ST. PÖLTEN UNIVERSITY
OF APPLIED SCIENCES

Informatik & Security

fh st.pölten

Post quantum cryptography, analysis of modern encryption

Deep dive into quantum algorithms and PQC families

Master thesis

for attainment of the academic degree of

Master of Science in Engineering (MSc)

submitted by

Dobre Radu-Cristian

52304914

in the

University Course Cyber Security and Resilience at St. Pölten University of Applied Sciences

Supervision

Advisor: FH-Prof. Dipl.-Ing. DR. Peter Kieseberg

Assistance: Dipl.-Ing. Paul Lackner, MSc, MA, BSc

Declaration of Honour

First name, Surname: Dobre Radu-Cristian

Matriculation number: 52304914

Title of the thesis: Post quantum cryptography, analysis of modern encryption

I hereby declare that

- I have written the work at hand on my own without help from others and I have used no other resources and tools than the ones acknowledged
- I have complied with the Standards of good scientific practice in accordance with the St. Pölten UAS' Guidelines for Scientific Work when writing this work.
- I have neither published nor submitted the work at hand to another higher education institution for assessment or in any other form as examination work.

Regarding the use of generative artificial intelligence tools such as chatbots, image generators, programming applications, paraphrasing and translation tools, I declare that

☒ no generative artificial intelligence tools were used in the course of this work.

☐ I have used generative artificial intelligence tools to proof-read this work.

☐ I have used generative artificial intelligence tools to create parts of the content of this work. I certify that I have cited the original source of any generated content. The generative artificial intelligence tools that I used are acknowledged at the respective positions in the text.

Having read and understood the St. Pölten UAS' Guidelines for Scientific Work, I am aware of the consequences of a dishonest declaration.

Abstract

In present, there are many options to encrypt data between two endpoints: RSA, AES, 3DES are some examples, but using quantum computers it has been proven the possibility of breaking most of these asymmetric encryptions. Quantum computers will soon be available for public use and can even be rented online. The demand will be high for their capabilities. Now it seems impossible to own a quantum computer in a home, but as history proved, even computers and smartphones were thought impossible until they were widely available to the public, and they became cheaper and smaller with each iteration. There are a vast majority of researches comparing the differences between existing encryptions that are used and how they will fall with the rise of quantum computers. The focus should also be on the future of cryptography and algorithms that are capable to withstand quantum computers computational power, more specifically the ones that are developed as of writing this thesis. Different algorithms that are developed and tested nowadays to future proof our encryption systems and also analyzing the proposed and now chosen algorithms by NIST. This thesis proposes to analyze the differences and capabilities of popular algorithms that show promise today as well as the crypto families they are based on. Explanation of why today's encryption like RSA is not good enough and how AES is still able to withstand the coming of quantum computers. Towards the end of the thesis Lattice space family is presented and its capabilities that made it so popular among the candidates in NIST rounds.

Contents

1. Introduction

Cryptography is an essential part in today's world. Attacks are growing everyday and more companies are affected by suboptimal security configuration inside the network. Even if an attacker has access to the data of a company, he should not be able to gather information, that is why encryption is so important. Encryption does not assure that our data will be forever protected but at least will take a lot of time(even decades) to decrypt. The stronger the encryption, the stronger the protection against perpetrators(unless the key to decrypt the data is not safe).

Quantum computers will prove to be a problem for the future of everyone using IT devices. The first quantum computer was developed in 1998 with only two-qubit, in 2024 "quantum computing company Quantinuum announced that their new 56-qubit H2-1 computer has broken a world record in quantum supremacy, topping the performance of benchmarking set by Google's Sycamore machine by 100-fold" [1]. At the end of 2024, Google announce their new quantum chip which claims unlimited power "Second, Willow performed a standard benchmark computation in under five minutes that would take one of today's fastest supercomputers 10 septillion (that is, 1025) years - a number that vastly exceeds the age of the Universe." [2]. The qubits of a quantum computer are very important, they are the basis of a quantum computer. They are usually a 1 or 0 like a bit, but they have a special state in which, according to quantum mechanics, can arbitrarily be a coherent superposition of all computable states simultaneously, so they can hold even more information and probabilities.

Using Shor's algorithm or Grover's algorithm quantum computers are able to break encryptions based on Discrete Logarithm Problem(DLP) like RSA, and AES respectively. That is why NIST (National Institute of Standards and Technology) announced their competition, like they did before and AES got chosen, for quantum resistant algorithms. As of writing this thesis NIST PQC (Post-Quantum Cryptography) round 4 submissions are the latest. This is an important step for the future of humanity because everything is connected to internet nowadays, probably in the future too and even more expanded. Encryption is the bases of today's confidentiality. In the NIST rounds a very interesting fact is that in each round there was a Lattice based encryption algorithm submitted and in all rounds it is a predominant option for PQC algorithms.

After analyzing everything we talked about so far, the following topics raised my curiosity:

- **Quantum computers**: How can they decrypt only some algorithms and not all? Are they not supposed to be better in every sense to normal computers? Why are they not working now?
- **Shor's Algorithm**: Why is it so effective against Diffie-Hellman and why is it not used nowadays? How can quantum computers improve it?
- **Post quantum Algorithms**: What is a Lattice-space and why most of the algorithms submitted use it? Why are lattice based algorithms resistant to quantum and classical computers? What other crypto families for PQC are there and how they compare to Lattice Space Algorithm?

These questions are the questions proposed for the thesis and why this curiosity aroused the pursue of future proof algorithms. The future depends a lot on these algorithms that are being developed right now, like AES was in 2000. Not only they have to resist to quantum computing, they should also resist to traditional computing.

Throughout this paper traditional or classical algorithms, computing, etc will refer to everything based before quantum computing widespread. Everything now is based on two bits, 0 and 1, and DLP algorithms work really great with the systems we invented so far because they adhere to standard rules, and quantum computing are changing these rules.

As a result, the main question of this thesis will be: "Why are lattice based algorithms widely used for post quantum algorithms and how do they compare to other schemas?". This question implies also the analysis of other proposed PQC families and a dissection of lattice-space algorithms. Note: As of writing this thesis, on August 13th, NIST announced the finalists for PQC[3] and two out of three standards are based on lattice space and learning with errors. This proves the importance of Lattice space and why it was used as the main crypto-family between the contestants and rounds, as well as its resistance against quantum and traditional computing.

1.1. Contribution

My contribution is based on the analysis of the different families for post quantum cryptography as well as explaining the basis and general rules on how they work. I will be taking my own approach to understanding Lattice space and generating them visually as well as trying to create improvements in this area on how it can be decrypted and computed. This thesis can help future engineers have a high level understanding of current PQC algorithms and how lattice based algorithms could be attacked and recreated. Also I will talk about the Shor's algorithm and Grover's algorithm and will discover why they do not affect Lattice spaces.

1.2. Thesis Outline

This section serves as a way to help the reader create a general idea of the content in this thesis It will provide a short description of each chapter and allow the reader to focus more specifically on any chapter of interest.

The overall structure of this document is as follows:

- **Introduction**: As discussed earlier, the introduction (referenced as chapter 1) provides an overview of the topic, challenges, motivation, and the scope of the research.
- **Background**: This section (referenced as chapter 2) covers necessary prerequisites and fundamental knowledge relevant to the subject.
- **Related Work**: The related work section (referenced as chapter 3) presents existing research and literature that pertains to our study.
- **Methodology**: In this part (referenced as chapter 4), we delve into the methodology employed in our research.
- **Quantum Computing Decryption**: crefsec:QuantumComputingDecryption presents the challenges that comes with quantum computers and the threat that cybersecurity faces. It covers traditional cryptography such as DLP and Symmetric key algorithm as well as the threats that they encounter because of different algorithms such as Shor's and Grover's algorithm.
- **PQC Families**: The PQC Families are presented in this chapter chapter 6 and the hope that they bring to cryptography. This chapter presents the crypto families that are used in NIST rounds and the advantages as well as disadvantages, and how well they are to securing the future.
- **Lattice Space**: In this chapter chapter 7 the focus is on the lattice-based cryptography and its potential for greatness. It will describe how it works, why is it so popular and the mathematical functions that is based upon.
- **Results**: In this section chapter 8, the results of the experiments are documented and presented.
- **Conclusion**: Finally, we draw conclusions and summarize the key findings and contributions of this thesis in chapter 10.

2. Prerequisites

This chapter serves as an essential foundation for comprehending the subsequent content of this work. It is advised that the reader acknowledges this topics because they will be discussed in detail in this thesis and will offer a better understanding of the topics presented.

In the first part of the thesis it will be discussed Discrete Logarithm Problem (DLP) and the cryptographic algorithms using it (RSA) as well as Symmetric key algorithms (AES). This will help the reader understand better the next topic about quantum computing and how they can break this type of algorithms. Python will be used for representation of the lattice space as well as other small code snippets for aiding in understanding the contents of this thesis. It is important to know what a quantum computers is. A short description of a quantum computer, it is a computer that is harvesting quantum mechanics properties such as superposition of particles. The qubit is the basic unit of computation for quantum computers and serves the same function as the bit, 1 and 0, in traditional computing. However, unlike a traditional bit, a qubit can exist in a superposition of its two "basis" states, which loosely means that it is in both states simultaneously.

The mathematical problems that these encryptions are based on have to be NP-hard. NP stands for non polynomial time to solve a problem. NP-hard is a type of problem that is hard to solve but easy to verify by having the answer, and they cannot be solved in polynomial time, meaning they would take a very long time to solve without the answer. This is a must for a strong encryption, because it confers the security that without the answer, there is almost no possible way to find it. All these algorithms have to abide by security reductions, which is the correlation of a cryptographic algorithm and a known hard mathematical problem, and are proofs to the difficulty of breaking the algorithm.

3. Related Work

For the purpose of this thesis multiple research papers and works published were analyzed on the vast application of quantum computing and their capabilities in the future. The research ranged from basic cryptography to the algorithms used in NIST rounds to the effect it will have on blockchains. Researchers are trying to find better applications for these algorithms and improve them. A big part of this thesis is trying to provide a new view on Lattice Space and the families that are used for post quantum cryptography.

- **A Decade of Lattice Cryptography**[4]: Chris Peikert analyzed in his work the evolution of lattice-based cryptography over the span of 10 years and focuses more on the new developments found in this practice, more importantly short integer solution (SIS) and learning with errors (LWE) problems (and their more efficient ring-based variants). He purposed to research this area because of lattice-based cryptography apparent resistance to quantum attacks in comparison to most number-theoretic cryptography.
- **Lattice Cryptography for the Internet**[5]: In this conference paper, Peikert, C. and Mosca, M. (eds) talked about the possibility of using lattice based cryptography for further use in the internet area, more specifically for key transport, encryption, and authenticated key exchange. One of the main goals of the paper was to invent low-bandwidth reconciliation technique to agree on the exact key, based on approximate data for the key. The cipher-text length is reduced by this technique compared to prior encryption schemes.
- **A Comparison of Security and its Performance for Key Agreements in Post-Quantum Cryptography**[6]: F. Borges, P. R. Reis and D. Pereira analyzed todays problem of cryptography. The main security problems are proven by Integer Factorization Problem and the Discrete Logarithm Problem, but they can easily be solved using Shor's quantum algorithm. So they started to analyze the NIST second-round mainly isogeny crypto-systems based on super-singular elliptic curves, error correction code-based encryption system, and lattice-based ring learning with errors by performing key agreements protocols and comparing them to the traditional ways.
- **Post-quantum generations of public-key cryptography**[7]: Gan, L and Yokubov, B are examining

the various algorithms of digital signatures of post-quantum generations of public-key cryptography and their performances, because Blockchain and other Distributed Ledger Technologies are liable to Grover's and Shor's algorithms. The research presented includes application domains where post-quantum blockchain may be used.

- **Faster Isogenies for Post-quantum Cryptography: SIKE**[8]: SIKE is another candidate for post-quantum cryptography purposed in NIST third round and the only isogeny-based implementation, but it is the slowest out of all candidates by computing series of isogenous mappings between curves, that accounts for 80 percent of its latency. In Elkhatib, R., Koziel, B., Azarderakhsh, R. paper, their new method obtains 10 precent and 5 precent speedups and reduced stack space based on modifications in the open source libraries targeting x86, ARM64, and ARM32 platforms. The family that SIKE is based on has already been broken and described in the next paper.
- **An Efficient Key Recovery Attack on SIDH**[9]: Castryck, W., Decru, T. amazing work discovered a way to attack effectively the Supersingular Isogeny Diffie-Hellman protocol using an efficient key recovery attack using Kani's reducibility criterion for isogenies from products of elliptic curves. This attack relies on the torsion point images that Alice and Bob exchange during the protocol. The attack works if one of the parties uses 2-isogenies and the starting curve comes equipped with a non-scalar endomorphism of very small degree. SIKE is based on SIDH and it advanced to the fourth round of NIST's standardization for post-quantum cryptography. The implementation successfully breaks SIKE in about ten minutes on a single core.
- **Emerging Networking EXperiments and Technologies**[10]: Dimitrios Sikeridis , Panos Kampanakis, Michael Devetsikiotis paper acknowledged the threat of quantum computers and purposed to analyze the performance of the NIST proposed quantum-resistant key encapsulation and authentication schemes. They are evaluating both post-quantum key exchange and authentication are integrated into TLS and SSH and discovered a handshake latency ranges between 1-300 precent for TLS and 0.5-50 precent for SSH depending on the post-quantum algorithms used. Also examining the initial TCP window size affects post-quantum TLS and SSH performance can result in a slowdown by 50 precent and analyzes alternatives to encourage early adoption with minimum overhead.
- **Vulnerabilities in traditional implementations of cryptographic algorithms**[11]: Ritik Bavdekar, Eashan Jayant Chopde, Ashutosh Bhatia, Kamlesh Tiwari, Sandeep Joshua Daniel, Atul studied post-quantum computers and traditional algorithms vulnerabilities, observing that most of the symmetric and asymmetric cryptographic algorithms are vulnerable to quantum algorithms like Grover's search algorithm that gives a square root time boost for the searching of the key in symmetric schemes and

for asymmetric algorithms Shor's factoring algorithm can solve the problems in polynomial time.

