to having a

function f that, when given an input x, yields an output y. On the other hand, if we were only provided the final solution y, how would we determine with certainty what value of x was inputted? In practice, we want this process to be very easy, simple, and straightforward in one direction, but not the other way around.

According to Paar and Pelzl [31], RSA and asymmetric algorithms in general are not meant to replace symmetric algorithms. This is because they are computationally costly. RSA is mainly used for secure key exchange between end nodes and is often used together with symmetric algorithms such as *AES*, or the advanced encryption standard, where the symmetric algorithm does the actual data encryption and decryption. Kirsch [22] states that RSA is theoretically vulnerable if a fast-factorizing algorithm is introduced or a huge increase in computation power can exist. The latter can be achieved using quantum mechanics on computers, known as quantum computers.

On par with the complexity of factoring is the *discrete logarithm problem* (DLP). Asymmetric cryptographic systems such as *Diffie-Hellman* (DH) and *Elliptic Curve Cryptography* (ECC) are based on DLP. The difficulty of breaking these cryptosystems is rooted in determining the integer r such that $g^r = x \ mod \ p$. The integer r is called the *discrete logarithm problem* of x to the base g, and we can write it as $r = log_g x \ mod \ p$. The discrete logarithm problem is very hard to compute if the parameters are large enough [28]. However, that did not stop others from devising their own workarounds.

B. Shor's and Grover's Algorithm

In 1994, mathematician Peter Shor from Bell Labs challenged the previous assumption that factoring integers with a thousand or more digits is practically impossible and instead proved that factorizing such a number would change fundamentally with a quantum computer [9, 37]. The procedure can be thought of as follows, involving some modular arithmetic and a process known as *period-finding* [39]:

Suppose we want to factor a large number N. Given N and another integer a, our goal is to find the smallest positive integer r such that $a^r - 1$ is a multiple of N. The number r is called the *period* of a modulo N. In modular arithmetic, the remainder of a division a/N is called the value of a modulo N and is denoted by $a \pmod N$. For example, $1 = 16 = 91 \pmod{15}$. Thus, the period a of modulo N is the smallest positive integer r such that $a^r = 1 \pmod N$.

Shor's algorithm in asymmetric cryptography can be used for solving discrete logarithm problems. Vazirani [40] explored in greater detail the methodology of Shor's algorithm and showed that by starting from a random superposition state of two integers, and by performing a series of Fourier transformations, a new superposition can be set up to give us a high probability of two integers that satisfy an equation. By using this equation we can calculate the value r, the unknown "exponent" in the DLP [28].

Just two years later, Lov Grover created an algorithm that uses quantum computers to search unsorted databases [16]. The algorithm can find a specific entry in an unsorted database of N entries in $\sqrt{N}$ searches. In comparison, a conventional computer would need $N/2$ searches to find the same entry. Bone and Castro [11]

remarked on the impact of a possible application of Grover's algorithm to crack the *Data Encryption Standard* (DES), its security reliant upon a 56-bit key. The authors added that the algorithm needs only 185 searches to find the key.

The discovery of Grover's algorithm showed that quantum computers have a quadratic speed-up in searching databases compared to classical computers. At the center of the algorithm, there is the use of quantum superposition, optimal in applying parallel execution, and well-known for applying unstructured search with a high probability of unique output [3]. Currently, to prevent password cracking, we increase the number of key bits to produce a larger key space. As a result, the number of searches needed to crack a password increases exponentially. Bernstein [9] stated that Grover's algorithm has some applications for symmetric cryptosystems, but it is not as fast as Shor's algorithm.

C. BB84 and EK91 Protocols

We now bring back Alice, Bob, and Eve when discussing two different protocols. Both have been designed to keep Eve's eavesdropping at bay as Alice and Bob share secret information. To do so, they first need to generate a shared, random, secret key that will allow them to encrypt and decrypt messages they send to and receive from each other. BB84 and EK91 rely on some special characteristics of quantum behavior to derive a sort of algorithm Alice and Bob can utilize, that of superposition and entanglement. We will first consider BB84.

Charles Bennett and Gilles Brassard came up with a scheme that takes advantage of the fact that taking a quantum measurement alters the state of the variable being measured in order to foil a potential eavesdropper [26]. The procedure, shortened to BB84 in relation to its inventors and founding year, follows a series of steps with one objective in mind: to "generate a random bit sequence that is only known to Alice and Bob and for which they can check if the transmission has been listened to" [36]. The steps are laid out as such:

> Assuming the pathway of communication is from Alice to Bob, Alice begins by generating two random binary strings of length $4n$, one of which is a key k consisting of 1s and 0s and the other a sequence of bases used to represent the state space of a qubit, such as the spin of an electron* or the polarization of a photon. For this example, we will imagine the qubits as electrons so that we can denote the bases as either V (for vertical spin) or H (for horizontal spin).

Aside: In practice, it is highly difficult to achieve true randomness, which is but one of the many underlying discrepancies between classical mechanics and quantum mechanics. In classical mechanics, there exists no real randomness, but *sensitive dependence to initial conditions* [8]. In other words, a slight change in the input can get amplified and produce an entirely new or unexpected outcome, merely an *illusion* of randomness.

Suppose Alice has a large number of said qubits at her disposal [36]. For each one, she chooses one of the two bases, V or H, at random and with equal probability (these make up her second binary string). After determining the orientation–or state–of the qubit, Alice sends the qubits off to Bob.

