Lattice-Based Cybersecurity In the Quantum Era:

An Exploration of the Future's Security

By Daniel Yost

May 15, 2020

Abstract - An Exploration of the Future's Security

Few people understand the risks of quantum computers as they relate to data security, and for good reason: little material is available to explain these concepts in a concise, common-sense manner. This paper describes the fundamental laws of quantum computing in an accessible way. In the process, a key vulnerability in traditional encryption is uncovered. Fixing this vulnerability using an advanced form of encryption is explained using simple polynomials and illustrated with comprehensive examples. The result is a practical implementation of a lattice cryptosystem based on open source libraries. The paper concludes with a call to action: to keep data safe from quantum attacks, more research is needed into the field of quantum-resistant cryptosystems.

Introduction

For the average person, the world of quantum mechanics seems beyond comprehension. For instance, how can matter be in a state of existence and nonexistence simultaneously? As the esteemed physicist and Nobel laureate Richard Feynman once put it, “I think I can safely say that nobody understands quantum mechanics” (Feynman). Today, as engineers and physicists begin to design computers based on this mysterious science, complexities continue to mount. When so little is understood about the potential of quantum computers, it is especially difficult to assess this technology for risks. A straightforward explanation of quantum concepts is needed so that researchers can better understand the vulnerabilities quantum computers may cause as well as methodologies to prevent them. This paper will highlight the benefits and drawbacks of entering into this new era.

What Is a Quantum Computer?

In essence, a quantum computer is a device that uses the properties of quantum physics to perform computations not easily done by a traditional, electrical signal-based computer. One way that quantum computers operate is using particles of light, which exhibit the behavior of both a wave and a particle simultaneously. A traditional computer, on the other hand, operates using an electrical signal transmitted through a conductor. This electrical signal can be in one of two states: on, which is typically represented by a 1, or off, represented by a 0. Quantum computers use a similar system. Where traditional computers have bits, quantum computers have qubits. Qubits can exist in an on or off state as well, and they have a special property of being able to exist in a *probability* of on and off states, which is often thought of as being in both states simultaneously. This property is called superposition. Once such a particle is observed or measured, it "collapses" into either an on or off state (Rieffel 1-13).

It may not seem like these particles are useful if they collapse into a normal state once observed. However, to perform calculations traditional computers cannot, quantum computers rely on another quantum principle. "Entanglement" is a somewhat counter-intuitive law of quantum physics which Einstein famously called "spooky action at a distance" (Rieffel 205). Using a special kind of logic gate, two qubits in a quantum computer can become "entangled" in a pair, meaning they are associated regardless of proximity to one another. Once one qubit in the pair is observed to be "on," the other in the pair is known to be "off" (Rieffel 205-206).

The concept of a particle existing between states in a superposition can be difficult to understand. Physicist Erwin Schrödinger, who was initially trying to highlight the nonsensical nature of quantum mechanics, ended up with a simple thought experiment that illustrates this

concept with more clarity. In Schrödinger's hypothetical experiment, a cat is placed in an isolated box where it cannot be observed with a sealed vial of poison. The vial of poison is placed underneath a hammer. That hammer is connected to a geiger counter, so that the hammer will drop and smash the vial (releasing the poison and killing the cat) if the geiger counter reads above a certain level. The geiger counter is then pointed at a bit of semi-radioactive uranium:

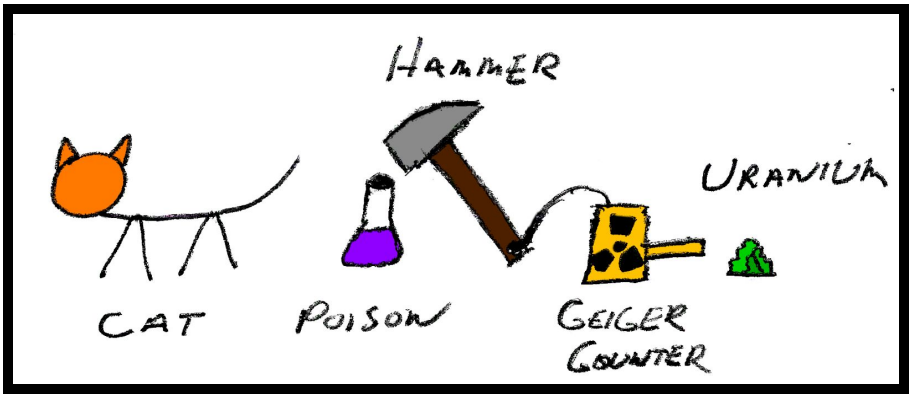


Figure 1.1 - Schrödinger's Cat Experimental Setup

Radioactive materials emit radiation based on the laws of quantum mechanics, meaning it is truly random whether or not the geiger counter will be triggered and drop the hammer, killing the cat (Schrödinger). Say the experiment was left to run for an hour. Without opening the box, is it possible to tell what state the cat is in — is the cat alive or dead?

The cat's state depends on the laws of quantum mechanics. This means until someone opens the box and observes the cat, the cat is in a superposition of life and death. While this does not mean that the cat is literally both alive and dead at the same time, it does mean that the cat has a certain probability of being either dead or alive once the box is opened based on the probability that the geiger counter was triggered. Scientists have even replicated non-felicidal

versions of this experiment in real life (Fein). Just like the cat, qubits are placed into a superposition using a device that leverages the unique properties of quantum mechanics.

Why Are Quantum Computers Useful?

Modern quantum computers still have some issues. Qubits cannot be duplicated, which means redundant error correction methods are ineffective in the quantum world. Even the most advanced quantum computers are error-prone (Rieffel 31). However, it is possible that in the near future more advanced quantum computers may leverage the principles of entanglement and superposition to perform more complex tasks than a traditional computer. Taking a closer look at qubits, one finds qubits in a superposition can be represented as a vector (A, B), where A is the probability of the qubit collapsing into a 0, and B is the probability of the qubit collapsing into a 1. By using these probabilities to store data, a qubit can mathematically represent 2^x more bits than a traditional computer (where x is the number of classical bits). This results in a sort of "exponential speedup" which (in theory) allows quantum computers to solve some complex problems exponentially faster than classical computers (DeCusatis).