- **Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks**[12]: Blockchain is another big field that will be affected by post quantum Cryptanalysis, as the foundation of blockchain is cryptography and their ability to provide transparency, redundancy and accountability. T. M. Fernández-Caramès and P. Fraga-Lamas studied in their paper is the current state of the art on post-quantum crypto-systems and how they can be applied to blockchains and DLTs. Moreover, they analyze the characteristics and performance of the most promising post-quantum public-key encryption and digital signature schemes for blockchains. Through this paper they hope to provide a broad overview of the current state of blockchain and the future involving post-quantum blockchains, and their comparison.
- **Improved Analysis of Kannan's Shortest Lattice Vector Algorithm**[13]: Lattice-based is one of the most used algorithm for building a crypto-system such as NTRU, GGH and Ajtai-Dwork and relies upon the intractability of computing a shortest non-zero lattice vector and a closest lattice vector to a given target vector in high dimensions. In Hanrot, G., Stehlé, D. paper, they observed the utility of Kannan's algorithm and most of the algorithm to solve lattice-based problems are based on it. Kannan's algorithm for solving the shortest vector problem (SVP) is in particular crucial in Schnorr's celebrated block reduction algorithm. But Kannan's algorithm has not been improved in years. This paper proposes to provide new insight into the algorithm and improve its practical cost of solving.
- **Improved Dual Lattice Attack**[14]: MATZOV is an unit of the Israel Defense Forces(IDF). They made the affirmation that many post-quantum key exchange and signatures schemes are based on the hardness of the Learning With Errors (LWE) and Learning With Rounding (LWR) problems and their algebraic variants. Primal and dual lattice attacks are considered great against these types of problems, where dual attacks are generally considered less practical. In this report the institute analyzed several algorithmic improvements to the dual lattice attack, which allowed to exceed the efficiency of primal attacks. They also reduced the gate-count of random product code decoding and performing less inner product calculations. With this improvements to the cryptanalysis techniques, they concluded that the resistance of Kyber, Saber and Dilithium, the LWE/LWR based finalists in NIST rounds, are reduced by a great margin.
- **Implementing post-quantum cryptographic schemes on embedded systems**[15]: Mostafa Taha and Thomas Eisenbarth paper main question is: "Are we ready to implement post-quantum cryptographic schemes on embedded systems?" because of the vast creation of different post-quantum cryptographic schemes. Their study shows that we are still not ready to implement on embedded sys-

tems and there is still a considerable amount of research that needs to be conducted before reaching a satisfactory level of security.

As it can be seen, some protocols like SIKE/SIDH were already broken, and multiple improvements to different algorithms are being developed for attacks against these protocols to assure they are safe. All these papers are part of understanding the scope of the thesis better and what it could explore to provide a different vision on the state of post quantum cryptography.

4. Methodology

This chapter will describe the experiments done in the thesis. The first part will analyze Shor's algorithm and Grover's algorithm for understanding the problem that quantum computers create and observe why they affect the traditional algorithms. Then a deep dive into the families used for Post Quantum Algorithms in NIST Rounds. This will allow comprehension of different approaches against quantum computers and how they cannot break some mathematical problems that these algorithms are based on.
Then focusing on Lattice Space algorithm and its popularity by recreating a lattice space and understanding its complexity and why Shor's algorithm and Grover's algorithm would not be able to decrypt it.

5. Quantum Computer Decryption

Quantum computers are based on qubits, like traditional computers are based on bits. They represent more than just a 1 or 0, they can be any value in between. Qubits can be created using different methods, many implementations are using Josephson junctions. A Josephson junction[16] works by using two superconductors electrodes, and separating them with a thin insulator, so that particles can tunnel through, and remain trapped in the system fig. 5.1. The qubits using Josephson Junctions are called Superconductor qubits.

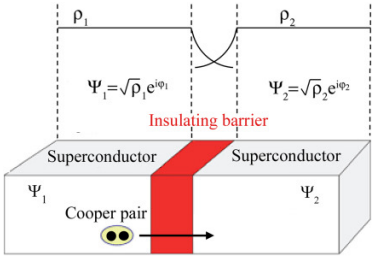

Figure 5.1.: Josephson junction representation. It is composed of two superconductors and a thin insulator that lets the particles tunnel through. The "X" representing a drop in current, is the most important part of this system and its property to let the particle go through. This way the particle stays blocked in this system and can move only between the superconductors[16].

The qubit is represented by a sphere, that is called a Bloch spherefig. 5.2 and is a geometrical representation of the pure state space of a two-level quantum mechanical system (qubit), named after the physicist Felix Bloch[17].

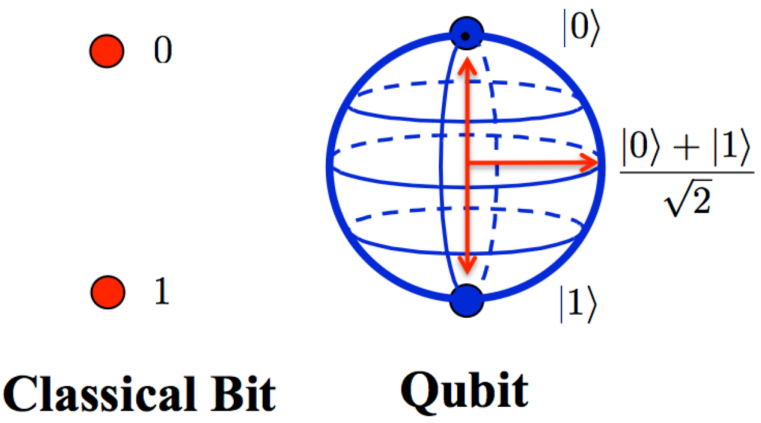

Figure 5.2.: On the left side there is the representation of classical bit, 0 or 1. On the right side we can see the complexity of the qubit, which can be either 0 or 1. The interesting part is the Bloch sphere which forms the qubit. The resulting qubit could be anywhere inside it, and the middle point is described by the formula presented in the image. This volume gives the qubit the ability defined by the Born rule or probabilistic output[18].

The formula for a qubit is $|\Psi\rangle = c_0|0\rangle + c_1|1\rangle$ and represents the superposition of a qubit with the basis states $|0\rangle$ and $|1\rangle$. **Quantum superposition** means that the qubit can be both values at the same time, or as described later parts of each value, the formula proves that the sum of these two variables generates a complete qubit, not only parts. The use of ket. Bra-"ket" notation, also called Dirac notation, is a notation for linear algebra and linear operators on complex vector spaces together with their dual space both in the finite-dimensional and infinite-dimensional case[19]. It is used in quantum computing because of what it represents, both finite and infinite results.

When measuring a qubit, the result is a probabilistic output of a classical bit. This probabilistic output is characterized by the Born rule[20]. It is a postulate of quantum mechanics that gives the probability that a measurement of a quantum system will yield a given result.

$$\alpha|0\rangle + \beta|1\rangle$$

where, α and β are complex numbers, satisfying

$$|\alpha|^2 + |\beta|^2 = 1$$

This can be recognized as $\sin$ and $\cos$ formula. Using the results of the $\sin$ and $\cos$ a circle with the radius of 1 can be mapped. It is also defined by its capability to have infinite results or a continuum number of

points. This is also reflected to qubits, $\alpha|0\rangle + \beta|1\rangle$ together form a wholefig. 5.3. A simple exemplification is imagining the qubit is $75\%|0\rangle + 25\%|1\rangle$, where α would be 75% and β is 25%, contrary to traditional computers where a bit can be either 100% 1(one) or 0(zero)fig. 5.4.

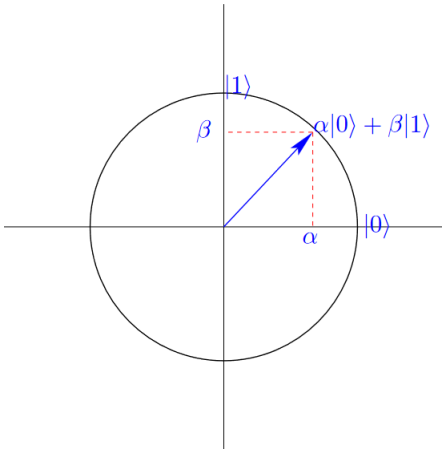

Figure 5.3.: This figure can be recognized as sin and cos circle or unit circle, which is well known for its capability of infinite results or a continuum number of points. Qubits share the same characteristic and can represents an infinite number of possibilities[21].

This capability to have infinite possibilities makes quantum computers important and the reason they are extremely fast and how they can calculate multiple solutions in seconds. Examples of systems that are being engineered to implement quantum computing include [18]:

- Trapped Ions: In these systems, strings of ions are stored in electromagnetic traps, and information is encoded on long-lived internal electronic or nuclear states of the atoms.
- Neutral Atoms: Atoms are trapped, e.g., in dipole traps generated by laser light, and again information is stored on internal electronic or nuclear states of the atoms and manipulated with laser light and inter-particle interactions.
- Superconducting Qubits: These involve circuits of superconducting material i.e **Josephson Junctions**.

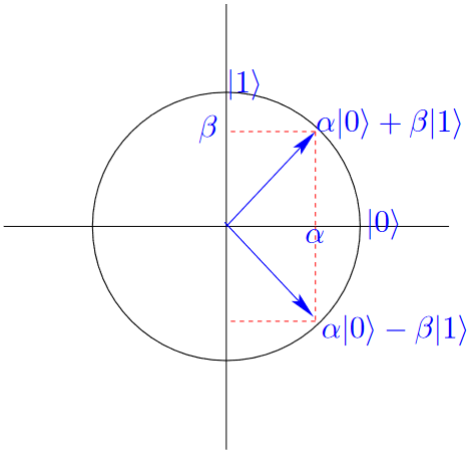

Figure 5.4.: This figure shows that $\alpha|0\rangle + \beta|1\rangle$ and $\alpha|0\rangle - \beta|1\rangle$ have the same probabilities for their measurement. However, they are distinct states which behave differently in terms of how they evolve[21].

Also, this changes the rules of implementing encryption so far. In the next section of this thesis it will be discussed DLP based encryption, including RSA section 5.1. This encryption based on the presumption that computers are very slow in computing the prime basis of a very big number. But using Shor's algorithm together with quantum computing we are able to compute them very easily, because of this statistically favorable answer given by quantum computers fig. 5.5.

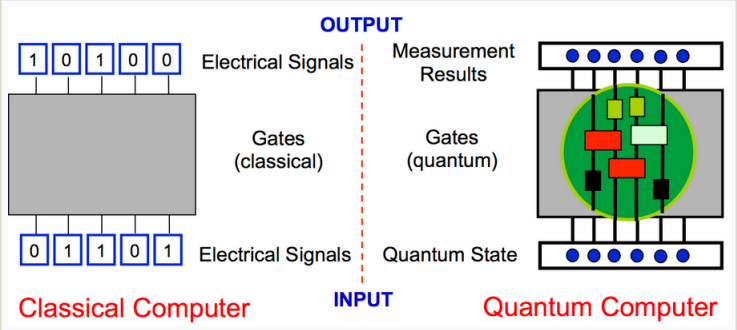

Figure 5.5.: Representation of traditional computers and quantum computers. For classical computers the input and output is the same, an electrical signal. For quantum computers the input is a quantum state and the result is decided through the measurement of the computation [18].

5.1. DLP based cryptography

In mathematics, logarithms and powers complement each other. Based on a 2D graph made by a horizontal Ox and vertical Oy axis, there is exponential growth, in which the formula $y = e^x$ is exponential and it will grow faster and faster approaching infinity. There is also the opposite, logarithmic growth in which at some point of x in $y = \log x$, the result y will remain steady around a specific number (as reference also $y = \sqrt{x}$)fig. 5.6.

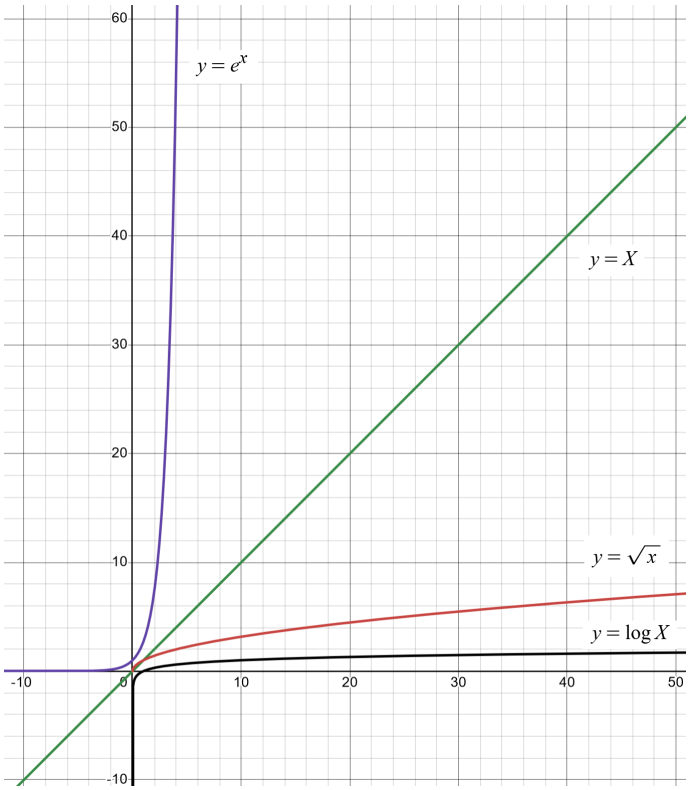

Figure 5.6.: Representation of time complexity of different functions (Big O notation). Ox axis is considered the input size and Oy is considered the time taking to solve the problem. The time it would take to solve a problem of complexity $O(e^x)$ is massively different from the speed it takes to solve $O(log(x))$. Added reference also $y = \sqrt{x}$.