Before receiving any qubits, Bob also generates a random string of bases. Upon receiving each qubit, he measures it using the corresponding basis and records the result. Once completed, Bob should have his own sequence of 1s and 0s.

Now that Alice and Bob both have a string of bits and of bases, they share their sequence of Vs and Hs publicly and look for any matches, discarding the portion of bases that disagree. This should yield a new string of roughly length $2n$.

Still on a public channel, Alice and Bob take half of this new string and compare the corresponding bits. Assuming Eve did not attempt to intercept the exchange at any point, there should be no disturbances and the bits should match. If so, Eve was probably not spying and Alice and

Bob can discard this current string since it was shared publicly, keeping the remaining key of length n that has been kept private the whole time. If not, or if they have reason to believe Eve may have intervened, they must either find a new key or try again later.

***To be more accurate and to not entirely mislead the reader, spin here is easier to visualize, yet it is really equated to an electron's charge or energy level.**

1. NO EAVESDROPPING

Alice's string of binary digits and sequence of bases (initial length $4n = 20$):

0	0	1	0	0	1	1	0	0	0	1	0	0	1	0	0	0	1	0	1
V	H	V	V	H	V	V	V	V	V	V	V	V	H	H	H	V	V	V	V

Bob's sequence of bases and respective measurements (same initial length):

H	H	H	H	H	H	H	V	H	V	V	H	H	H	V	V	H	V	V	H
0	0	0	1	0	1	0	0	0	0	1	1	1	1	1	0	0	1	0	1

After getting rid of the mismatched bases (as depicted by the grayed-out boxes)...

Alice's *new* string of binary digits and sequence of bases (updated length $2n = 8$):

0	0	0	0	1	1	1	0
H	H	V	V	V	H	V	V

Bob's *new* sequence of bases and respective measurements ($2n = 8$):

H	H	V	V	V	H	V	V
0	0	0	0	1	1	1	0

Confirming that the remaining bits match, or at least enough to safely conclude Eve was not eavesdropping, Alice and Bob can discard half of their bits and use the rest as a private key $k = 0011$. Theoretically, this key would be a lot longer in real applications, but more important to note is how the final length $n = 4$ is roughly ¼ of the original length. This is intentional, starting with a key of a longer, specified length that slowly dwindles down to a final subset of length n as the protocol continues.

2. EAVESDROPPING

Alice's string of binary digits and sequence of bases (initial length $4n = 20$):

1	1	0	1	1	1	1	1	0	0	1	0	1	0	0	1	0	1	1	0
V	H	H	V	H	H	H	V	V	V	H	V	V	H	V	H	H	H	H	V

Eve's sequence of bases and respective measurements (chosen at random intervals):

H					V				H			H				H	V		V
0					1				0			1				0	1		0

Bob's sequence of bases and respective measurements (initial length $4n = 20$):

V	V	V	H	H	V	V	V	V	H	V	H	V	H	V	V	H	H	V	H
1	0	0	1	1	1	1	1	0	0	1	0	0	0	0	1	0	0	0	1

After getting rid of the grayed-out boxes (same as before)...

Alice's *new* string of binary digits and sequence of bases (updated length $2n = 9$):

1	1	1	0	1	0	0	0	1
V	H	V	V	V	H	V	H	H

Bob's *new* sequence of bases and respective measurements (updated length $2n = 9$):

V	H	V	V	V	H	V	H	H
1	1	1	0	0	0	0	0	0

In some cases, Eve's measurements have caused errors that might strike Alice and Bob as suspicious when comparing their bits, as illustrated by the fifth and ninth boxes where the bits are unequal. This could just be due to noise that they need to

correct and account for, but it is better to be safe than sorry, and so Alice and Bob ultimately decide to abandon this session.

Of course, this operation is not completely foolproof, and there is always the possibility of errors happening during transmission even with the absence of an eavesdropper that are simply out of Alice and Bob's control, but it is a good starting point. So comes along an approach that avoids the delivery of qubits altogether.

Artur Ekert, in 1991, postulated that if both Alice and Bob have a stream of qubits that are entangled, meaning they behave as a single quantum system [41], then they could each measure their qubits with one of three different bases. This property of entanglement occurs when a pair of entangled particles are created simultaneously. When they become separated, the influence of one member of the pair over the other persists. As soon as the state of one of these particles is realized, the other is determined instantaneously, regardless of distance [41]. This is the basis of Ekert's EK91 protocol for establishing a secret shared key.

Similar to the BB84 protocol, Alice and Bob are responsible for keeping track of the result and the basis that they choose for each measurement. After $3n$ measurements, they announce the sequences of bases that they chose over an insecure channel, as the result is still hidden. They will logically agree on approximately n of them, and, so long as Eve is not snooping around, they will have made the opposite measurement in each place they have selected the same basis [36, 41]. Thus, the resulting string of 1s and 0s becomes their key [9].

Provided Alice and Bob picked the orientations of their detections randomly and independently, the correlation between their results should come out to be exactly $-2\sqrt{2}$ [36]. On the other hand, had Eve been present, she would have had to make measurements as well, unentangling the particles and disrupting the system. A significant departure from the above calculation would indicate this [41].