P vs NP

The chessboard problem can be used to explain the idea of an "exponential speedup" in more depth. In this problem, a great knight appears before the king. The king, in honor of the knight's victory in a critical battle, promises him anything his heart desires. In response, the knight has a curious request. He asks the king to get him a chessboard. Then, on the first square of the board, place 1 grain of wheat. Next, the knight says, simply double the number of grains

for each square, so that the second square has 2, the third has 4, etc. The king accepts this seemingly simple request. However, he soon realizes the task to be impossible. If the king had a traditional computer, he would find that calculating such a large number of rice grains is exponentially difficult as the number gets larger (DeCusatis). By plotting time taken to calculate versus number of rice grains calculated on a graph, it is clear that time taken shoots up rapidly as the value increases:

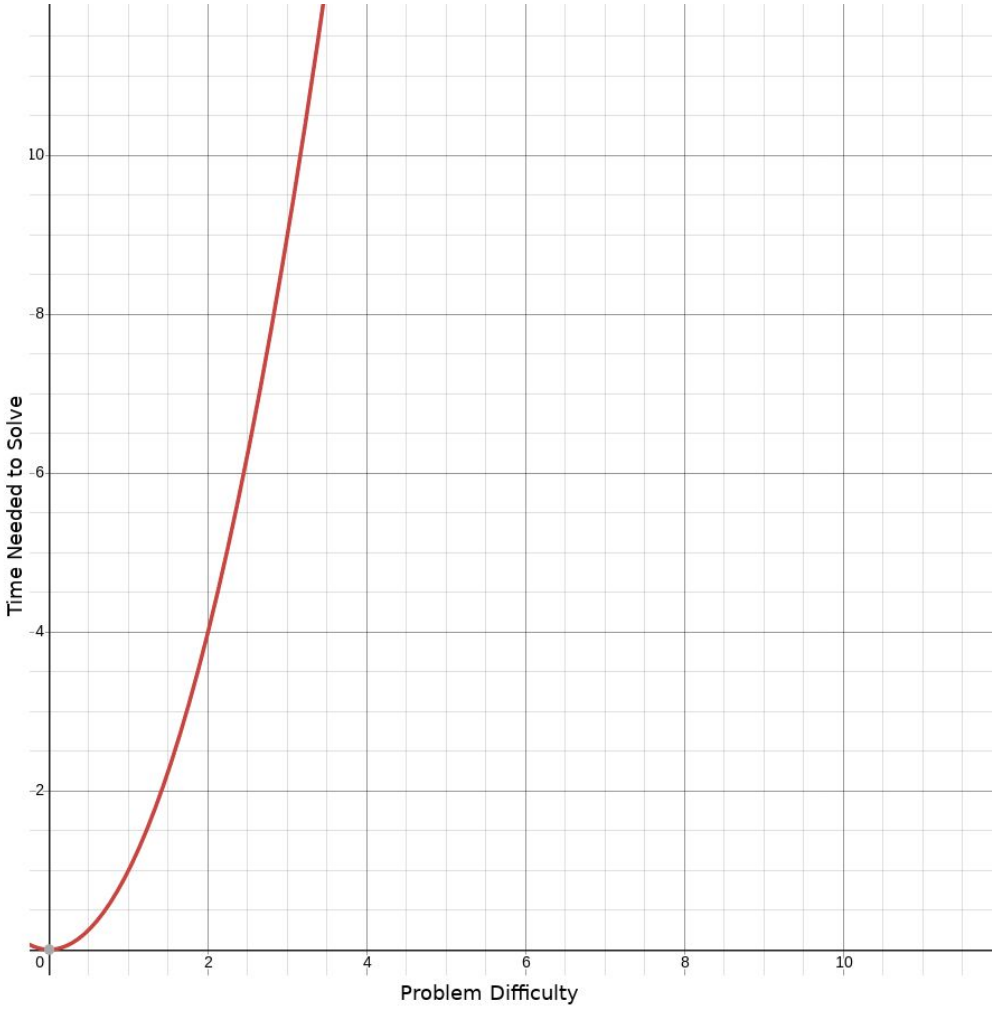


Figure 2.1 - Exponentially increasing graph of rice grains: $y = 2^x$

The number of grains of rice it would take to fill every square is 2^64 - 1, or more than 18 trillion grains. Problems like this are said to have $O(2^n)$ difficulty, because their difficulty to calculate increases exponentially as the size of the number increases. As far as we know, when using a traditional computer, there is no mathematical solution that allows us to calculate these exceedingly large exponents more efficiently. Problems like this are called NP Problems, because they are solvable in Non-Deterministic-Polynomial time (Aaronson 5). There are also P Problems, which are solvable much more quickly relative to NP problems. Looking at the same graph with a P Problem, it could look much more like a straight line, steadily increasing:

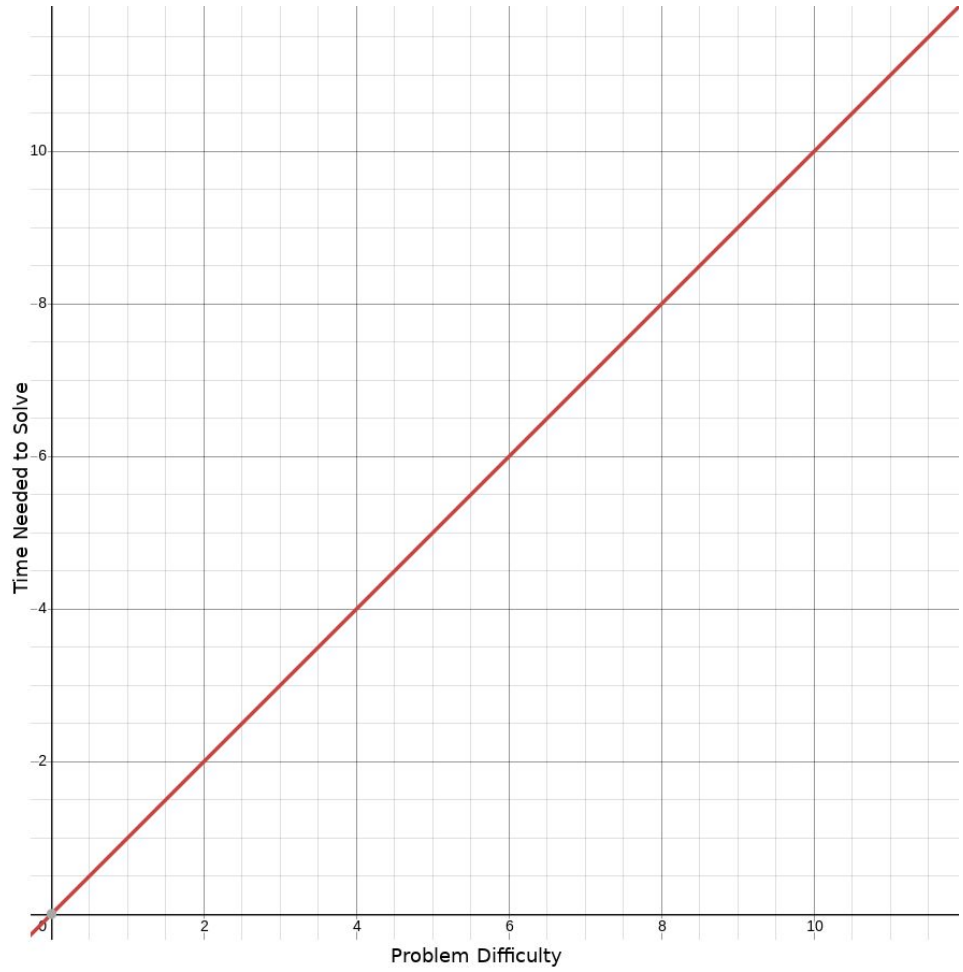


Figure 2.2 - Steadily increasing line: y = x

Problems like this are said to have O(n) difficulty. With a quantum computer, certain NP problems undergo a significant speedup. It is important to note however that most NP problems (especially a special class called NP-complete problems) do not experience this speedup. Quantum computers only affect a few essential problems in a significant way. One such problem is the factoring of large numbers, which is relied upon to encrypt the majority of the world's data (Aaronson 1-4). Thus, it is possible that, with a powerful enough quantum computer, global data security could become compromised. To understand why, one must first understand how modern encryption protects data.

The Basics of Encryption

All forms of encryption work using what is called a trapdoor function. This can be any sort of mathematical operation which is easy to solve in one direction but very difficult to reverse, much like a trapdoor is easy to fall through but hard to climb out of. Most modern encryption methods rely on the difficulty of factoring out two large prime numbers once they have been multiplied together. With a traditional computer, this is an NP problem and a reliable trapdoor function because it takes a really long time to solve as the numbers get larger and larger.

All types of encryption use some kind of trapdoor function. When trying to better understand how encryption works, the most common kind — public/private key encryption — is a good place to start. This encryption relies on each person in the system having a public key, typically a lengthy alphanumeric value fully visible to anyone, a private key, a similar value only visible to the individual, and a message the individual wants to transfer. To transfer an encrypted message to a recipient, the sender uses their message and the recipient's public key in a trapdoor

function to produce an encrypted message. What is critical here is that public/private key encryption relies on a special kind of trapdoor function — one that can only be reversed using the public key's corresponding private key. This encrypted message will appear as garbled nonsense to anyone trying to access it without permission, even the person who sent it. But the recipient can easily use their private key to decrypt the message and read its contents (Miles).

This system also allows users to add verification that their message was indeed sent by them. This relies on an important insight: it is known that the public key can be used to encrypt a message that can only be decrypted with the private key, but if the private key is used to encrypt a message, the inverse is true: that message can only be decrypted with the public key. It might seem pointless to encrypt a message in this way if anyone can decrypt it. However, if you successfully decrypt a message with a public key, you have proven the message came from the sender, since only the sender would have been able to encrypt the message with their private key in the first place. By using multiple layers of encryption, you can add both user verification and guaranteed security to a message (Miles).

When sending a message, a person can use their private key to encrypt the message, then use the other person's public key to encrypt the message a second time. The message recipient then reverses the two layers of encryption by running the message through the trapdoor function with their private key, then repeating the process with the sender's public key. Since the sender's public key will only successfully decrypt the message if the message was encrypted using the sender's private key, the recipient will know the message was indeed sent by the sender (Miles).

Most modern public/private key cryptosystems rely on multiplication operations and the difficulty of factoring large numbers. However, encryption methods that use factoring are

particularly vulnerable to quantum attacks. Using a solution called Shor's algorithm, factoring two numbers that have been multiplied together becomes particularly easy with a quantum computer (Bernstein 1). This compromises the security of the trapdoor function. The safety of these algorithms lies in their near impossibility to crack by a traditional computer: hacking a typical password would take centuries of brute-force work. With a powerful quantum computer, a hacker could find the same password in a matter of moments (Bernstein 1-2).

What Is Lattice Encryption?

There are trapdoor functions quantum computers cannot crack, however. One such solution is to use a series of lattice operations instead of the traditional multiplication method. A lattice is a multidimensional mathematical group formed from a basis. The basis can be any set of linearly independent vectors whose components are whole integers. For example, a two-dimensional basis could be the vectors (0,2) and (1,0). A lattice is essentially a grid of points formed by creating a linear combination of the basis vectors. Continuing this example, a 2D lattice formed from the given vectors would look like this:

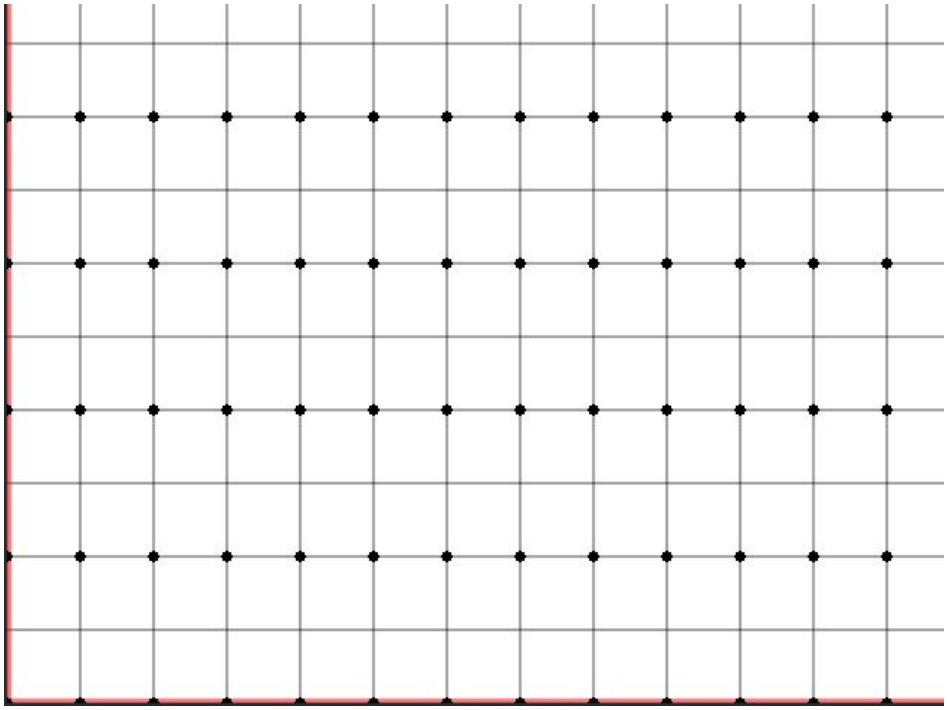


Figure 3.1 - A 2D lattice with basis vectors (0,2) and (1,0)

The mathematical process of forming a lattice is similar to the trapdoor functions used in other encryption applications. It is quite easy to form a lattice given a basis: just multiply the vectors by whole integers and take all their linear combinations as points. But a number of problems are very difficult to solve when given an untenable basis for a lattice. For example, finding the closest vector to any given point in the lattice becomes exponentially more difficult as the number of dimensions in the basis vectors increases. This is called the closest vector problem (CVP). While it may seem simple to solve when looking at the lattice in Figure 3.1, adding dimensions greatly increases the difficulty of finding an appropriate solution. While a fast solution may be possible, CVP has consistently been shown to be NP-hard. This means it is a potential candidate to build a public/private key cryptosystem around (Alwen).

GGH Cryptosystems

There are many different kinds of cryptosystems that rely on the CVP to encrypt and decrypt a message. The simplest of these to understand is the Goldreich-Goldwasser-Halevi (GGH) encryption scheme (Suzuki). GGH relies on the most intuitive solution to CVP, often referred to as Babai's algorithm (Suzuki). For example, if we are given a 3-dimensional lattice with the basis vectors v_1, v_2, and v_3, we can find any point in the lattice p using a linear combination of those vectors, where each term in the linear combination is multiplied by an integer coefficient (a_1, a_2, a_3).

$$p = a_1 * v_1 + a_2 * v_2 + a_3 * v_3$$

Keep in mind all values v are vectors here, so the vector values can be broken up into a system of equations that can be represented like this:

$$x_p = a_1 * x_1 + a_2 * x_2 + a_3 * x_3$$

$$y_p = a_1 * y_1 + a_2 * y_2 + a_3 * y_3$$

$$z_p = a_1 * z_1 + a_2 * z_2 + a_3 * z_3$$

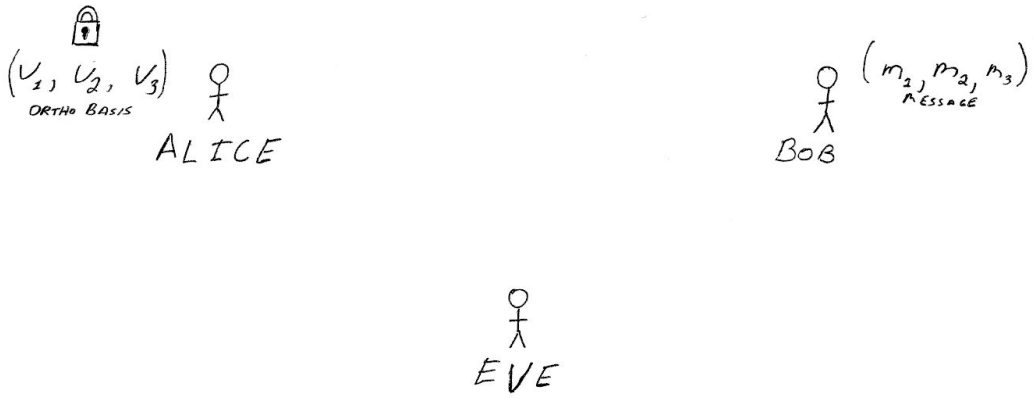
By solving this system of equations, one can derive the coefficients a. To find the closest lattice point p to a given point q, simply solve a similar linear combination for q:

$$q = b_1 * v_1 + b_2 * v_2 + b_3 * v_3$$

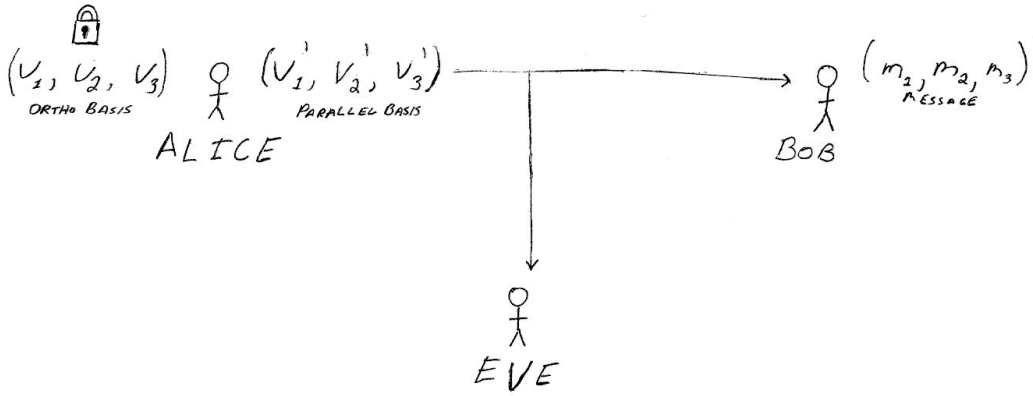
Where all values b can be any real number, not just integers. Then, by rounding all values b to the nearest whole integer and solving for p, one can derive a solution to the closest vector problem. Babai's algorithm can only approximate the closest lattice point if the basis

vectors are “reasonably orthogonal,” or roughly perpendicular to one another. This is the key insight that enables the construction of a cryptosystem.