Discrete Logarithm uses this strong correlation between the two functions, because $\log_b a = x$ and $b^x = a$ also use modular arithmetic, which is the rest of the division between two numbers. The definition for the DLP is that in any group G, powers b^k can be defined for all integers k, and the discrete logarithm $\log_b a$ is an integer k such that $b^k = a$[22]. DLP motivation is that mathematical problems are easy to compute, but hard to reverse, and knowing the right answer is even harder.

Diffie-Hellman is based on the Discrete Logarithm Problem(DLP) and was proposed by Whitfield Diffie and Martin Hellman in the context of cryptography and serves as the theoretical basis of the Diffie-Hellman key exchange and its derivatives[23]. Diffie-Hellman Problem (DHP), described as g is a generator of some group (typically the multiplicative group of a finite field or an elliptic curve group) and x and y are randomly chosen integers.

Discrete Logarithm Problem is the base for many cryptography algorithms today because some computations are hard to reverse. The number 527 which its the product of the multiplication between 17 and 31. Without knowing these two numbers it would be hard to deduce them. Modulo can make this even harder, applying $527 mod 143 = 98$ one can't come close to guessing the initial number, even less likely to discover the primes.

RSA (Rivest-Shamir-Adleman) is a public-key cryptosystem using asymmetric encryption. It is based on modular exponentiation, like Diffie-Hellman. Modular exponentiation is exponentiation performed over a modulus[24]. The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption. A basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d, and n, such that for all integers m ($0 \leq$m$<$n), both $(m^e)^d$ and m have the same remainder when divided by n (they are congruent modulo n).

However, when given only e and n, it is extremely difficult to find d. The integers n and e comprise the public key, d represents the private key, and m represents the message. The modular exponentiation to e and d corresponds to encryption and decryption, respectively. Even though it is an easy concept to grasp and use, this detail helped today's encryption to resist attackers and it is easy to compute. It does require a long time to calculate the encrypted message and decrypt even though it is much simpler than symmetric cryptography. Today's computers are not able to identify these numbers if they are very big because it will require immense computational power and time. RSA is also using prime numbers which are even harder to find because it requires a pool of validations and they are rare. Prime numbers are used because they are harder to be decomposed or guess, e.g. $323(17 * 19)$ than a composite number like $100(2 * 5 * 5)$.

Sieve of Eratosthenes is an ancient algorithm for finding all prime numbers up to any given limit. It can be implemented for calculating the prime numbers but still would take a long time because of the high range the RSA is using; 2048 bits to 4096 bits is recommended[25]. Sieve of Eratosthenes is very effective for finding all the primes up to a relatively small number. It runs in $O(N)$ space because of the boolean array and $O(n * log(log(n)))$ time. An implementation of Sieve of Eratosthenes, which uses a boolean array to keep count of the prime numbers, can be found in appendix A. It is based on marking the multiples of each number up to N (because multiples of any given number are not prime), where number N is the limit. The

result of input $N = 25$ would be:

```
Primes numbers up to 25 are:
2 3 5 7 11 13 17 19 23
```

RSA is doing something different to find primes, there are three steps[26]:

- Preselecting a random number of the selected bit-size
- Verify that the number is not divisible by a pre-generated list of prime numbers
- Apply Rabin Miller Primality Test iterations based on an acceptable error rate.

Rabin Miller Primality Test is a probabilistic algorithm which determines the likelihood of a number to be prime. It is a way to test if a number is prime without computing all the numbers up to it. For a given odd integer $n > 2$, write $n - 1$ as $2^s * d$ where s is a positive integer and d is an odd positive integer. Integer a, called the base, which is coprime to n (meaning they do not share any other factor than 1), then n is said to be a strong probable prime to base a if one of these congruence relations holds: $a^d \equiv 1 \pmod{n}$ or $a^{2^r d} \equiv -1 \pmod{n}$ (some papers write $-1 mod n$ as $n - 1 mod n$ which is the same thing, when using modulo on negative numbers you must add the modulo until it returns positive integer) for some 0 r < s (increase r one by one). This is done multiple rounds until a prime number is found, because these numbers are random it is not assure that the answer is right the first time.

Example of Rabin Miller Primality Test on number $n = 53$, $n - 1 = 2^2 * 13$ (is written as $2^s * d$), so $s = 2$ and $d = 13$, picking a random $a = 7 (2 \leq a < n - 1)$. Because the numbers are very big we will use Modular exponential, modulus is distributive, meaning it can be split among the numbers, e.g. $7^6 mod 53 \equiv (7^3 mod 53) * (7^3 mod 53) \equiv (343 mod 53) * (343 mod 53) \equiv (25 mod 53) * (25 mod 53) \equiv = 42 mod 53$

$$7^{13} mod 53 \equiv (7^6 mod 53) * (7^6 mod 53) * (7 mod 53) \equiv 42 * 42 * 7 mod 53 = 52 mod 53$$

$a^{s^0 d} \bmod n \rightarrow 7^{2^0 13} \bmod 53 \equiv 7^{13} \equiv 52$. Since $52 \neq 1$ but $52n - 1$

, we confirmed that 53 is prime.

This means that 53 is prime (which in this case is) or that the base that we chose $a = 7$ is a strong liar, otherwise we would have tested using $r = r + 1$, e.g $7^{2^1 13}$. In this case we would try multiple random bases a to verify our assumption that $n = 53$ is prime. The point of the whole Rabin Miller test is to verify that a number is prime, so why could a be a strong liar? Because this test may sometimes fail, in this paper F. ARNAULT proven that some composite numbers can also pass the Rabin Miller test [27]. Optimization were made to test some specific numbers that can help determine if a number is prime depending on the size of n. Gerhard Jaeschke in "On strong pseudoprimes to several bases" [28] and Carl Pomerance, J.

L. Selfridge and Samuel S. Wagstaff in "The pseudoprimes to $25 * 10^9$ (paraphrasing)" [29] shown that a specific set of numbers depending on the size of n can be used to quickly discover if n is prime or composite. Using these steps, RSA is able to generate high prime numbers that together compute a strong key.

5.2. Shor's algorithm

Different algorithms, like Shor's algorithms, have been invented to check the security of algorithms based on DLP. This conceptual algorithm is trying to verify the quality and authenticity of DLP algorithms before an attacker was able to find it. Algorithms are still being tested even after they are a standard, because attacks are ever evolving. Shor's algorithm [30] was developed by the mathematician Peter Shor and is used to find the prime factors of an(very big) integer. Most algorithms for finding the prime factors of an integer are in exponential time, while Shor's algorithm is polynomial. Going back to the figure fig. 5.6, we can see how fast the exponential equation grows; any polynomial equation is better than that. Specifically, it takes quantum gates $O\left((\log N)^2(\log\log N)(\log\log\log N)\right)$ time using fast multiplication, or even $O\big((\log N)^2(\log\log N)\big)$ utilizing the asymptotically fastest multiplication algorithm currently known due to Harvey and Van Der Hoven, where N is the size of the integer given as input. This is extremely fast for a problem that now would take decades to compute on classical computers. This algorithm is a problem for:

- The RSA scheme
- The Finite Field Diffie-Hellman key exchange
- The Elliptic Curve Diffie-Hellman key exchange

Some steps in Shor's algorithm are similar to Rabin Miller test(afterall, this is what the test does). Shor's algorithm takes the following steps to discover the prime numbers:

1. Check the number N isn't prime, even(divisible by 2) or of form $N = x^a$, where a is an integer that satisfies the equation (example: if $N = 27$, then N can be written as 3^3, x and a would be equal to 3)
2. Choose a random number $x \in [1..N]$
3. Check that $GCD(x, N) = 1$. This means that x is co-prime with N.
4. Find the minimum r such that $x^r \equiv 1(modN)$. If r is odd, then start over again with a new x.
5. Then $(x^{r/2}-1)(x^{r/2}+1) \equiv 0(modN)$ and therefore $GCD[(x^{r/2}-1)(modN), N]$ and $GCD[(x^{r/2}+1)(modN), N]$ are both factors.

Choosing a random number x is simple because there are already many methods to find common prime factors between two numbers using GCD (greatest common divisor), the best being Euclidean algorithms[31], reference appendix B. Subtracting a smaller number from a larger one (we reduce a larger number), GCD

doesn't change. So, if we keep subtracting repeatedly the larger of two, we end up with GCD. Now instead of subtraction, if we divide the larger number, the algorithm stops when we find the remainder 0. This returns us the result very quickly for these two numbers $a = 33521 * 11897$ $b = 33521 * 24281$ (the numbers were generate using a random prime number generator):

```
Iteration 0:a=398799337 b=813923401
Iteration 1:a=16324727 b=398799337
Iteration 2:a=7005889 b=16324727
Iteration 3:a=2312949 b=7005889
Iteration 4:a=67042 b=2312949
Iteration 5:a=33521 b=67042
Iteration 6:a=0 b=33521
Prime common factor between 398799337 and 813923401 is: 33521
--- 0 ms ---
```

The bigger the gap between the numbers the faster the process would work. But using only this would not yield any good results, because it is very unlikely to pick a number that shares a prime factor with the number N. That is why these formulas $(x^{r/2} - 1)(x^{r/2} + 1)$ are a better guesses based on a mathematical fact that if x, N are integers with no common factors, if you multiple x enough times by itself the result will be: $x^r = c * N + 1$ or $x^r - 1 = c * N$ (which gives $(x^{r/2} - 1)(x^{r/2} + 1) = c * N$) where c and r are some constant numbers. Randomly assigning x will give a chance of 37.5% of being right("there is at least a 37.5% chance this method will work")[32], but in the case the number gives a false answer, you can just assign another time a random value [33][34].

Everything done so far can also be done by a classical computer, why are encryptions still safe now then? Brute forcing r would take a long time using classical computers, especially with big numbers, but a quantum computer is great for this task because of the property described earlier in fig. 5.2, superposition. Quantum computers can formulate an equation that computes all the possible answer and returns only the right one (or most probable one) because the other answers destructively interfere with each other. Shor's algorithm is using this property of probabilistic answer for computing r.

One last mathematical rule that quantum computers and Shor's algorithm are using is the exponentiation repetition. If $x^p = m * N + k$, if r is the right answer then also $x^{p*r} = m * N + k$, where p, m and k are constant numbers, and r is the answer from before. This gives the quantum computer equation the power to destructively interfere with the other answers and you will be left with a series of answers with a period of r which is a more likely answer because k will be more present in the initial series (e.g.

$|z + r, k\rangle, |z + 2 * r, k\rangle, |z + 3 * r, k\rangle$, where z is a constant). Using a Quantum Fourier Transform on the previous result $|z + r, k\rangle, |z + 2 * r, k\rangle, |z + 3 * r, k\rangle$, will return the frequency of the series $|1/r\rangle$. These are the basics of Shor's algorithm and why it can compromise RSA, because RSA keys are formed using two big prime numbers[35]. This algorithm on quantum computers is very fast compared to traditional application. Nevertheless, quantum computers are still not big enough. The biggest one, as of writing this, is Condor by IBM and it has 1,121 qubits which is not enough for even the smallest RSA key (you would need 5,900 qubits for RSA-230)[36]. DLP based asymmetric crypto is still secure, but as evolution already proved itself referencing Moore's law, everything could change very fast: "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years"[37].

5.3. Symmetric-key primitives

Cryptographic primitives are low level algorithms that are used to build cryptographic protocols. Symmetric-key primitives, like asymmetric keys discussed in section 5.1, are very common and they are the basis of many applications. For example a document encrypted using a pre-shared key, can be accessed only using that key, but in the case of asymmetric keys there is a public and a private key. The public key is used to encrypt the document, and the private key to decrypt it. When using the pre-shared key there are many ways to use it, like using key agreement protocols to transfer it or a usb stick like a yubikey to store it physically. It is very important not to lose this key, otherwise it will not be possible to decrypt the document or if someone else obtains the key, he will be able to decrypt the data.

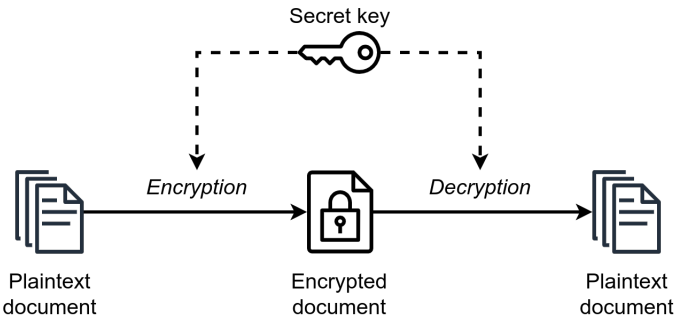

Figure 5.7.: The flow of encryption and decryption in symmetric key protocol. The same key is used to both encrypt and decrypt the message. Losing the key can disrupt the access to the file. If an attacker gets the key, they can decrypt the file/contents.[38].