3. NO EAVESDROPPING

Qubit pair no.	Alice's measurements in direction…			Qubit pair no.	Bob's measurements in direction…		
	a	b	c		a	b	c
1			1	1			0
2		0		2		1	
3	1			3		0	
4			0	4		0	
5		1		5		0	
6	0			6			0
7	1			7		0	
8			1	8			0
9	0			9			1
10			0	10			1
11		1		11	1		
12	0			12	0		
13	0			13		1	
14			1	14			0
15	1			15		0	
16	0			16	0		
17	0			17		1	
18	1			18	0		
19	1			19		1	
20	0			20	0		
21		0		21		1	
22			1	22	0		
23		1		23			0
24		0		24	0		

Table 1. *EK91 No Eavesdropping* [36]

As shown in the table, the white cells are when Alice and Bob measured in the *same* directions. Likewise, the gray cells are for measurements made in *different* directions. Because this is the case of no eavesdropping from an outside source, they can use the resulting string of digits as their key $k = 1011010$ of length $n = 7$, approximately ⅓ of the original $3n = 24$.

The great feature of EK91 is that the process itself generates the key [8]. Nothing needs to be stored beforehand, therefore eliminating one of the paramount security threats to encryption. The protocol has even been successfully carried out in the lab using entangled photons [8, 41].

III. Post-Quantum Cryptography

Post-quantum cryptography can be split into five different categories, or families, in relation to the *Internet of Things* (IoT): lattice, hash, code, multivariate, and super-singular [23]. Others exclude super-singular and instead focus on the first four. We will only be looking at the lattice and hash-based signatures, as those were the ones implemented in the algorithms standardized by NIST (discussed in the following section) and have shown the most promise in quantum-resistant cryptosystems.

A. Lattice

Miklos Ajtai [1] was the first to demonstrate lattice-based algorithms, with the suggestion of designing stable cryptographic algorithms based on the NP-hard lattice problem. [2] A lattice-based public-key encryption scheme was adopted [2], but one that was sufficiently robust and proven stable was not presented until 2005 by Oded Regev. His proposed method uses both lattices and a generalization of the problem of parity learning [2], a problem in machine learning that involves finding a function that computes the parity of bits at specific locations.

A lattice, given in n-dimensional vector space, is an infinite structure of particularly arranged points–or dots–that is periodic in nature. These two-dimensional algebraic constructs come with a variety of uses and qualities, including not being easily defeatable with quantum computing schemes. Lattice-based cryptographic algorithms are mostly based on either the *closest vector problem* (CVP), which requires finding the point in the grid that is nearest to a fixed central point in the space–the origin–or the more vital *shortest vector problem* (SVP). This is easy to solve with relatively few dimensions, but, as the number of dimensions grows, quantum machines struggle to solve the problem effectively [4].

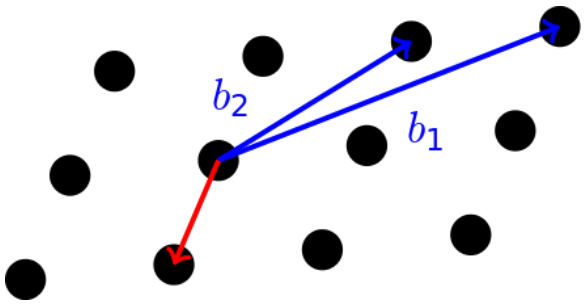

Figure 3. *Illustration of the shortest vector problem (basis vectors in blue, shortest vector in red)* [Schmittner, 2015]

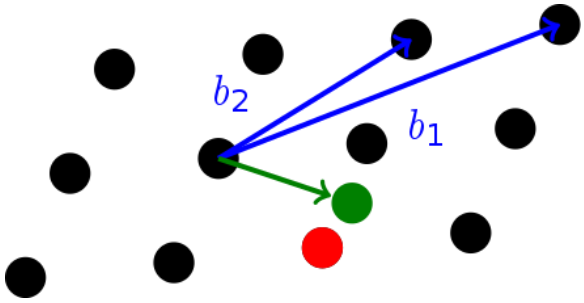

Figure 4. *Illustration of the closest vector problem, a generalization of the SVP (basis vectors in blue, external vector in green, closest vector in red)* [Schmittner, 2015]

In most lattice-based cryptographic algorithms, the cryptographic builders used are very time-efficient and simple, while still providing security proofs based on the worst-case hardness [32]. Oftentimes, a handful of the simple problems used in these cryptographic algorithms tend to be quantum resistant, since they are not based on any of the complicated problems solved by Shor's algorithm [38]. This results in one of a few types of algorithms that are believed to carry promise as potential candidates for post-quantum cryptography.

Several security specialists and scholars agree that the lattice-based cryptography algorithm is the path forward to deliver quantum-resistant encryption and, albeit vigorous. The fact that lattice-based cryptography provides fast, quantum-safe, fundamental primitives, and enables the construction of primitives previously thought impossible, makes it the top contender for IoT applications [18].

B. Hash

A hash-based signature scheme stems from a *one-time signature* (OTS), where each key pair only needs to be used to sign a message [10]. If an OTS key pair signs two different notes, this threatens the network, as a hacker could easily fake signatures to expose an individual's personal details. Ralph Merkle took inspiration from Lamport [25] and its variations. Merkle [19, 30] recommended that a binary hash tree, later named the Merkle tree, be used to create a many-time signature scheme. The leaves, or the bottom most nodes, are the hash values of OTS public keys. Each inner node is measured as the hash of the concatenation of its two child nodes. If a collision tolerant hash function is used, all leaf nodes [33] can be authenticated using the root node.