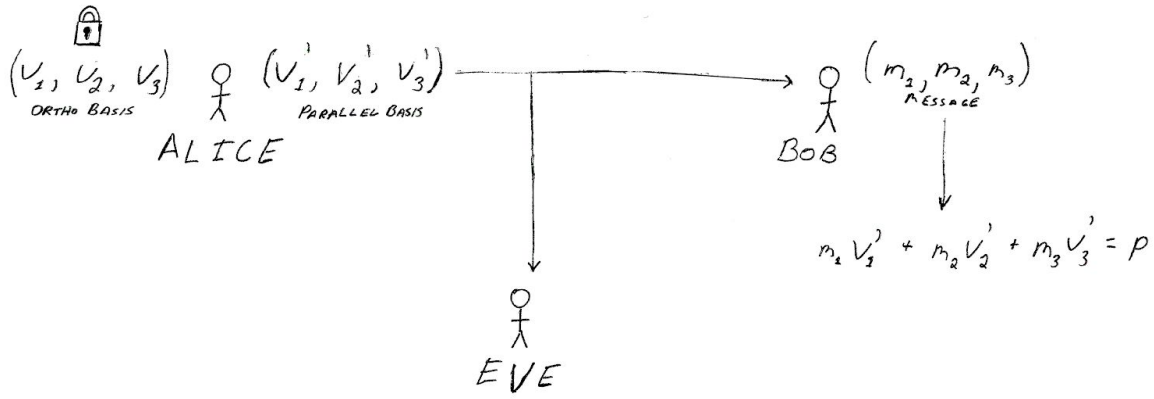
If one can create a lattice with at least two different bases: one nearly orthogonal and one nearly parallel, those bases can be used as private and public keys, respectively. To understand this better, take the classic cryptography example of Alice, Bob, and Eve. Bob wants to send a secret message to Alice, but Eve is trying to eavesdrop. To encrypt the message, Alice first decides on a basis of reasonably orthogonal vectors (v_1, v_2, v_3) and makes this her private key:



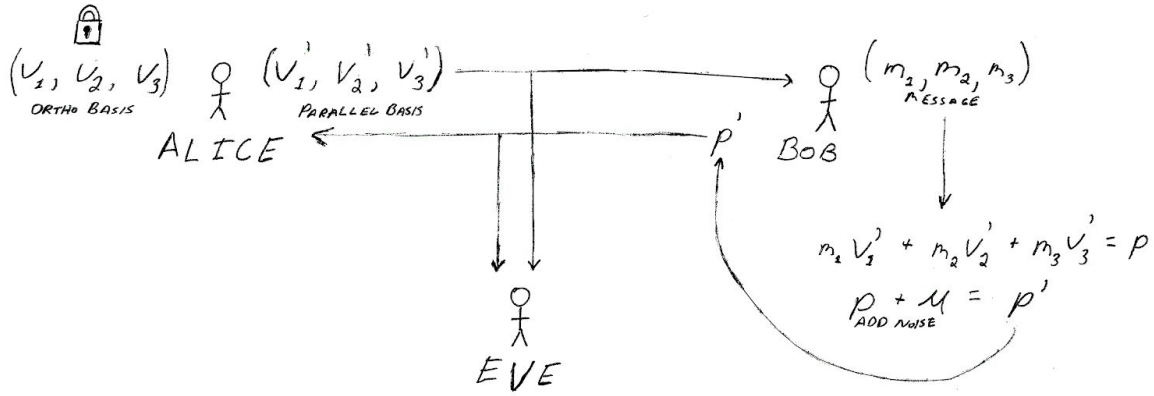
She then finds a basis of roughly parallel vectors from the same lattice (v'_1 v'_2, v'_3) and makes this her public key, visible to Eve and Bob:



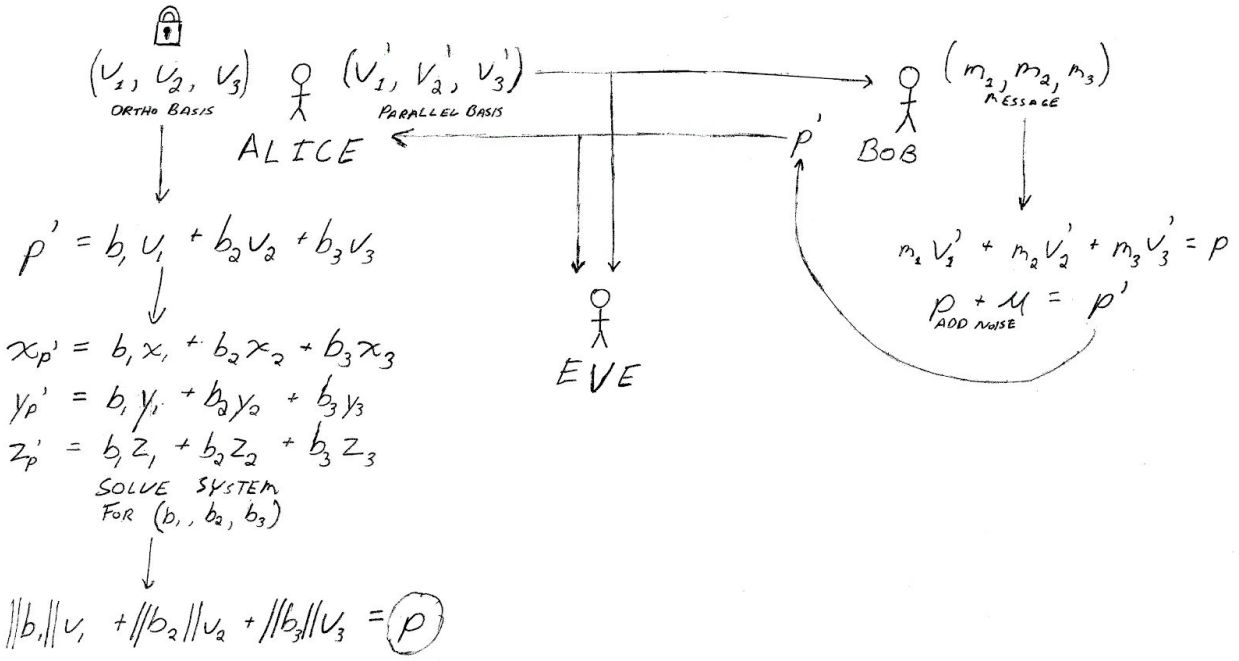
To send Alice a message, Bob uses the binary values of his message (m_1, m_2, m_3) as the coefficients in a linear combination of Alice's public basis, which results in a lattice point p:



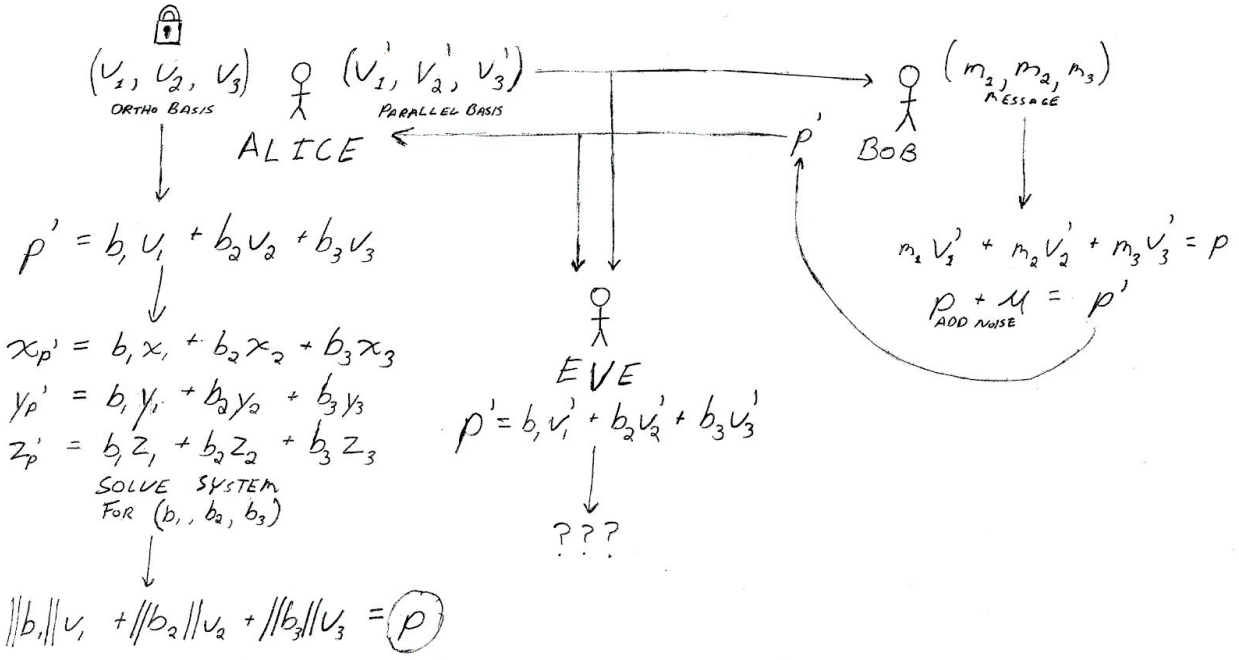
Then, Bob adds a noise vector μ to his lattice point, shifting it slightly from its original position in the lattice. He then sends his shifted message p' to Alice (it is also intercepted by Eve):



Because the point is close to a point in the lattice which contains the message, but not exactly on it, Alice needs to solve the closest vector problem using Babai's algorithm to find the message. Most of the time this will be really easy for her because she can use her private basis of near-orthogonal vectors:



For Eve, however, who only has access to Alice's public, near-parallel basis, will most likely get an incorrect answer (when using only Babai's algorithm, at least):



Thus, Alice successfully receives a secured message from Bob without Eve being able to decipher it. Normally, this would be done with a very large number of dimensions to increase the difficulty of the problem, not just three.

The Problem with GGH

Sadly, the simplest solution is not always the best. GGH is quite useful when first learning how to create a lattice encryption system, because it serves as a relatively understandable framework for a more sophisticated system. Unfortunately, it has a key flaw: there are many algorithms that Eve can use to find a reasonably orthogonal basis when given a nearly parallel basis. This would make it easy for her to solve the CVP and break the cryptosystem's security.

One of the most effective known solutions to the CVP in this particular situation is the Lenstra–Lenstra–Lovász (LLL) lattice basis reduction algorithm (Suzuki). Using LLL, Eve can take Alice's public set of basis vectors and reduce it into a set of nearly orthogonal vectors. This is accomplished through a repetitive process that relies on a type of vector reduction called a Gram-Schmidt reduction (Suzuki). To start, Eve performs the Gram-Schmidt reduction on the first two vectors in the basis and produces a new potential basis vector. Next, an inequality called the Lovász condition is checked to determine whether or not this new basis vector is an adequate replacement for the second vector. The Lovász condition essentially tests whether or not the new basis vector is sufficiently small and orthogonal. If it is not, the two original basis vectors are swapped and the whole process starts over. If it is, the new basis vector replaces the second vector. Then, the Gram-Schmidt reduction process is repeated with the second and third

vectors. This algorithm repeats until the Lovász condition is satisfied for every vector in the basis. Once every vector satisfies the Lovász condition, a nearly parallel set of basis vectors has been successfully converted into a nearly orthogonal set, and the encryption has been broken (Suzuki).