AES (Advanced Encryption Standard) is an symmetric-key algorithm. It was selected by NIST in 2001 under as Federal information processing standards (FIPS) 197 through a competition, same as now with post quantum cryptography. It uses a table that is publicly available for substitution of the bytes. This table is called Rijndael S-box, even though this table that is very important for the encryption is publicly available, it does not constitute a liability for the algorithm.

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 5.8.: Rijndael S-box from NIST FIPS 197[39]

Attackers cannot use this table alone to decrypt the message, there are many layers to encrypting AES, and the S-box is just a small step. AES works by breaking the message into groups of two hexadecimals, then it correlates maps the data to the Rijndael table and then substituted the value. This algorithm is also using round keys which are part of AES key schedule and are derived using the master key. Again this is publicly

available [39] and is just a function that is based on the hexadecimals of the message used in calculating the round key.

These are the steps for encrypting one round of AES:

1. **AddRoundKey** - each byte of the state is combined with a byte of the round key using bitwise XOR.
2. **SubBytes** - a non-linear substitution step where each byte is replaced with another according to a lookup table.
3. **ShiftRows** - a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
4. **MixColumns** - a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
5. **AddRoundKey** after each round. This algorithm has 10,12 or 14 rounds. (depending on the desired security, grows the time and complexity)
6. In the final rounds it is only **SubBytes**, **ShiftRows**, **AddRoundKey**.

These steps are done multiple times, and this depends on the size of the key: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. Nowadays the recommended key is 256-bit because of quantum attacks discussed in the following sectionsection 5.4.

A good cipher uses **confusion and diffusion**, defined by computer scientist Claude Shannon in 1940 [40]. **Confusion** is the act of changing something (a to b) also known as substitution. **Diffusion** is arranging the data in a different way (e.g. $abcd$ -> $cbad$), named also transmutation or permutation. AES is using both, and is very hard to decrypt, because even though these steps are well known and the table for Rijndael S-box is publicly available this algorithm still manages to obfuscate the data for the attacker because he does not have the right sequence of transmutation and also does not know the substitutes. Without the key, it is almost impossible to discover the changes that confusion did, or the initial order of data (these steps are done many times, so the initial data is "lost" without the right reverse order). AES stands at the top of algorithms for protecting everyone's data. It's used worldwide, it has been adopted by many companies and even governments.

5.4. Grover's algorithm

Grover's algorithm was developed by Lov Grover in 1996. It is a quantum algorithm that finds with high probability the output of a black box function with $O(\sqrt{N})$, where N is the size of the function[41]. This algorithm can be used to find a desired answer to an unsorted array. In classical computing it is not possible

to compute this in less time then $O(N)$, meaning the worse case scenario would be searching the whole array. Many algorithms were invented to shorten this time but none as fast as Grover's. "Divide et impera" or "divide and conquer" is a very well known algorithm for sorting an array, which divides each subsequent array in half and combines them in the end. After having the array sorted one can use a Binary Search(which takes a mid point, checks if the data is in the upper or lower group and repeats these steps), which is a very effective way to finding a result in $O(\log_N)$, **with the condition that the array is sorted**. But even with all optimizations, one would not be able to get the result of Grover's algorithm using quantum computing. In fig. 5.6 outlines clearly the improvements of Grover's algorithm. The main focus of the algorithm is to flip the desired value and then amplify the amplitude of the object searched and decrease all other probability amplitudes to increase the chance of the correct answer through disruptive interference and balanced superposition, discussed in chapter 5.

The steps of Grover's algorithm are:

1. Start with a balanced superposition and assign to the chosen ket a phase of -1 using the formula $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$. So the balanced superposition:

   $$|\Psi\rangle = \frac{1}{\sqrt{4}}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle + \frac{1}{2}|3\rangle$$

   and assigning $|11\rangle$ as the chosen ket the formula will become:

   $$|\Psi\rangle = \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle - \frac{1}{2}|3\rangle$$

   and this is called U_f.
2. Perform the Grover iteration $r(N) \approx \frac{\pi}{4}\sqrt{N}$ times by applying the operator U_f and the Grover diffusion operator $U_s = 2\,|s\rangle\,\langle s| - I$. By doing this, it amplifies the correct answer and provides a higher chances of finding it, because the amplitude will be much higher than the rest of the kets in the end.
3. By measuring the result it should able to observe the correct answer.

How can this algorithm damage the security of existing encryption? A common way to protect users accounts is by hashing the passwords using a salt (a random "key"). Even if an attacker obtains the database of accounts, he would not be able to use the accounts, because the passwords are not in plaintext, but he could compare the hashes in the database to already known hashes from passwords that were converted. This process takes a long time because of the traditional finding problem, which takes $O(N)$, and as discussed using Grover's algorithm this takes $O(\sqrt{N})$. This is only a small example to the capabilities of this algorithm, not only in security but also in real world situations. In the future when quantum computers are the norm it will improve the speed and capabilities of softwares and servers by a large margin. In the case of AES,

is similar to password hashing, comparing the result encryption to existent results, increasing the speed by which the right answer is found and information is obtained. Nowadays AES-128 would take 2^{128} iterations to find key, but using Grover's it would take $\sqrt{2^{128}}$ which is 2^{64}. It still takes a long time to break even the AES-128, now the recommendation is to use only AES-256 because of the halving which can damage the longevity of the encryption. If a perpetrator is really determined and believes the document is important, he will wait, so any improvement to security is important.

5.5. Remarkable mentions

One remarkable mention is XOR, it provides security unaffected by changes in technologies[42], it uses the XOR operator between bits. This is an additive cipher and is not used by itself, it can be easily brute-forced and find the answer using today's technologies. Even if an answer is found, someone cannot know if it is the right one. Imagining the word "TOY", which is formed using three letters, by using XOR and not knowing the right key or sequence, the result can be any combination of three letter words like: "JOY", "LOT", "HOT", etc. In a big text, it might be able deduce the right text, but it is still a very good way to disable automation because a human should verify the correctness of the resulting decrypted text.

6. Post-Quantum Cryptography Families

After analyzing traditional cryptography, quantum computers and their new algorithms that can break them, its obvious why new algorithms are needed to assure the security of the IoT devices, softwares and information. The focus of security experts and especially National Institute of Standards and Technology (NIST) is on the following families which present a promising success in the future. Each of them focus on a hard mathematical problem which proves to take a long time to solve, even by using the best technologies nowadays. One of them stands out in particular because of its proven security and small key size with a different approach from the others. That is the lattice based crypto-family explained in chapter 7, which is the most used for NIST challenge competitors and passed so far the tests that they were subject to and onto which the thesis will be focusing in depth. This will be done by analyzing the problem and how hard the solution is, as well as some caveats that have to be thought of while implementing it.

All these algorithms have to abide by security reductions, which is the correlation of a cryptographic algorithm and a known hard mathematical problem, they are proofs to the difficulty of cracking the algorithm. In other words, the harder the mathematical problem proves itself to be solved by any means, the stronger the encryption. The mathematical problems that these encryptions are based on must be NP-hard. NP stands for non polynomial time to solve a problem. NP-hard is a type of problem that is hard to solve but easy to verify by having the answer, and they cannot be solved in polynomial time, meaning they would take a very long time to solve. This is a must for a strong encryption, because it offers the assurance that without the answer/key, there is no possible way to find it.

As an example, taking a complex polynomial system with n variables and m equations, where $m >= n$ there can be a solution. This system is also known as a matrix, where there are three matrices, for this case it will be marked as A, B, C matrices. A the matrix of coefficients, B the variables, and C the matrix of results.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

$$A * B = C$$

Where a_{xx} are the coefficients, x_x are the variables, c_x are the results. Matrix multiplication is "not commutative", meaning the order of multiplication matters $A * B \neq B * A$. It is hard for humans to find the solution, the same applies for computers, and still for quantum computers. But having the variables known, it is easy to input them and verify the correctness of the equation. This makes a problem NP-hard, in this case the polynomial system.

6.1. Multivariate cryptography

This cryptographic algorithm is based on multivariate polynomials over a finite field F, F representing a set of equations. This is in fact asymmetric cryptographic primitives, and as discussed RSA is based on it too. If the polynomials have the degree two, we talk about multivariate quadratics, which could help improve the security. Solving systems of multivariate polynomial equations is proven to be NP-complete. Although it is a strong candidate for post-quantum cryptography, it proves to be more efficient to build signature schemes primarily because multivariate schemes provide the shortest signature among post-quantum algorithms. The signature version for this algorithm is using the private key and then is verified using the public key to prove the correctness. Then using a known hashing function, the signature is hashed. The receiver of the signed document must have the public key in possession, then computes the hash and checks that the signature fulfills the equation. Unbalanced oil and vinegar schema (UOV) is based on this algorithm and is a modified version of the oil and vinegar scheme designed by J. Patarin, and both are signature protocols[43].
UOV is based on a system of equations, similar to the example above.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

For the **signature creation** a system of equations must be solved. The $y = (y_1, y_2, \ldots, y_m)$ is a given message which should be signed, where y_x is a function of type $y_x = f_x(x_1, \ldots, x_n)$. The valid signature is $x = (x_1, x_2, \ldots, x_n)$, the solution of this system. An oversimplification example from the system of equation above would be: $y_1 = a_{11} * x_1 + a_{12} * x_2 + a_{13} * x_3$. Each function is of type $y_i = \sum \gamma_{ijk} a_j \acute{a}_k + \sum \lambda_{ijk} \acute{a}_j \acute{a}_k + \sum \xi_{ij} a_j + \sum \acute{\xi}_{ij} \acute{a}_j + \delta_i$. The coefficients, $\gamma_{ijk}, \lambda_{ijk}, \xi_{ij}, \acute{\xi}_{ij}, \delta_i$, must be chosen secretly, meaning that it is very important to be truly "random" (it is hard to achieve real randomness, because computer randomness is based on seeds or clock time, it can be achieved using lava lamps for example [44])

and not shared. This again is just an equation which is part of the system, and is describing the creation of such equation to make it secure and hard to be solved.

The vinegar variables $\acute{a}_j$ are chosen randomly and the resulting linear equations system gets solved for the oil variables a_i. The vinegar and oil variables build the pre-signature $A = (a_1, \ldots, a_n, \acute{a}_1, \ldots, \acute{a}_v)$. Finally A gets transformed by the private transformation S to give the valid signature $x = S^{-1}(A)$.

Signature validation is done through a similar system of equations and is performed using a public key. This system is different so that an attacker cannot discover the secret coefficients used. Every function has to be solved for the validation of the signature.

This complexity can be boiled down to some main points. Creating a system of equations and choosing the secret coefficients. Vinegar variables are chosen randomly and then generating the valid signature. For validation a similar system is given but different so a perpetrator cannot figure out the secret coefficients from the original one. This system is very hard to be solved and its complexity is NP-hard. Requires many tries and mathematical formulas to solve if the solution is not known, as discussed before using a system of matrices. Some encryptions that are using this type of algorithm in the NIST rounds are: Rainbow [45] which focus is on signature and Giophantus [46] for PKE/KEM (Public key encryption/ Key encapsulation mechanism)

6.2. Hash-based cryptography

Hash-based cryptography was created as an alternative to number based algorithms, like DSA or RSA, which are based on logarithms and exponents. The foundation is based on hash functions and especially on Merkle trees invented by Ralph Merkle an American computer scientist and mathematician [47]. Hash functions work based on the assumption that providing a unique input it will return a unique output, so the answer cannot be replicated unless given the right input. The hash function used to convert this data is almost impossible to be found. This cryptography scheme uses Merkle trees and they work by hashing the data blocks into one root hash. This scheme is more use for digital signatures and validation, and is very common in blockchains where validation needs to be done at top level hash.

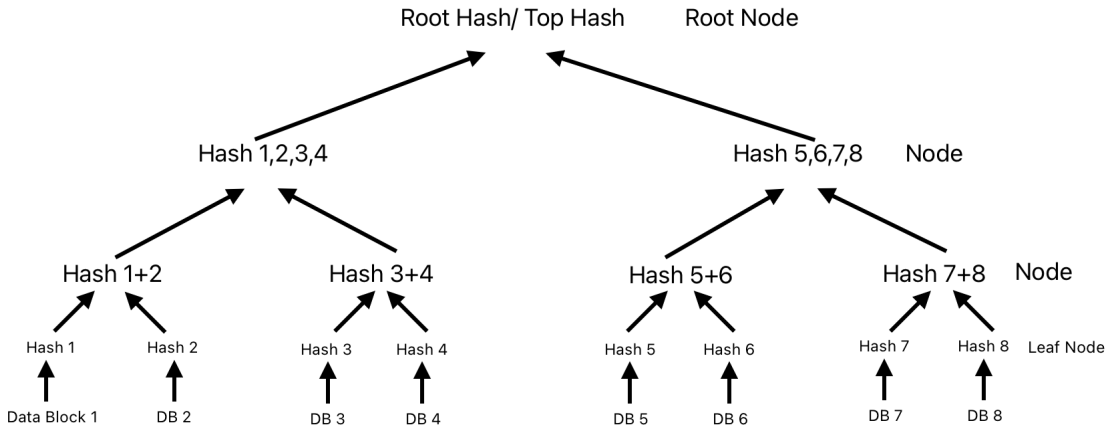

Figure 6.1.: In this figure is presented the way Merkle Trees work, starting from the data blocks and hashing them. Afterwards combining the hashes until you get to top level hash. This root hash can be used to validate all the other hashes composing it, without needing the other data blocks. Thats the usefulness of Merkle Trees, they can quickly validate the data blocks without any other information needed, without keeping the hash of each data block and just combining them into one.