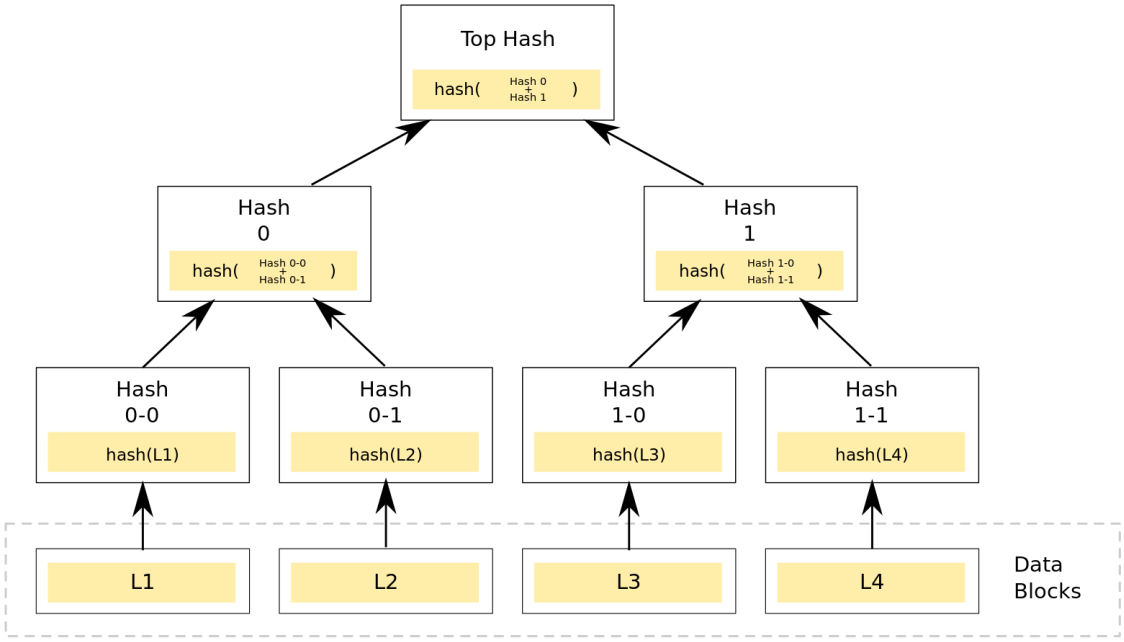

Figure 5. *An example of a binary hash tree. Hashes 0-0 and 0-1 are the hash values of data blocks L1 and L2, respectively, and hash 0 is the hash of the concatenation of hashes 0-0 and 0-1.* [Göthberg, 2005; Azaghal, 2012]

In a *Merkle signature scheme* (MSS), the root node of the Merkle tree turns into a public key and the set of all OTS hidden keys becomes the secret key. Random bit strings are the hidden keys for hash-based OTS. Therefore, one can store a short seed and (re)generate the OTS secret keys using a cryptographically protected pseudo-random generator instead of storing all OTS secret keys [29]. To prevent reuse of OTS key pairs, they are used according to the order of the leaves, starting with the leftmost leaf [20]. To do this, the scheme keeps the index of the last used OTS key pair as an internal state or condition [4].

IV. NIST Competition

In December of 2016, NIST called for proposals to standardize quantum-secure cryptographic primitives, initiating “a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms” [13]. The competition spanned over six years, overall hosting four rounds of submissions and subsequent eliminations. Round 1 saw a hefty total of 82 algorithms, 69 of which were accepted as “complete and proper” [34].

The federal agency’s latest announcement involved four different quantum-resistant cryptographic algorithms, all of which are based on structured lattices and hash functions. Furthermore, they “rely on math problems that both conventional and quantum computers should have difficulty solving, thereby defending privacy both now and down the road” [12].

The announcement was made on a page from NIST's very own website [12]:

> **For general encryption,** used when we access secure websites, NIST has selected the CRYSTALS-Kyber algorithm. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation.
>
> **For digital signatures,** often used when we need to verify identities during a digital transaction or to sign a document remotely, NIST has selected the three algorithms CRYSTALS-Dilithium, FALCON and SPHINCS+ (read as "Sphincs plus"). Reviewers noted the high efficiency of the first two, and NIST recommends CRYSTALS-Dilithium as the primary algorithm, with FALCON for applications that need smaller signatures than Dilithium can provide. The third, SPHINCS+, is somewhat larger and slower than the other two, but it is valuable as a backup for one chief reason: It is based on a different math approach than all three of NIST's other selections.

The following sections are taken from each of the finalists' home webpages [5, 7, 6, 15, 21].

A. CRYSTALS

The authors of the CRYSTALS suite define it in the following manner [5]:

> The "Cryptographic Suite for Algebraic Lattices" (CRYSTALS) encompasses two cryptographic primitives: Kyber, an IND-CCA2-secure *key-encapsulation mechanism* (KEM); and Dilithium, a strongly EUF-CMA-secure digital signature algorithm. Both algorithms are based on hard problems over module lattices, are designed to withstand attacks by large quantum computers, and have been submitted to the NIST post-quantum cryptography project.