NTRU Cryptosystems

Fortunately, cryptosystems employing these same lattice concepts in a more complex manner have proven resistant to LLL basis reduction. The frontrunner is the NTRU Cryptosystem. The NTRU cryptosystem is not just quantum-resistant: its proponents claim that in many situations it can also be faster than traditional encryption methods (Etzel). NTRU operates on the same basic principles as GGH, only with a few extra layers of security to combat the LLL basis reduction method.

As previously discussed, the polynomial form can be used to determine any point in a lattice. Most of the calculations in NTRU rely on a method of multiplying polynomials together called a cyclic convolutional product, which is sometimes represented as a circled asterisk ⊛. This is essentially a special form of traditional polynomial multiplication that takes any exponent in the polynomial greater than *N*-1 and “wraps" it back around to 0. So if two terms in the polynomial are multiplied together and produce x^N, the value N becomes zero and the expression evaluates to $x^0 = 1$. This continues so that x^{N+1} evaluates to x^1, x^{N+2} evaluates to x^2, etc. It’s important that these terms wrap around to 0 because lattices in this context are of a fixed dimension, meaning a polynomial used to represent a lattice must have no exponent in it greater

than or equal to the dimension N. Specifically, the cyclic convolutional product can be represented as follows:

$$f \circledast g = c_0 + c_1x^1 + c_2x^2 + \ldots + c_{N-1}x^{N-1}$$

$$\text{where each term } c_k = f_0g_k + f_1g_{k-1} + \ldots + f_kg_0 + f_{k+1}g_{N-1} + f_{k+2}g_{N-2} + \ldots + f_{N-1}g_{k+1}$$

In a sense, the cyclic convolutional product simply multiplies every term in *f* by every other term in *g*. In an equation, it has the same position in the order of operations as regular multiplication, meaning a simple convolutional product takes precedence over addition and subtraction. This special product is the backbone of the NTRU cryptosystem (Stoev).

The example of Alice, Bob, and Eve can be used to explain NTRU. To set up the cryptosystem, Alice and Bob first define three publicly available variables. First they define *N*, which represents the maximum degree of all polynomials used in the cryptosystem. Then they create two variables *p* and *q.* These numbers can be pretty much anything as long as *q* is significantly larger than *p,* and the greatest common denominator between *p* and *q* is one. All equations in NTRU are taken either modulo *q* or modulo *p*, meaning the larger value *q* creates a sort of "upper bound" on any possible point in a lattice — none of the points will exceed the value *q*.

Next, Alice creates two sets of polynomials of degree N-1, L_f and L_g. These will be used to generate her public and private keys. In the meantime, Bob creates two sets L_Φ and L_m. L_m contains polynomials whose coefficients are the binary message values that Bob wants to send to Alice. L_Φ is a little harder to explain, but in essence it contains "noise" polynomials that are similar to the noise vectors used in GGH.

To begin the encryption process, Alice chooses two polynomials at random from L_g and calls them *f* and *g*. The polynomial *f* will be her private key. Then, using the inverse-calculating Euclidean algorithm, she computes the inverse of *f* when taken modulo *q* and calls this new polynomial F_q. She also generates a similar polynomial F_p, which is the inverse of *f* when taken modulo *p*. Adhering to the properties of modulus, all the coefficients in the inverse F_q must be less than *q*, and all the coefficients in the inverse F_p bust be less than *p*. If these inverses do not exist for *f*, Alice must pick a new polynomial for *f* and try again. Next, Alice generates her public key *h* by calculating the following:

$$h = F_q \circledast g \ (mod\ q)$$

For now, Alice is done with her computations. To send a message to Alice, Bob first picks two polynomials from his generated sets. From L_m he selects his message *m*, and from L_Φ he selects his noise polynomial *Φ*. Using Alice's public key *h,* he generates the encrypted message *e*:

$$e = p\ (\Phi \circledast h) + m\ (mod\ q)$$

He then sends *e* to Alice. To figure out how to decrypt *e*, Alice needs to do a bit of algebra. She essentially reverses all the math that has been done thus far, expanding the equation until she is left with only the initial variables:

$$e = e$$

$$e = p\ (\Phi \circledast h) + m\ (mod\ q)$$

$$e = p\ (\Phi \circledast F_q \circledast g) + m\ (mod\ q)$$

Now she performs a few mathematical operations to derive *m.* First she multiplies the whole equation by her private key *f* and distributes it. Since *f* and F_q are inverses, they cancel each other out:

$$e \circledast f = [p\ (\Phi \circledast F_q \circledast g) + m\ (mod\ q)] \circledast f$$

$$e \circledast f = p\ (\Phi \circledast F_q \circledast g \circledast f) + m \circledast f\ (mod\ q)$$

$$e \circledast f = p\ (\Phi \circledast g) + m \circledast f\ (mod\ q)$$

Next, she needs to cancel out the term $p\ (\Phi \circledast g)$. Since the entire term is multiplied by p, this can be done by taking the entire equation modulo p. Since $p\ (mod\ p) = 0$, the entire term can be removed, and since $p < q$, *mod q* can also be removed:

$$e \circledast f\ (mod\ p) = [p\ (\Phi \circledast g) + m \circledast f\ (mod\ q)]\ (mod\ p)$$

$$e \circledast f\ (mod\ p) = m \circledast f\ (mod\ p)$$

Finally, all Alice has to do is multiply the entire equation by the value she stored at the beginning, F_p. This cancels out $f\ (mod\ p)$, resulting in the decrypted message *m:*

$$F_p\ [e \circledast f\ (mod\ p)] = [m \circledast f\ (mod\ p)]\ F_p$$

$$F_p\ [e \circledast f\ (mod\ p)] = m$$

Therefore, all Alice has to do to decrypt *e* and find *m* is multiply *e* by her private key *f (mod p),* then multiply by the inverse F_p (Stoev).