Although they are very effective, they do have a big drawback, the capability of any hash-based public key has a limit on the number of signatures that can be signed using the corresponding set of private keys. Because of this fact, there are not many implementations in traditional programming, but their resistance to quantum computing rose the interest again. There are no quantum algorithms that can discover the hash function based on the answers, even if many answers using the same function are discovered. The reason is that a lot of functions can give the same answer, so there is no way to discover which function is right. But the drawback of one-time or bounded-time functions even with the addition of the security still remains. In 1989, Moni Naor and Moti Yung designed a signature with unlimited-time in use based on hashing which can be used[48].

As of writing this paper the NIST's finalist were published, SPHINCS+[49] is one of the finalist and recommended algorithm for digital signature.

6.3. Code-based cryptography

Code-based cryptography works by using error-correcting code or ECC. Error-correcting code is an old mechanism of finding errors in the bytes of data transmitted. The first implementation of such algorithm

was developed by Richard Hamming[50] and it used a basic logic principle, by cross scanning the bytes and identifying changes in bits. Richard Hamming initially developed "Hamming codes", error-correcting codes, because of the errors introduced in punched card readers, around 1950. It could detect only an odd number of error bits, but it was very useful in that period especially because its use size was very small and does not occupy much space from the data block (which was already small enough).

The purpose of this cryptography scheme is to have the error correcting code for solving an ambiguous data block. Without it, the data makes no sense and cannot be decrypted, is just a random sequence of bits. In the beginning, computers would use punching cards as a way to store data and there would be readers using electromechanical unit record equipment that used mechanical brushes. If there was a hole, these brushes would touch, sending that data to the computer. But this technology sometimes would give errors, and even more recent ones like CDs would be prone to such reading errors (because of scratches on the surface, the reading using laser would be inaccurate).

Hamming codes are a very elegant implementation that helps detecting such errors when reading storage. The data in reality is just an array of 1's and 0's. Flipping these bits can result in incorrect results from the initial data: $0b1010 = 10$ but $0b0010 = 2$ ("0b" denotes writing in binary, or enclosing the number in parentheses and writing the base e.g $(1010)_2 = (10)_{10}$, where $(x)_{10}$ denotes decimal, base 10 writing is usually implied if there are no specifications). Now if the data gets changed along the transmission no one could verify its authenticity. One solution to this could be to reserve one bit to verify the parity of the data that was received.

Figure 6.2.: This is a block of data, composed of 16 bits or 2 bytes. The first bit is reserved to denote the parity of the block. It is 0 if the number of bits of **one** is even or 1 if the they are odd. In the $EX1$ it can be seen that the number of 1 bits are 8 so they are even, as a result the reserved parity bit is 0. Also in $EX2$ the reserved bit becomes 1 for the purpose to make the block even overall. Green and Blue was used to depict the Special Bit, Green-0 and Blue-1 – Yellow was used to denote the bits of 1 and no fill was used for 0. The Red color was used to drive attention to the switched bit in the second scenario where the parity bit had to be changed.

Using this strategy, it can be detectable if an error within the data block occurred by counting the numbers of ones and compering them to the special bit.

Figure 6.3.: Depiction of a block of data that contains an error bit. The special bit is set to 1 but the number of 1's in the data block is even (8 bits of 1s). Using the special bit an error has been visualized. Green and Blue was used to depict the Special Bit, Green-0 and Blue-1 – Yellow was used to denote the bits 1 and no fill was used for 0. The Red color was used to drive attention to the switched bit from 6.2 $EX2$ and the remained Parity bit of 1, which indicates an error within the data block.

This is the basis of Hamming codes, in the data block there is a special bit that is reserved for the parity of the content, and it can easily be verified. But it can already be seen there are two problems:

1. The position of the wrong bit cannot be determined.
2. What if the numbers of errors are a multiple of two(the parity bit would be right because it cannot deduce the number of error bits) or more errors than one?

These problems are described in the following paragraphs.

1. For solving the errors in data blocks, it is not useful enough to know just that something is wrong, there should be a way to determine what is actually wrong, or more specifically the bits that are out of place. For this Richard Hamming came with a great solution, just add more parity bits. The only drawback is again, consumption of useful data bits. Using the same concept but on specific lines instead of the whole block, and grouping those lines for memory reduction purposes, it can also be determined the position of that wrong bit.

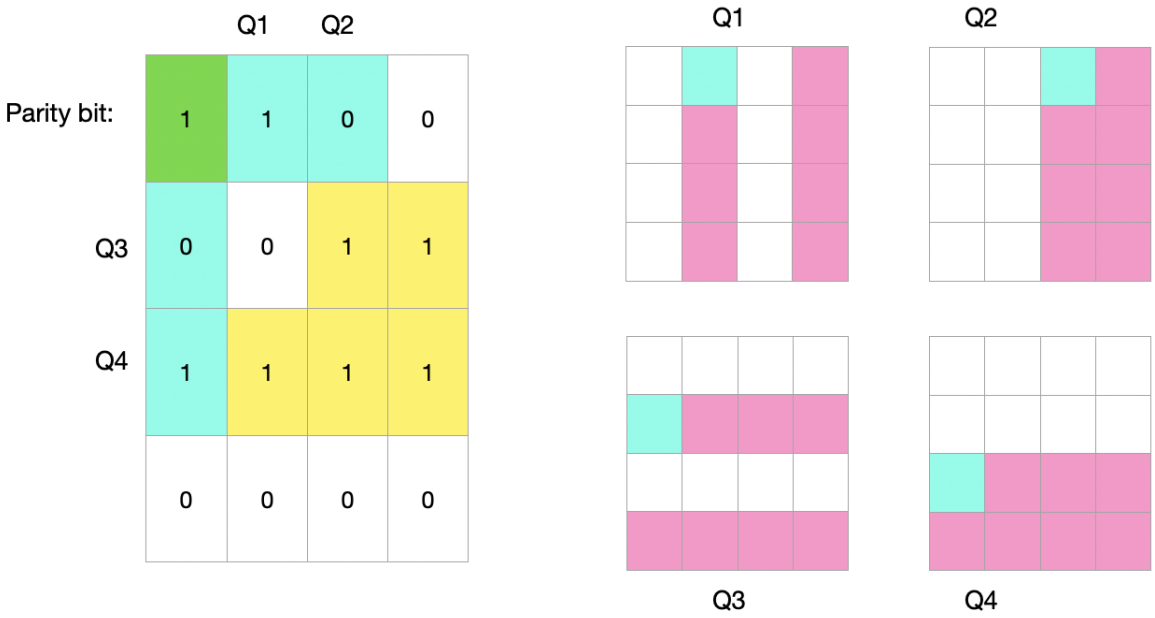

Figure 6.4.: In this figure, there is still the same parity bit at the beginning of the data block. Additionally there are 4 more bits, denoted in blue, which are used to check the lines denoted in pink. As an example for $Q1$, it calculates the parity for column 2 and 4. The parity bit for $Q1$ is 1 because the number of 1's bits on these lines are 3. For this 16 (or 2^4) bit block, there are 4 quadrants. Green used to depict the Parity Bit, which can be either 0 or 1 – Yellow was used to denote the bits of 1 and no fill was used for 0. Blue color was used to drive attention to quadrant bit, which can be either 0 or 1. Pink was used to represent the lines which quadrant bits cover in their calculation.

This elegant solution gives a more precise location of the bit at fault by sequentially going through each block, it can easily be pin pointed by marking the positions that overlap.

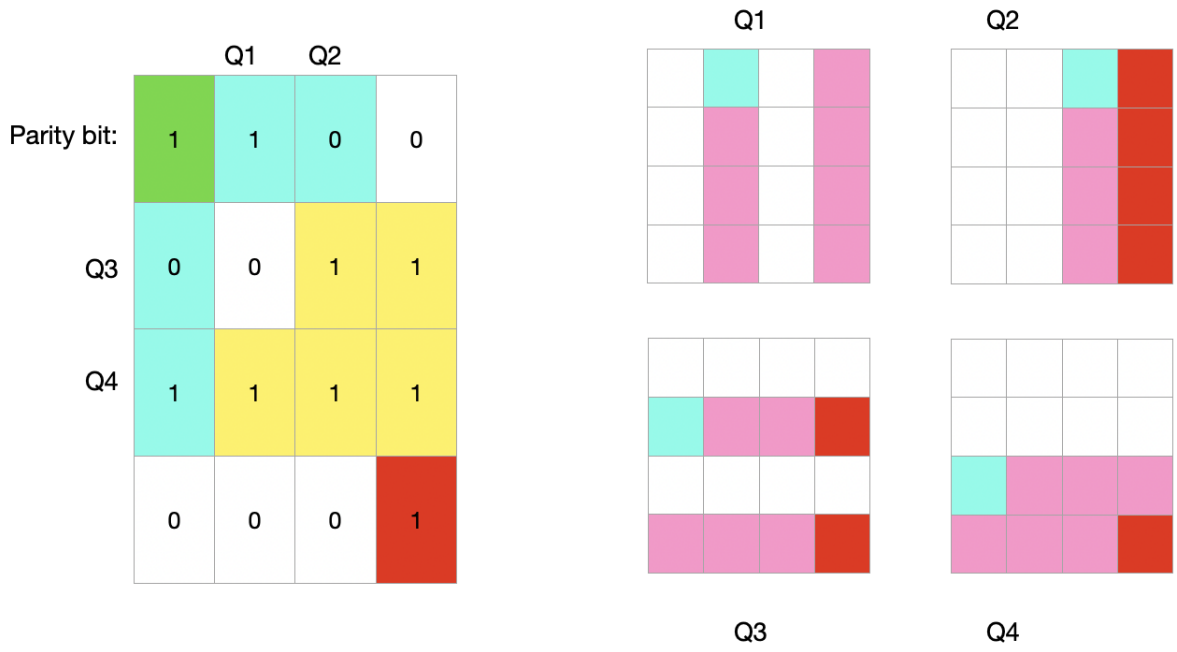

Figure 6.5.: An example of how Hamming Code works. All parity bits are wrong marking that in all quadrants there is a misplaced bit. Sequentially searching each quadrant and adding them together, the exact location of the error bit can be deduced. Same colors were used for marking purposes as in fig. 6.4.

This innovation can be applied even to larger blocks of data, by just expanding the parity bits, which are not chosen randomly. At closer look at the parity bits, can be seen that they are at positions of powers of 2.

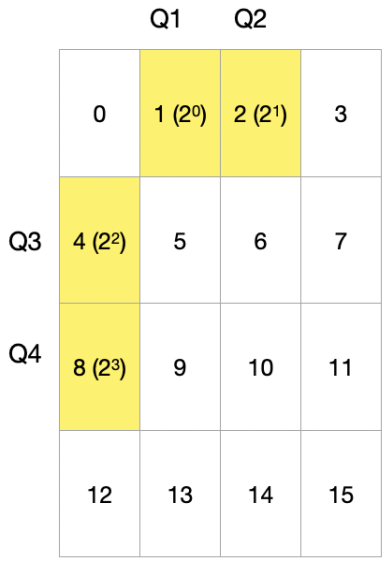

Figure 6.6.: Depiction of the bits position, and the parity bits in each quadrant are located in the position of power of 2. Using this method they can easily be located, even for bigger blocks and are a reserved to be power of 2. Yellow was used to indicate the position of the quadrants bits.

2. This solution proved to be useful, but what if the number of error is even, or bigger than two errors. This is the limitation of this code, it can detect one specific error in the data block, but it can only identify if there are more than 2 errors without correcting it. Also what was discussed so far is an Extended Hamming Code because it uses the 0 position parity bit. Without it usually you would eliminate the 0 position bit and it would become a 15 bit block of data, where 4 bits are reserved, this is called a (15,11) Hamming Code. In case of Hamming code you still couldn't use the 0 position bit and you would lose one bit of data. The advantage of using Extended Hamming Code the bit is put to use and offers an additional check, is beneficial in most cases (depending if the error can be observed from parity, like discussed), otherwise a bit would be lost for every block.

This begs the question, what does error correcting code has to do with cybersecurity and more specifically cryptography? Quantum computers are also not good at detecting errors, they are not able to detect if a block of data is wrong, like any other traditional computer. The transfer of data can be sent having errors, by having a specific error correcting code the receiver would be able to decrypt it and understand the message. It is like creating a decipher for understanding the message, for outsiders is none sense, but for those having the solution is valuable information.

Some algorithms that use code-based cryptography are McEliece[51] and Niederreiter[52] (which is based

on McEliece cryptosystem) encryption algorithm. Message encryption is done by encoding the message as a binary string, then a random $n-$bit vector is generated using to compute the cipher text. To decrypt the message a decoding algorithm is necessary. Another system based on code based algorithms would be Sardinas-Patterson Algorithm [53], which purpose is to check if a code sequence is uniquely decodable. This could be used also as a cryptoscheme, where a set of characters is created with a corresponding binary code used to decrypt the message. The algorithm works on the values that were already generated, creation is based on the known values, using composition (that is, $x_1w = y_1$ for some suffix w. If one tries first $x_1 = 011$ and $y_1 = 01110$ the suffix is $w = 10$). For example having the set code $\{ a \mapsto 1, b \mapsto 011, c \mapsto 01110, d \mapsto 1110, e \mapsto 10011 \}$ it is not possible to decrypt without the order of keys because it is not uniquely decodable – 011101110011 can be interpreted as the sequence of codewords $01110 - 1110 - 011$, but also as the sequence $011 - 1 - 011 - 10011$ (example is based on Berstel et al. (2009), Example 2.3.1 p. 63).

Encryption and decryption can only be done using the unique set of characters defined. Having the encrypted message a computer would not be able to decrypt it, because it is just a string of ones and zeros.