The authors continue by expanding more on module lattices [5] and introducing the Ring-LWE problem [27], a variant of the LWE, or *learning-with-errors*, problem:

> Module lattices can be thought of as lattices that lie between the ones used in the definitions of the LWE problem, and those used for the Ring-LWE problem. If the ring underlying the module has a sufficiently high degree (like 256), then these lattices inherit all the efficiency of the ones used in the Ring-LWE problem, and additionally have the following advantages, when used in our cryptographic algorithms:
>
> - The only operations required for Kyber and Dilithium for all security levels are variants of Keccak, additions/multiplications in Z_q for a fixed q, and the NTT (*number theoretic transform*) for the ring $Z_q[X]/(X^{256}+1)$.
>
>   This means that increasing/decreasing the security level involves

> virtually no re-implementation of the schemes in software or hardware. Changing a few parameters is all that one needs to convert an optimized implementation for one security level into an optimized implementation for a different one.
>
> - The lattices used in Kyber and Dilithium have less algebraic structure than those used for Ring-LWE and are closer to the unstructured lattices used in LWE. It is therefore conceivable that if algebraic attacks against Ring-LWE appear (there are none that we are aware of at this point), then they may be less effective against schemes like Kyber and Dilithium.

Keccak is a hash function.

1. CRYSTALS-Kyber

On their website, the creators of CRYSTALS-Kyber refer to it solely as Kyber [7]:

> Kyber is an IND-CCA2-secure key encapsulation mechanism (KEM), whose security is based on the hardness of solving the learning-with-errors (LWE) problem over module lattices. Kyber is one of the finalists in the NIST post-quantum cryptography project. The submission lists three different parameter sets aiming at different security levels. Specifically, Kyber-512 aims at security roughly equivalent to AES-128, Kyber-768 aims at security roughly equivalent to AES-192, and Kyber-1024 aims at security roughly equivalent to AES-256.

The following table defines the actual LWE cryptosystem, though we note that we are not fully familiar with the mathematical notation.

Parameters:	➢ n, q, m positive integers, ➢ $\alpha \in R$ (real numbers) such that $0 < \alpha < 1$, and ➢ $\chi = D_Z$, discrete distribution over Z (integers)
Private Key:	$s \in Z_q^n$ uniformly at random
Public Key:	select m vectors $a_1, \ldots, a_m \in \Sigma\, Z_q^n$ independently according to the uniform distribution, then draw $e_1, \ldots, e_m \in Z$ from χ and get the public key $\{a_i, b_i\}_{i=1}^m$, with $b = \langle a_i, s \rangle + e_i \; mod \; q$
Encryption:	let $\mu \in \{0,1\}$ be the bit to encode, choose a random set $S \subset [m]$, then to encrypt μ one sends $(a, b) = (\Sigma_{i \in S} a_i \Sigma_{i \in S} b_i + \mu \frac{q}{2})$
Decryption:	if $b - \langle a, s \rangle$ is closer to 0 than to $\frac{q}{2} \; mod \; 0$ output 0, otherwise decrypt as 1

Table 2. *LWE-Cryptosystem Algorithm* [4]

The website further asserts Kyber's unique background as such [7]:

> The design of Kyber has its roots in the seminal LWE-based encryption scheme of Regev. Since Regev's original work, the practical efficiency of LWE encryption schemes has been improved by observing that the secret in LWE can come from the same distribution as the noise and also noticing that "LWE-like" schemes can be built by using a square (rather than a rectangular) matrix as the public key. Another improvement was applying an idea originally used in the NTRU cryptosystem to define the Ring-LWE and Module-LWE problems that used polynomial rings rather than integers. The CCA-secure KEM Kyber is built on top of a CPA-secure cryptosystem that is based on the hardness of Module-LWE.

Below is some further specification on the NTRU cryptosystem.

Parameters:	➢ n power of 2, ➢ $f(X) = X^n + 1$ and q odd sufficiently large, ➢ we define $R = Z[X]/f(X)$ and $R_q = \frac{R}{qR}$
Private Key:	$s, g \in R$ short polynomial (i.e. small coefficients) such that s is invertible mod q and mod 2
Public Key:	$h = 2g \cdot s^1 \in R_q$ with $g \in R$ short polynomial
Encryption:	choose a short $e \in R$ such that $e \ mod \ 2$ encodes the desired bit, choose $r \in R_q$ randomly and compute the ciphertext $c = h \cdot r + e \in R_q$ accordingly
Decryption:	multiply the ciphertext with the secret key to get $cs = 2gr + es \in R_q$, lift it in R as $2gr + es$ (possible if the following variables (i.e. g, r, e, s) are short enough compared to q and reduce it to obtain es and, therefore, the initial bit

Table 3. *NTRU Encryption Scheme* [4]

Finally, Kyber's creators provide an overview of how well the algorithm executes [7]:

> The tables below gives an indication of the performance of Kyber. All benchmarks were obtained on one core of an Intel Core-i7 4770K (Haswell) CPU. We report benchmarks of two different implementations: a C reference implementation and an optimized implementation using AVX2 vector instructions.

For clarity, we have highlighted the C reference implementation in yellow and the AVX2 vector instructions in red.

Kyber-512					
Sizes (in bytes)		**Haswell cycles (ref)**		**Haswell cycles (avx2)**	
sk:	1632	**gen:**	122684	**gen:**	33856
pk:	800	**enc:**	154524	**enc:**	45200
ct:	768	**dec:**	187960	**dec:**	34572

Table 4. *Kyber-512 performance overview* [7]

Kyber-768					
Sizes (in bytes)		**Haswell cycles (ref)**		**Haswell cycles (avx2)**	
sk:	2400	**gen:**	199408	**gen:**	52732
pk:	1184	**enc:**	235260	**enc:**	67624
ct:	1088	**dec:**	274900	**dec:**	53156

Table 5. *Kyber-768 performance overview* [7]

Kyber-1024					
Sizes (in bytes)		**Haswell cycles (ref)**		**Haswell cycles (avx2)**	
sk:	3168	**gen:**	307148	**gen:**	73544
pk:	1568	**enc:**	346648	**enc:**	97324
ct:	1568	**dec:**	396584	**dec:**	79128

Table 6. *Kyber-1024 performance overview* [7]

pk and sk are called the encryption key and decryption key, respectively.

ct stands for ciphertext.