It is important to note that NTRU has a minor vulnerability. Because the algorithm relies on Alice solving the CVP to find Bob's message, it is possible that Alice could find the wrong point in the lattice and come up with an incorrect answer. However, this case is extremely

unlikely for sufficiently large q values, so the consequences of this vulnerability can be minimized (Etzel).

Released in tandem with this paper is a program called LatLock (Yost) which utilizes a publicly available Java implementation of the open-source NTRUEncrypt library (Etzel). This program serves as a sort of lockbox that can encrypt and decrypt a secret string of text, keeping it safe from a quantum attack. While LatLock is not fully vetted to operate in an actual security system, it serves as a prime example of the potential of lattice encryption (Yost).

Conclusion

Quantum computing does not have to be an alien science. The more people who understand its benefits and drawbacks, the better prepared humanity will be for its eventuality. Quantum computers rely on the principles of superposition and entanglement in order to perform tasks that traditional computers cannot. One of these tasks is the speedy factoring of large numbers using Shor's algorithm. Most modern encryption methods rely on the difficulty of factoring these large numbers, so the world's data security could become compromised. A solution to this problem can be found in lattice-based public/private key cryptosystems. First, GGH was explored. It provides a straightforward framework for a lattice cryptosystem, but has a number of problems that keep it from being practical. This framework was used to explain a feasible solution to the problem: a cryptosystem that could be usable in real-world applications called NTRU. To keep the world's data secure, it is essential that lattice encryption and other quantum-resistant encryption methods be further researched and better understood. Perhaps it is possible that one day an encryption system like NTRUEncrypt will become the world standard.

Glossary of Cryptography Terms

Real Coordinate Space - R^n where n is the number of dimensions. Allows n variables to be related to each other as a single point in a dimensional space.

Basis - a subset of the set of elements that a binary operation can be used on to derive all elements of the set. A linearly independent spanning set.

Linear Independence - A set of vectors where none of the vectors can be defined as linear combinations of the others.

Linear Combination - an expression constructed from a set of terms found by multiplying each term by a constant coefficient and adding them together. Example: a linear combination of x and y is 5x+2y.

Norm - a function that assigns a strictly positive length/size to a vector. Example: the Euclidean norm, $a^2 + b^2 = c^2$.

Lattice - for any basis of the real coordinate space R^n, the subgroup of all linear combinations with integer coefficients of the basis vectors forms a lattice.

Closest Vector Problem (CVP)- Find the vector in the lattice closest to a random vector for any given lattice.

Trapdoor Function - A mathematical method which is easy to calculate in one direction, but hard to reverse. For example, it's easy to multiply two numbers together, but hard to extract the original numbers from the product.

Orthogonal - Perpendicular. When two vectors are at 90-degree angles with respect to one another.

Cryptosystem - Short for cryptography system. A framework of algorithms allowing users to send secure messages to one another.

Babai's Algorithm - the most intuitive solution to the closest vector problem. This algorithm is an essential part of the GGH cryptosystem.

GGH Cryptosystem - A simple public key cryptosystem that relies on the hardness of the closest vector problem to encrypt messages. It is known to be vulnerable.

NTRU Cryptosystem - An advanced public key cryptosystem that utilizes the CVP. So far, no vulnerabilities have been found in it.

Cyclic Convolutional Product - A method of combining two polynomials together in a ring that multiplies every term in one polynomial by every term in the other. Any exponent greater than or equal to the dimension of the polynomial N “loops back” to 1.

Mod/Modulus/Modulo - A mathematical operation which produces the remainder of one number divided by the other. For example, 11 mod 5 = 1. It can be used to put an “upper bound” on a set of number values.

Works Cited

Aaronson, Scott. “The Limits of Quantum Computers.” *Scientific American,* 2008, pp. 62-69.

Alwen, Joël. “What is Lattice-based cryptography & why should you care.” *Medium,* Wickr, 15 June 2018, web. https://medium.com/cryptoblog/what-is-lattice-based-cryptography-why-should-you-care-dbf9957ab717

Bernstein, Daniel J. and Tanja Lange. "Post-Quantum Cryptography." *Nature*, Vol. 549, No. 7671, 2017, pp. 188-194.

DeCusatis, Casimer, "Introduction to Quantum Computer Algorithms and Programming", *Proc. NSF Enterprise Computing Conference,* Poughkeepsie, NY, 6 June, 2020, web.

Etzel, Mark, and William Whyte. “NTRU Crypto - Project Overview.” *Github,* 11 May 2018. https://github.com/NTRUOpenSourceProject/ntru-crypto

Fein, Yaakov Y., Philipp Geyer, Patrick Zwick, Filip Kiałka, Sebastian Pedalino, Marcel Mayor, Stefan Gerlich, and Markus Arndt. “Quantum superposition of molecules beyond 25 kDa.” *Nature Physics,* vol 15, December 2019, pp. 1242-1245.

Feynman, Richard. *The Character of Physical Law.* Cornell University, 1965.

Miles, Robert. “Public Key Cryptography - Computerphile.” Youtube, July 22 2014, web. https://www.youtube.com/watch?v=GSIDS_lvRv4

Rieffel, Eleanor, Wolfgang Polak, William Gropp, and Ewing Lusk. *Quantum Computing: A Gentle Introduction.* MIT Press, 4 March 2011, pp. 9-205.

Schrödinger, E. Die gegenwärtige Situation in der Quantenmechanik. *Naturwissenschaften* 23, 1935, pp. 807–812. https://doi.org/10.1007/BF01491891

Stoev, Vihren. “The Essence of NTRU: Key Generation, Encryption, Decryption.” *Medium,* 21 August 2019. https://medium.com/tixlcurrency/the-essence-of-ntru-key-generation-encryption-decryption-7c0540ef8441

Suzuki, Jeff. “Cryptography.” Youtube, 27 April 2017, web. https://www.youtube.com/playlist?list=PLKXdxQAT3tCssgaWOy5vKXAR4WTPpRVYK

Yost, Daniel. “LatLock” *Github*, 15 May 2020. https://github.com/danstuff/LatLock