Although this is a great solution to post quantum cryptography, precaution should be taken into assuring that it is safe enough. In the paper "Timing attacks on Error Correcting Codes in Post-Quantum Schemes" written by Jan-Pieter D'Anvers, et. al. [54], they discuss the vulnerabilities encountered on ECC using side-channel attacks by using timing information to distinguish between cipher texts due to the variable execution time of the ECC decoding algorithm. They were testing this theory on two round 1 candidates to the NIST Post-Quantum Standardization Process: the Ring-LWE scheme LAC and the Mersenne prime scheme Ramstake (a candidate using code-based family) and can obtain the full secret key using a limited number of timed decryption queries with minimal time (under 2 minutes). The most important attack strategy against the "one way encryption" of the original McEliece crypto-system is information-set decoding (ISD), which was introduced by Prange in 1962 [55][56] .

6.4. Isogeny-based cryptography/ECC

Isogeny-based cryptography is known because of elliptic curves (also called Elliptic Curve Cryptography/ECC), which is used by Diffie-Hellman-like key exchange CSIDH, an alternative to Diffie-Hellman and elliptic curve Diffie-Hellman key-exchange methods. RSA and Diffie-Hellman was discussed in section 5.1. This crytoscheme is based on isogeny graphs of elliptic curves and abelian variety over finite fields, and very specifically super-singular isogeny graphs. The math behind this family is very complex.

An elliptic curve has the mathematical generic formula $y^2 = x^3 + ax + b$, this elliptic curve is written in a Weierstrass form.

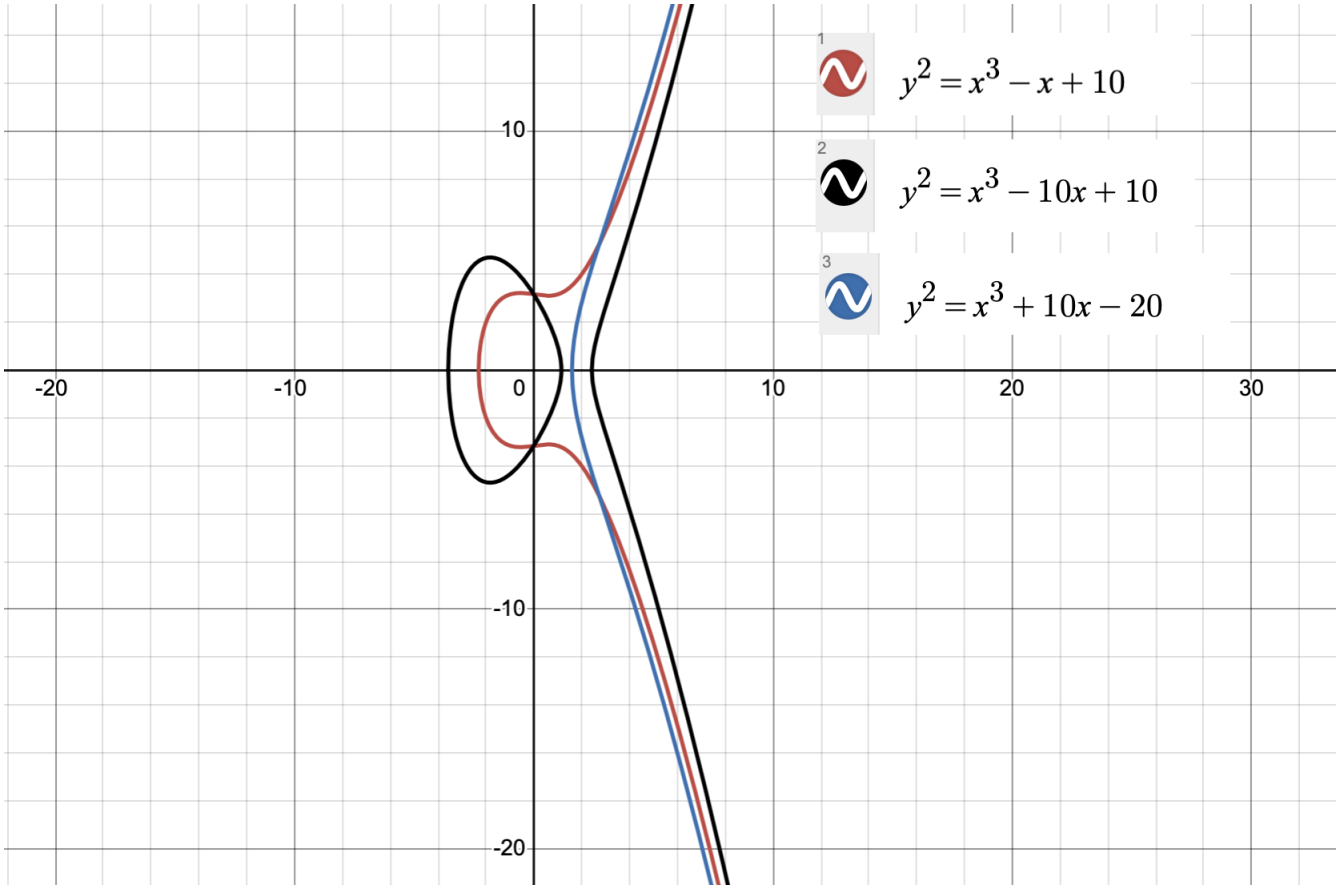

Figure 6.7.: Examples of elliptic curves defined over $\mathbb{R}$, and the different shape they can take. Red line is a generic elliptic curve with formula $y^2 = x^3 - x + 10$. Using black it depicts an elliptic curve that forms a blob $y^2 = x^3 - 10x + 10$. In blue it can be seen that the elliptic curve ($y^2 = x^3 + 10x - 20$) is looking like a parabola sideways (the general formula of parabolas is $y = ax^2 + bx + c$, but it can also be written sideways by changing x and y, $x = ay^2 + bx + c$ fig. 6.8)

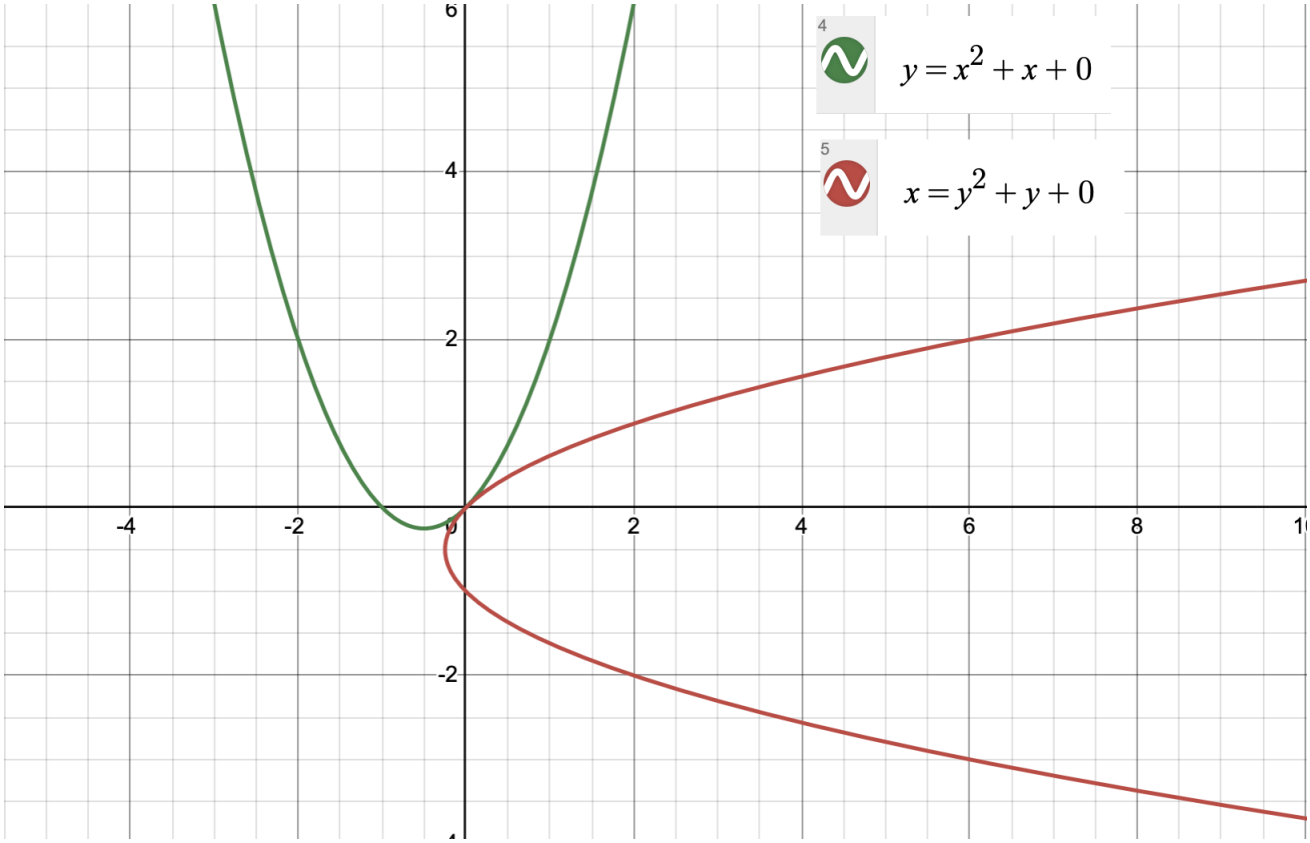

Figure 6.8.: For reference, generic parabola using green and formula $y = x^2 + x + 0$ and sideway parabola in red using $x = y^2 + y + 0$. It can be seen the similarities between the formulas and mapping of the curve.

Elliptic curves have torsion points, which can be used to find their "Origin point" if they are multiplied by themselves n times [57], meaning for some value l there is $l * X = X$, where X is a point on the curve. The property of a torsion point in case of elliptic curves is that $l * X = X + X + ... + X = 0$, for a point X in the group E, where E is an elliptic curve over F, and $l > 1$, then X is an $l - torsion$ point. Abelian group is a set A, together with an operation $\cdot$, that combines any two elements a and b of A to form another element of A denoted $a \cdot b$. A group is an Albeian group if it satisfies the four axioms: Associativity ($(a \cdot b) \cdot c = a \cdot (b \cdot c)$), Identity element (an element that is neutral, $e \cdot a = a \cdot e = a$), Inverse element ($a \cdot b = b \cdot a = e$, where e is the identity element), Commutativity ($a \cdot b = b \cdot a$) [58]. Identity element is a neutral element (like 1: $1 * 5 = 5 * 1 = 5$).

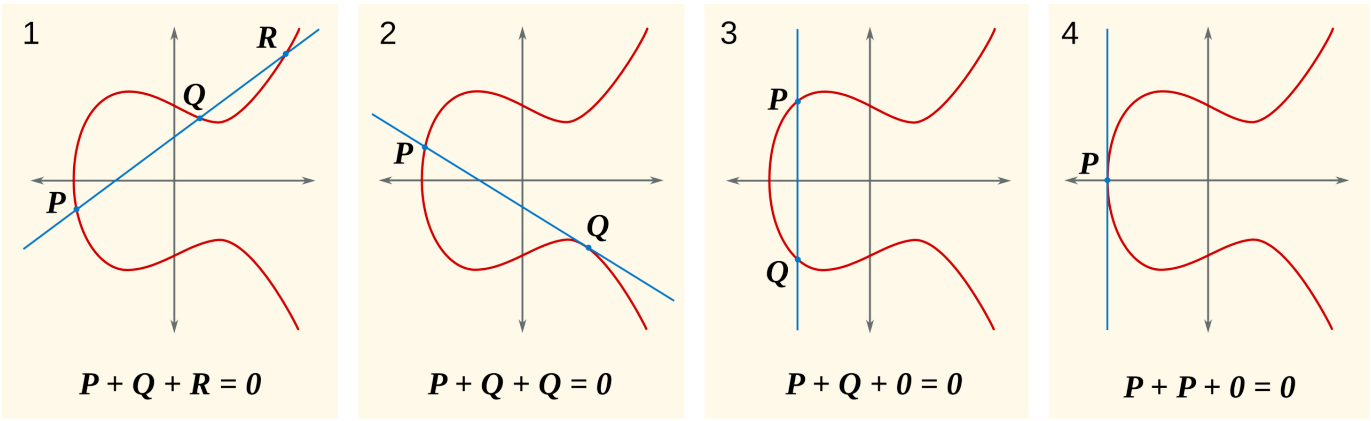

Figure 6.9.: These are the properties of a Weierstrass elliptic curve that enables an elliptic curve defined over group $E(k)$ (where E is an operation group defined over k, k can be $\mathbb{R}$) to be an Abelian group, it has the four axioms discussed: Associativity, Identity element, Inverse element, Commutativity. 0 (zero) is the point at infinity, also the identity element. Three points on a line sum up to 0. In this case the $\cdot$ function is addition $+$. For a better understanding refer to [59].