All algorithms: key (gen)eration, (enc)ryption, and (dec)ryption [42]

2. CRYSTALS-Dilithium

Authors of the CRYSTALS-Dilithium algorithm commence with an introduction [6]:

> Dilithium is a digital signature scheme that is strongly secure under chosen message attacks based on the hardness of lattice problems over module lattices. The security notion means that an adversary having access to a signing oracle cannot produce a signature of a message whose signature he hasn't yet seen, nor produce a different signature of a message that he already saw signed. Dilithium is one of the candidate algorithms submitted to the NIST post-quantum cryptography project.

More is realized by the authors as they delve into Dilithium's design origins [6]:

> The design of Dilithium is based on the "Fiat-Shamir with Aborts" technique of Lyubashevsky which uses rejection sampling to make lattice-based Fiat-Shamir schemes compact and secure. The scheme with the smallest signature sizes using this approach is the one of Ducas, Durmus, Lepoint, and Lyubashevsky which is based on the NTRU assumption and crucially uses Gaussian sampling for creating signatures. Because Gaussian sampling is hard to implement securely and efficiently, we opted to only use the uniform distribution. Dilithium improves on the most efficient scheme that only uses the uniform distribution, due to Bai and Galbraith, by using a new technique that shrinks the public key by more than a factor of 2. To the best of our knowledge, Dilithium has the smallest public key + signature size of any lattice-based signature scheme that only uses uniform

> sampling.

To conclude, the authors present a clear and concise analysis on Dilithium's administration [6]:

> The table below gives an indication of the performance of the Dilithium with all the updates we applied to the parameter sets for round-3 of the NIST PQC project. All benchmarks were obtained on one core of an Intel Core-i7 6600U (Skylake) CPU. We report benchmarks of two different implementations: a C reference implementation and an optimized implementation using AVX2 vector instructions.

Again, we have taken the liberty of color coordinating these implementations, placing the C reference in yellow and AVX2 in red.

Dilithium2					
Sizes (in bytes)		**Skylake cycles (ref)**		**Skylake cycles (avx2)**	
		gen:	*300751*	**gen:**	*124031*
pk:	*1312*	**sign:**	*1355434*	**sign:**	*333013*
sig:	*2420*	**verify:**	*327362*	**verify:**	*118412*

Table 7. *Dilithium2 performance overview* [6]

Dilithium3					
Sizes (in bytes)		**Skylake cycles (ref)**		**Skylake cycles (avx2)**	
		gen:	*544232*	**gen:**	*256403*
pk:	*1952*	**sign:**	*2348703*	**sign:**	*529106*
sig:	*3293*	**verify:**	*522267*	**verify:**	*179424*

Table 8. *Dilithium3 performance overview* [6]

Dilithium5					
Sizes (in bytes)		**Skylake cycles (ref)**		**Skylake cycles (avx2)**	
		gen:	*819475*	**gen:**	*298050*
pk:	*2592*	**sign:**	*2856803*	**sign:**	*642192*
sig:	*4595*	**verify:**	*871609*	**verify:**	*279936*

Table 9. *Dilithium5 performance overview* [6]

Here sig denotes signature.

B. FALCON

The following is taken from under the 'about' heading on the Falcon website [15]:

> Falcon is based on the theoretical framework of Gentry, Peikert and Vaikuntanathan for lattice-based signature schemes. We instantiate that framework over NTRU lattices, with a trapdoor sampler called "fast Fourier sampling". The underlying hard problem is the *short integer solution* problem (SIS) over NTRU lattices, for which no efficient solving algorithm is currently known in the general case, even with the help of quantum computers.

Researchers have noted the following about FALCON, or *Fast-Fourier Lattice-based Compact Signatures over NTRU* [15]:

> Falcon offers the following features:
>
> - **Security:** a true Gaussian sampler is used internally, which guarantees negligible leakage of information on the secret key up to a practically infinite number of signatures (more than 2^{64}).
> - **Compactness:** thanks to the use of NTRU lattices, signatures are substantially shorter than in any lattice-based signature scheme with the same security guarantees, while the public keys are around the same size.
> - **Speed:** use of fast Fourier sampling allows for very fast implementations, in the thousands of signatures per second on a common computer; verification is five to ten times faster.

- **Scalability:** operations have cost $O(n\ log\ n)$ for degree n, allowing the use of very long-term security parameters at moderate cost.
- **RAM Economy:** the enhanced key generation algorithm of Falcon uses less than 30 kilobytes of RAM, a hundredfold improvement over previous designs such as NTRUSign. Falcon is compatible with small, memory-constrained embedded devices.