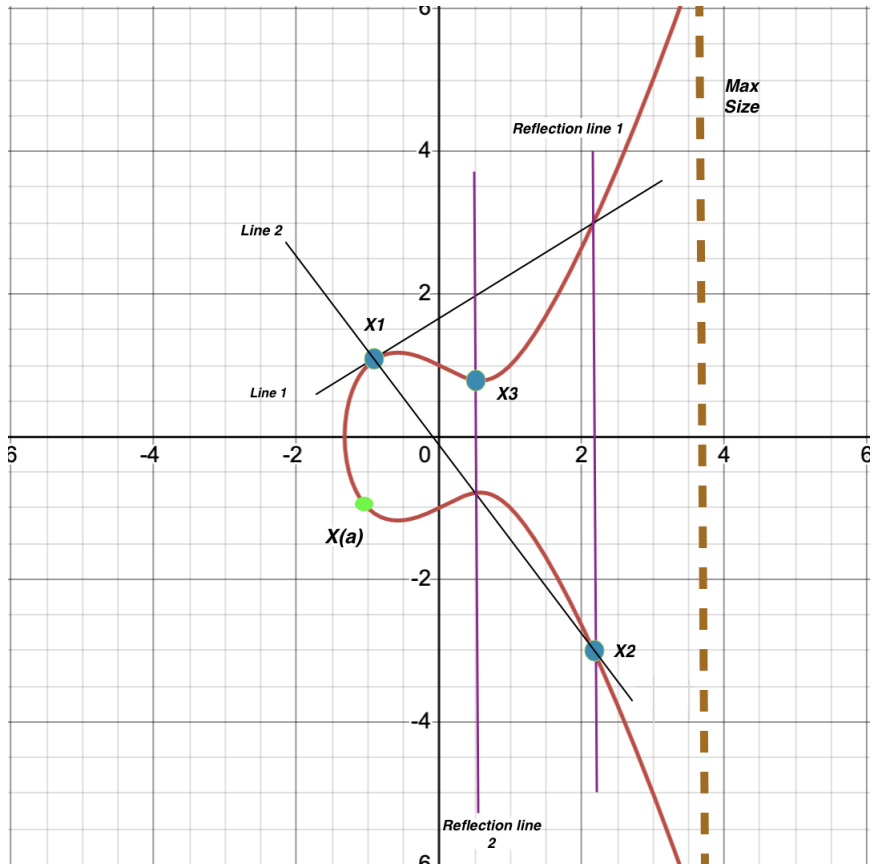

Figure 6.10.: Mapping the multiples of $X1$. Intersection between to points result in a temporary point X_{temp} on the curve drawn in black $Linex$, after that a reflection line is drawn vertically that maps the actual point X. The green point $X(A)$ is a private unknown point that is a multiple of $X1$. The dotted brown line is a max key size, that decides the difficulty of the algorithm (bigger is better, but it's slower).

Addition of points in elliptic curves work in a different way. Adding to point together to form a third requires more steps (e.g $X+Y=Z$, where X, Y, Z are points on the elliptic curve). We must draw a line between X and Y and find the first point that intersects with the elliptic curve, let's call it Z_{temp}. From that point, Z_{temp}, we must draw a straight vertical line and where it intersect again with the elliptic curve is the actual point Z. The hardness of the problem is that we can easily add these points, and after many additions we get a point somewhere on this curve, call it $K_{public}/X(a)$. We know the original point (depicted in fig. 6.10 as $X1$, the blue point) that started the whole process of mapping, but if we share only the $K_{public}/X(a)$ (depicted with a green point) that would be almost impossible to find a, how many times the point $X1$ got multiplied by itself (these are very big numbers fig. 6.13). The key size is defined by the allowed Ox axis distance (the brown dotted line in fig. 6.10), because as the X coordinate grows, the further away the curve goes(Oy

height). Modulo can be applied to make the problem harder, the group $E(k)$ can be written over modulo p like $E(Z/pZ)$, where $k = Z/pZ$ meaning the group is formed by integers over modulo p. Modulo is an amazing function that can greatly improve security, also seen in Lattice-based cryptography section 7.4. In fig. 6.10, a point $X1$ is multipled by itself multiple times, so $X2 = X + X$, $X3 = X * 3 = X + X + X$, etc. Addition between points on elliptic curves work in a different way, we must draw a tangent line from $X1$, $Line1$ with black, and find the first intersection with the elliptic curve. That is the intersection point, but we must reflect that point to find $X2$ using a straight line, "Reflection Line 1" drawn in purple, that is $X2$. Then for $X3$, we draw a line through X and $X2$, $Line2$, and that result is the intersection point of $X3$, after drawing the $ReflectionLine2$ we get the actual position of $X3$. It is assured that a line drawn between these points will always intersect with the curve in only one point, after that a reflection line is drawn and where it collides with the curve again that is the actual point. In blue are the X_n points. Using black is the tangent/connection line between points. In purple are the Reflection lines. The green point $X(a)$ and $X1$ are the "public key", these points are given publicly, and the "private key" is the number a, which is how many times the initial point $X1$ was multiplied by itself, similar to RSA where the public key is shared but not the private one. The brown dotted line is the $Maxkeysize$, because further on the Ox axis, the higher the curve is allowed to go on Oy, so bigger numbers.

Isogeny is a morphism between elliptic curves. It creates a rational morphism (operation) that can map points from one elliptic curve to another and also maps the point at infinity for both. This is similar to RSA by sharing the public key and each party has a private key, but they need a common ground to calculate the keys that they share (using the exponential function and prime numbers). The following figure shows how one party calculates a public key using the original point with a private variable, and the other a similar point, but in the end they combine. This is possible thanks to isogeny, because they share a function that can map points between two different curves, otherwise the final point may not exist on one curve or the other.

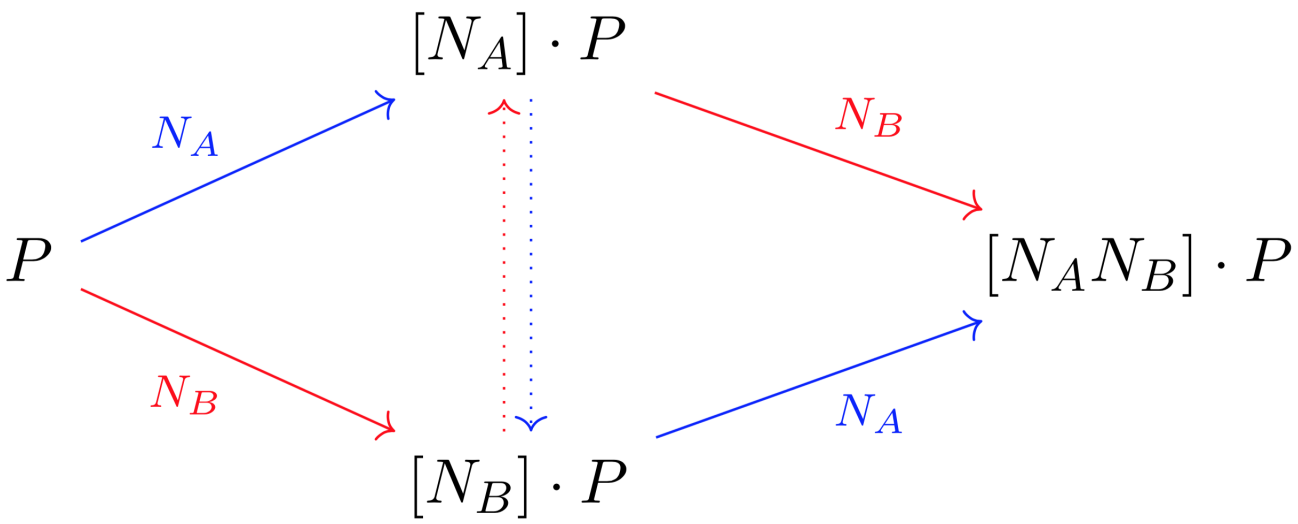

Figure 6.11.: Elliptic Curve Diffie-Hellman (ECDH). Scalar multiplication can be thought of as a function between elliptic curves. One party calculates the arrows in red, and the other party calculates the arrows in blue (almost identical to RSA). The dotted lines represent the parties exchanging the points $[NA]P$ and $[NB]P$. [60].

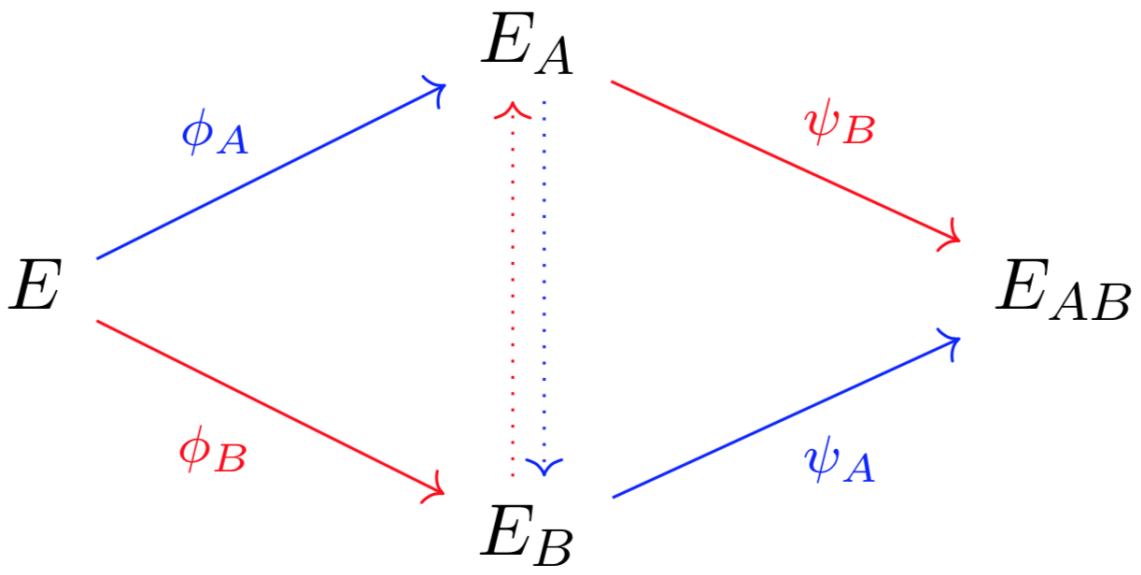

Figure 6.12.: Supersingular Isogeny Diffie-Hellman (SIDH). Both parties start with the same elliptic curve E. Each party chooses a secret number, call one rA and the other rB. One party creates a function A from rA and calculates the image curve $EA = A(E)$. The other party creates a function B from rB and calculates the image curve $EB = B(E)$. [60].

Thanks to the small key sizes that achieve the same security as RSA, it reduces the memory as well as the speed at which the keys are generated.

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

Figure 6.13.: NIST recommendations on the key sizes for ECC vs RSA. ECC uses smaller key sizes than RSA and obtains the same level of security [61].

A big observation is that the SIDH/SIKE family was broken in 2022 as discussed in chapter 3, although the attack is specific to this algorithm and does not mean that the whole isogeny-based cyrptosystem is broken. This family still relies on prime numbers, which was already established that post quantum computers are very good at guessing.

6.5. Symmetric key quantum resistance

Another cryptoscheme that is proposed is the symmetric key which was already discussed in AES 5.3. This system is still quantum safe if big enough keys are used, e.g AES-256, because of the time it would take to find the key. Key management system and protocols like Kerberos, which is an authentication protocol between internet nodes (like clients, servers, etc) and uses now aes256: "Beginning with the krb5-1.21 release, the KDC will not issue tickets with triple-DES or RC4 session keys [...] To facilitate the negotiation of session keys, the KDC will assume that all services can handle aes256-sha1 session keys [...]." [62]. Also 3GPP Mobile Network Authentication Structure is also inherently secure against attacks by a quantum computer because it is built on symmetric key cryptography.

6.6. NIST Competition for PQC

Overall the most important conclusion is to decide which is the most secure schema out of all these proposed families, but this also presents a problem. All of them have different implementations, size, purpose (signature or data encryption). That is why NIST created a competition to discover the most secure algorithm to promote security in the future. "The National Institute of Standards and Technology (NIST) was founded in 1901 and is now part of the U.S. Department of Commerce. [...] The mission is to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life"[63].

NIST also held rounds for traditional cryptography, where AES was suggested and implemented as one of the best cryptographic systems and as a standard. As of writing this thesis, currently its the fourth round, and there are still submissions sent to be reviewed[64]. The progress of the rounds can be found here: [65], where it can be seen that for each round there were a lot of Lattice based algorithms submitted, especially for the first round. There were a lot of submissions for Code-based algorithms also in first round.

Update: While writing this thesis, the finalists were chosen and the standards were set[3]. The Secretary of Commerce has approved three Federal Information Processing Standards (FIPS) for post-quantum cryptography: FIPS 203, Module-Lattice-Based Key-Encapsulation Mechanism Standard; FIPS 204, Module-

Lattice-Based Digital Signature Standard; FIPS 205, Stateless Hash-Based Digital Signature Standard. Two out of the three chosen standards are lattice based, which was the purpose of this thesis: to present an overview of all the families and observe lattice family resilience and overall capability to be sustainable against post quantum computing and traditional computing as well (see section 7.5).

FIPS 205, Stateless Hash-Based Digital Signature Standard is derived from the SPHINCS+ submission. This was discussed in section 6.2; it uses hash functions to validate other signatures fast, also uses one time signatures Winternitz type one-time signature scheme(WOTS+). Andreas Hülsing paper about "WOTS+ – Shorter Signatures for Hash-Based Signature Schemes" focuses on the security feature and provides proof of this unbreakable scheme[66].

FIPS 205 - SPHINCS+ (Hash-based) and FIPS 204 - CRYSTALS-DILITHIUM (Lattice-based) both are used for digital signature, and FIPS 203 - CRYSTALS-KYBER (lattice-based) is used for key encapsulation.

7. Lattice Space LWE

Lattice Space or a lattice-based algorithm is rooted in the hard problem of learning with errors (LWE)[67]. Learning with errors is widely used for creating strong cryptosystems and is based on sending secret information as a set of equations by masking the data using noise/errors. The LWE problem was introduced by Oded Regev in 2005 (who won the 2018 Gödel Prize for this work). Another variation is learning with errors over rings (RLWE) and is based on polynomial rings over finite fields [67]. By taking a simple polynomial $a(x)$ such as $a(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-2}x^{n-2} + a_{n-1}x^{n-1}$, using addition and multiplication creates an infinite polynomial ring $F_q[x]$ (meaning infinite possible functions).