Similar to both CRYSTALS-Kyber and CRYSTALS-Dilithium, the researchers working on Falcon leave a table detailing the algorithm's performance [15]:

> While resistance to quantum computers is the main drive for the design and development of Falcon, the algorithm may achieve significant adoption only if it is also reasonably efficient in our current world, where quantum computers do not really exist. Using the reference implementation on a common desktop computer (Intel® Core® i5-8259U at 2.3 GHz, TurboBoost disabled), Falcon achieves the following performance:

variant	keygen (ms)	keygen (RAM)	sign/s	verify/s	pub size	sig size
FALCON-512	*8.64*	*14336*	*5948.1*	*27933.0*	*897*	*666*
FALCON-1024	*27.45*	*28672*	*2913.0*	*13650.0*	*1793*	*1280*

Table 10. *FALCON performance overview* [15]

The website goes on to say the following [15]:

> Sizes (key generation RAM usage, public key size, signature size) are expressed in bytes. Key generation time is given in milliseconds. Private key size (not listed above) is about three times that of a signature, and it could be theoretically compressed down to a small PRNG seed (say, 32 bytes), if the signer accepts to run the key generation algorithm every time the key must be loaded.
>
> To give a point of comparison, Falcon-512 is roughly equivalent, in classical security terms, to RSA-2048, whose signatures and public keys use 256 bytes each. On the specific system on which these measures were taken, OpenSSL's thoroughly optimized assembly implementation achieves about 1140 signatures per second; thus, Falcon's reference implementation, which is portable and uses no inline assembly on x86 CPUs, is already more than five times faster, and it scales better to larger sizes (for long-term security).

Byte-sized items are put in blue.

C. SPHINCS+

The SPHINCS+ page describes the scheme succinctly [21]:

> SPHINCS+ is a stateless hash-based signature scheme, which was submitted to the NIST post-quantum crypto project.

To be *stateless* means one need not keep track of (say, by using a counter of some sort) or update any state when spawning a signature. The site continues [21]:

> The design advances the SPHINCS signature scheme, which was presented at EUROCRYPT 2015. It incorporates multiple improvements, specifically aimed at reducing signature size.

From here reviewers mention what all was exactly submitted under the SPHINCS+ package [21]:

> The submission proposes three different signature schemes:
>
> - SPHINCS+-SHAKE256
> - SPHINCS+-SHA-256
> - SPHINCS+-Haraka
>
> These signature schemes are obtained by instantiating the SPHINCS+ construction with SHAKE256, SHA-256, and Haraka, respectively.
>
> The second round submission of SPHINCS+ introduces a split of the above three signature schemes into a simple and a robust variant for each choice of hash function. The robust variant is exactly the SPHINCS+ version from the first round

> submission and comes with all the conservative security guarantees given before. The simple variants are pure random oracle instantiations. These instantiations achieve about a factor three speed-up compared to the robust counterparts. This comes at the cost of a purely heuristic security argument.

V. Conclusion

There are many avenues within quantum computing that are appealing, such as its ability to run large-scale programs in a fraction of the time a conventional computer can. Many have come to appreciate what this field has to offer, an assurance of sorts. Others have remained curious, expressing concern about what implications it has for its cryptographical counterpart. While the two subjects may both still be in their infancy, they have come to evolve and accelerate alongside one another, propelling forward the scientific community and making for quite enticing research premises.

At the heart of this project is a mission: a mission to inform the general body of how far along quantum computing really is and what advancements in the field could mean for the future of our data. As post-quantum cryptography tries to manufacture cryptographic schemes that will not be trivial to unravel even with such refined apparatus, it is worth contemplating what a shift towards that ideal might entail. The objective is not to raise or spread alarm but awareness and to highlight the importance of being readily prepared as we continue this journey.

References

[1] Ajtai, M. *Generating hard instances of lattice problems*. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 99–108.

[2] Ajtai, M. *Representing hard lattices with O (n log n) bits*. In Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005; pp. 94–103.

[3] Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., ... & Liu, Y. K. (2022). *Status report on the third round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, NIST.

[4] Asif, R. (2021, February 5). *Post-Quantum Cryptosystems for Internet-of-Things: A Survey on Lattice-Based Algorithms*. IoT 2, no. 1: 71-91. https://doi.org/10.3390/iot2010005

[5] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., & Stehle, D. (2022, February 25). *Crystals*. CRYSTALS | Cryptographic Suite for Algebraic Lattices. https://pq-crystals.org/

[6] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., & Stehle, D. (2022, February 25). *Crystals*. Dilithium. (2021, February 16). https://pq-crystals.org/dilithium/index.shtml

[7] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., & Stehle, D. (2022, February 25). *Crystals*. Kyber. (2020, December 23). https://pq-crystals.org/kyber/index.shtml

[8] Bernhardt, C. (2019, February 22). *Quantum Computing for Everyone*. MIT Press.

[9] Bernstein, D. J. (2009). *Introduction to post-quantum cryptography* (pp. 1-14). Springer Berlin Heidelberg.

[10] Bernstein, D.J.; Hopwood, D.; Hülsing, A.; Lange, T.; Niederhagen, R.; Papachristodoulou, L.; Schneider, M.; Schwabe, P.; Wilcox-O'Hearn, Z. *SPHINCS: Practical stateless hash-based signatures*. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 17–21 October 2015; pp. 368–397.

[11] Bone, S., & Castro, M. (1997). *A brief history of quantum computing*. Imperial College in London, http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/spb3.

[12] Boutin, C. (2022, July 7). *NIST Announces First Four Quantum-Resistant Cryptographic Algorithms*. NIST. Retrieved from https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms

[13] Computer Security Division, I. T. L. (2017, January 3). *Post-Quantum Cryptography*. CSRC. Retrieved from https://csrc.nist.gov/projects/post-quantum-cryptography

[14] Dušek, M., Lütkenhaus, N., & Hendrych, M. (2006). *Quantum cryptography*. Progress in optics, 49, 381-454.

[15] Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., & Zhang, Z. (2017, November 30). *Falcon*. FALCON | Fast-Fourier Lattice-based Compact Signatures over NTRU. https://falcon-sign.info/

[16] Grover, L. K. (1996, July). *A fast quantum mechanical algorithm for database search*. In Proceedings of the twenty-eighth annual ACM symposium on theory of computing (pp. 212-219).

[17] Hidary, J. D. (2019, August 29). *Quantum Computing: An Applied Approach*. Springer Nature Switzerland AG.

[18] Hoffstein, J.; Howgrave-Graham, N.; Pipher, J.; Whyte, W. *Practical lattice-based cryptography: NTRUEncrypt and NTRUSign*. In The LLL Algorithm; Springer: Berlin/Heidelberg, Germany, 2009; pp. 349–390.

[19] Hofheinz, D.; Jager, T. *Tightly secure signatures and public-key encryption*. Des. Codes Cryptogr. 2016, 80, 29–61.

[20] Huelsing, A.; Butin, D.; Gazdag, S.; Rijneveld, J.; Mohaisen, A. *Xmss: Extended Merkle Signature Scheme*; Technical Report; Internet Research Task Force: Wilmington, DE, USA, 2018.

[21] Hülsing, A., Aumasson, J.-P., Bernstein, D. J., Beullens, W., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.-L., Kampanakis, P., Kölbl, S., Kudinov, M., Lange, T., Lauridsen, M. M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., & Westerbaan, B. (2023, August 2). *SPHINCS+*. SPHINCS+ | Stateless hash-based signatures. https://sphincs.org/index.html

[22] Kirsch, Z., & Chow, M. (2015). *Quantum computing: The risk to existing encryption methods*. Retrieved from URL: http://www.cs.tufts.edu/comp/116/archive/fall2015/zkirsch.pdf.

[23] Kumar, A., Ottaviani, C., Gill, S. S., & Buyya, R. (2022). *Securing the future internet of things with post-quantum cryptography*. Security and Privacy, 5(2), e200.

[24] Kurose, J. F., & Ross, K. W. (2022, January 1). *Computer Networking: A Top-Down Approach* (8th ed.). Pearson Education.

[25] Lamport, L. *Constructing Digital Signatures from a One-Way Function*; Technical Report, Technical Report CSL-98; SRI International: Palo Alto, CA, USA, 1979.

[26] Loepp, S., & Wootters, W. K. (2006). *Protecting Information: From Classical Error Correction to Quantum Cryptography*. Cambridge University Press.

[27] Lyubashevsky, V., Peikert, C., & Regev, O. (2013, June 25). *On Ideal Lattices and Learning with Errors Over Rings*. Cryptology ePrint Archive. https://eprint.iacr.org/2012/230.pdf

[28] Mavroeidis, V., Vishi, K., Zych, M. D., & Jøsang, A. (2018). *The impact of quantum computing on present cryptography*. arXiv preprint arXiv:1804.00200.

[29] McGrew, D.; Kampanakis, P.; Fluhrer, S.; Gazdag, S.L.; Butin, D.; Buchmann, J. *State management for hash-based signatures*. In Proceedings of the International Conference on Research in Security Standardisation, Gaithersburg, MD, USA, 5–6 December 2016; pp. 244–260.

[30] Merkle, R.C. *A certified digital signature*. In Proceedings of the Conference on the Theory and Application of Cryptology, Houthalen, Belgium, 10–13 April 1989; pp. 218–238.

[31] Paar, C., Pelzl, J., Paar, C., & Pelzl, J. (2010). Introduction to public-key cryptography. *Understanding Cryptography: A Textbook for Students and Practitioners*, 149-171.

[32] Peikert, C. *A decade of lattice cryptography*. Found. Trends Theoretical Computer Science. 2016, 10, 283–424.

[33] Pereira, G.C.; Puodzius, C.; Barreto, P.S. *Shorter hash-based signatures*. J. System Software. 2016, 116, 95–100.

[34] Perlner, R. (2019, May 19). *Code based Crypto in the NIST PQC Standardization Process*. 7th Code-Based Cryptography Workshop - CBC 2019. https://drive.google.com/file/d/1nruEobwdeJbtwouJssbjZCK0WQiBN7rW/view

[35] 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.

[36] Scherer, W. (2019, November 13). *Mathematics of Quantum Computing: An introduction*. Springer.

[37] Shor, P. W. (1994, November). *Algorithms for Quantum Computation: Discrete Logarithms and Factoring*. In Proceedings 35th annual symposium on foundations of computer science (pp. 124-134). Ieee.

[38] Shor, P. W. *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*. SIAM Rev. 1999, 41, 303–332.

[39] *Shor's algorithm*. IBM Quantum. (n.d.). Retrieved from https://quantum-computing.ibm.com/composer/docs/iqx/guide/shors-algorithm

[40] Vazirani, U. (1998). *On the power of quantum computation*. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 356(1743), 1759-1768.

[41] Williams, C. P., & Clearwater, S. H. (1998). *Explorations in Quantum Computing*. Springer.

[42] Xagawa, K., Ito, A., Ueno, R., Takahashi, J., & Homma, N. (2021). *Fault-Injection Attacks against NIST's Post-Quantum Cryptography Round 3 KEM Candidates*. Cryptology ePrint Archive. https://eprint.iacr.org/2021/840.pdf