The Lattice problem[67] is a mathematical problem in a vectorial space. This space is created using two base vector and mapping the infinite values that they create by addition and subtraction of these two vectors. It has the property that by adding or subtracting two random points created by these base vectors, it will result in a third point also mapped by the same base vectors. This problem is considered NP-hard with an average-case scenario to be hard and even worst-case hard. Because of the difficulty it poses, it is considered to be secure.

By combining these two problems, lattice problem and LWE, it becomes a very effective cryptosystem against computer's attacks, including quantum computers. Adding errors obfuscates the result and only the one having the secret can determine the real solution. Using lattice problem which is already considered NP-hard to find a point can make it more secure by adding noise.

Shortest Vector Problem(SVP) and Closest Vector Problem(CVP) are both classified as NP-hard and are based on the lattice space. Both these problems are more effective using errors, and even thought they are similar, they are solved in different ways. This is discussed in the next section section 7.1 where GGH (Goldreich-Goldwasser-Halevi) is analyzed, which was already broken, but it provides the basis of lattice space.

7.1. Lattice cryptofamily

Lattice cryptofamily is constructed on the basis of lattice space problem, a multidimensional space that maps vectors generated using the base vectors. The most basic lattice space can be represented in two dimensions, higher dimensions results in a bigger complexity as well as bigger key size. Two dimensional lattice space will be used for representation. The lattice space is created using two base vectors, which are close to or completely perpendicular in relation to each other, both starting from the origin. Based on this initial assumption, the space can be populated with the resulting points from addition and subtraction of these two vectors. This results in a space filled by infinitely many points, which is computationally impossible to map out. A section of the graph can be taken to deduce the pattern that it is forming. Because of the perpendicularity of these vectors, the space is evenly filled. The next point that is mapped to the space can be easily predicted. This is called the good basis or private vectors, and is the private key because it is used as the main "function or base". What is being sent to the client, is the bad basis or public basis, which is derived from the good lattice space (created by the good vectors) by taking two random points that start from the origin, and having the opposite characteristic, by being almost parallel. Actually because the bad basis is derived from the good one, similar points can be found in the space, mapped in the same locationfig. 7.7. This presents a very hard challenge to find the solution of the lattice space, and the bigger the vector as well as the angle, the harder it is to solve. There are two problems that offer the security of the lattice space and why is categorized as NP-hard.

Closest vector problem (CVP) is the main problem presented in the lattice graph. Finding the shortest non zero vectors combination using the base vectors to a random point in the group of points. Using the preferable basis that initiated the process, this point can be easily found, but using the bad vectors can pose a challenge. Facilitating learning with errors provides even more obfuscation to the problem. An error is added to the picked point so it is not an exact point that is part of the lattice space, it is off position, and finding an answer might not be the right one, even if it seems close enough.

Similar to CVP problem, there is **Shortest Vector Path (SVP)** which is based on the same assumption, with the correction that the point is always the origin, and finding the shortest non zero vectors combination is now bound by the closest point to the origin, which again is a very hard task to be found using a bad basis. The closest vector problem is a generalization of the shortest vector problem, so they both share the same hardness and security reduction.

7.2. Creation of the lattice space

Python was used to represent graphically a vectorial space. There are different programming methods to accomplish this: recursive functions, dynamic programming, backtracking, machine learning (which can be trained to generate the points and exact location of the solution, but this would be similar to creating a good algorithm to solving the problem, because it is a math problem in this case, so you would have to create a formula or let the machine train to find one), graphs and so on. The libraries used for the following snippets of code are:

- **Matplotlib**: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.[68]
- **Numpy**: Provides different array capabilities to work easier with large data such as: powerful N-dimensional arrays, numerical computing tools.[69]

Note: Small vectors were chosen to facilitate the learning and research purpose of the thesis, and a better understanding of the process. Also, an old and broken version of lattice space was used Goldreich-Goldwasser-Halevi (GGH) which was published in 1997 [70]. A lot can be learned and understood from this version even though it is broken and how the lattice space had to evolve to combat being vulnerable to different attacks. The new and improved versions are LWE and RLWE, which also the finalists use.

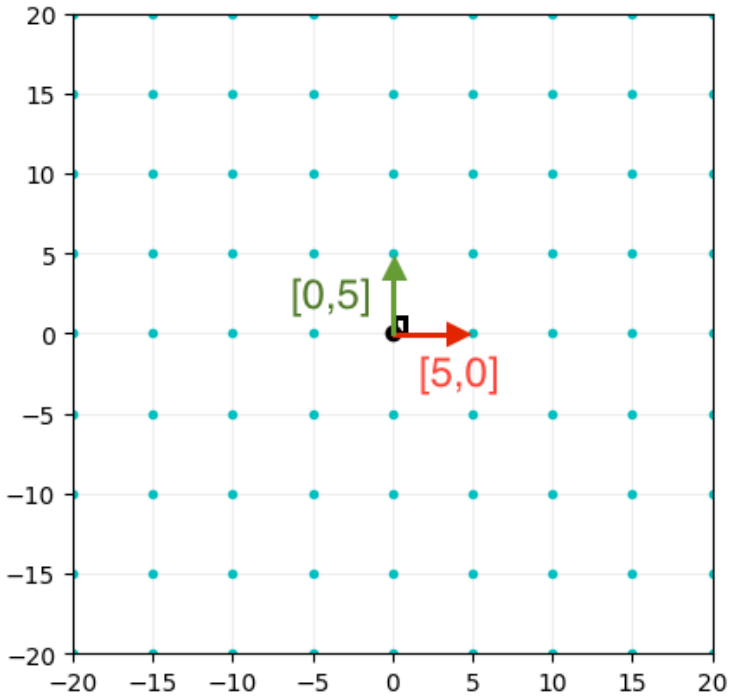

Figure 7.1.: Representation of a good basis lattice. The red vector is mapped to Ox axis with vector value $[5, 0]$, and the green one is mapped to Oy with values $[0, 5]$, which are forming a 90 degree angle. The points are mapped in a repetitive way and the next point can be predicted because of the simple vectors. Based on this representation, a bad basis can be generated picking two random points that are close to parallel like in fig. 7.2.

Good basis have to be almost perpendicular, does not necessary have to be perfectly perpendicular. In this example, because of the vectors being perfectly perpendicular, the mapped points form a predictable pattern (I have observed that all lattice spaces form a predictable pattern, but the answer is not so easy to compute because of added errors). Each vector is being mapped to one axis ($[5, 0]$ is mapped to Ox and $[0, 5]$ on Oy), computing any point in this group can be done using division. Taking the point $[125, 55]$ using the lattice in fig. 7.1, dividing each point by 5 will result the combination of vectors, in this case $125/5 = 25$ and $55/5 = 11$, denoting the result as a tuple $(25, 11)$ where the first value is the count of base vector $[5, 0]$, respectively $[0, 5]$. These simple vectors $[5, 0]$ and $[0, 5]$ (which can also be written as $5i + 0j$, and $0i + 5j$ respectively, where i represents the Ox axis and j the Oy axis, also called Cartesian coordinate system) translates its predictable nature to the bad basis and becomes more strongly accentuated as it can be seen in the figure fig. 7.2. Using the points in fig. 7.1 and picking randomly two of them, almost parallel, a bad basis can be derived. Even though a lattice has a bad basis, a pattern still seems to be forming, observed

in the following figure fig. 7.2. **Note: For the following graphs, there are missing represented points. The reason is also explained in the code section section 7.3 and it is because of computational power and time reduction, but the identified pattern can help predict the rest of the lattice points without computing each point.**

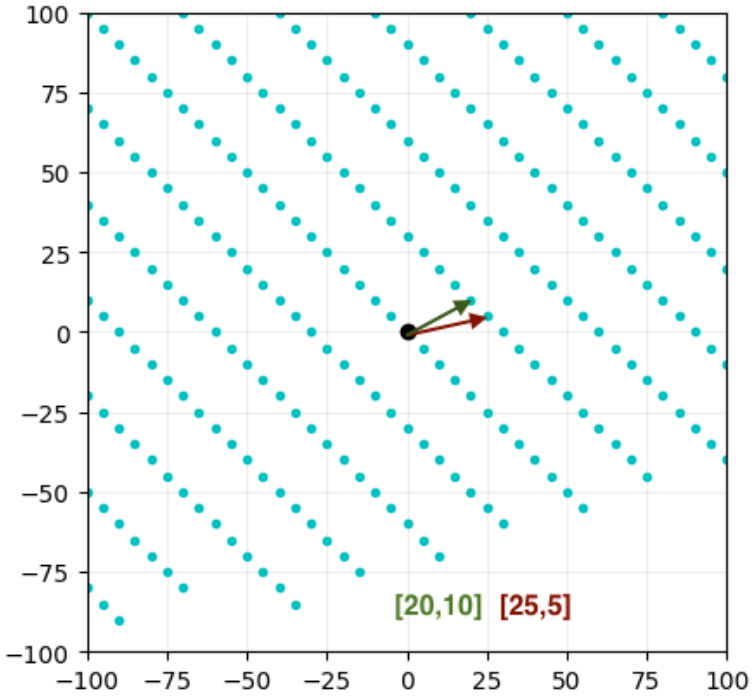

Figure 7.2.: The "bad" lattice, with vectors from previous figure fig. 7.1. The vectors $red = [25, 5]$ and $green = [20, 10]$ gives a predictable pattern of diagonal lines, which seems again to be evenly spaced and not much randomness.

Using another base vectors $[3, 0]$ and $[1, 3]$, and choosing the bad basis as $[7, 3]$ and $[11, 6]$ this presents a bigger problem, because of the added dimension on Ox for $[1, 3]$. It is possible to compute a lattice space having opposite vectors, not only in the same Quadrant, so the following figure fig. 7.5 has the bad basis $[-7, -3]$ and $[11, 6]$ computes the same space as fig. 7.4. Parallel vectors can be also in opposite directions not only in the same one.

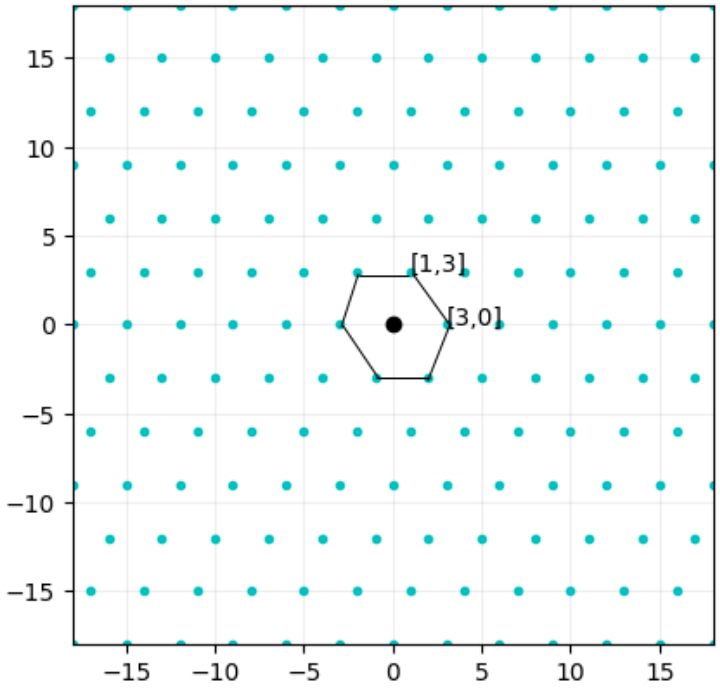

Figure 7.3.: Good basis with vectors $[3, 0]$ and $[1, 3]$, the vectors are not forming exactly $90degrees$, again a similar pattern to fig. 7.1. A hexagon like formation can be observed around all the points in the lattice, even using bad vectors.

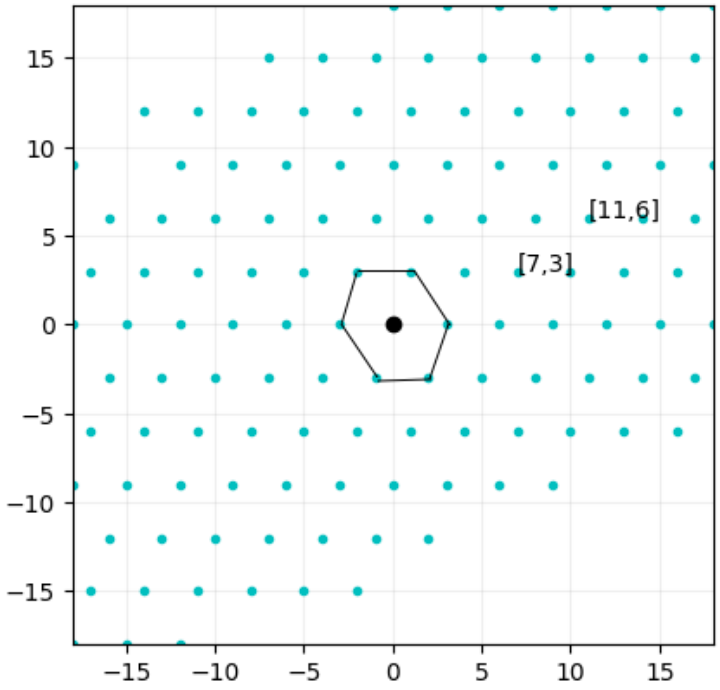

Figure 7.4.: The following graph share similar features to fig. 7.5, both have similar bad basis $[7, 3]$ and $[11, 6]$, with the exception that in fig. 7.5 the vector $[7, 3]$ is negative. Because the possibility of subtraction in lattice space as seen in fig. 7.6, the vectorial space is the same to fig. 7.5, same points with the same values, only the construction (combination of vectors) of the points are different.

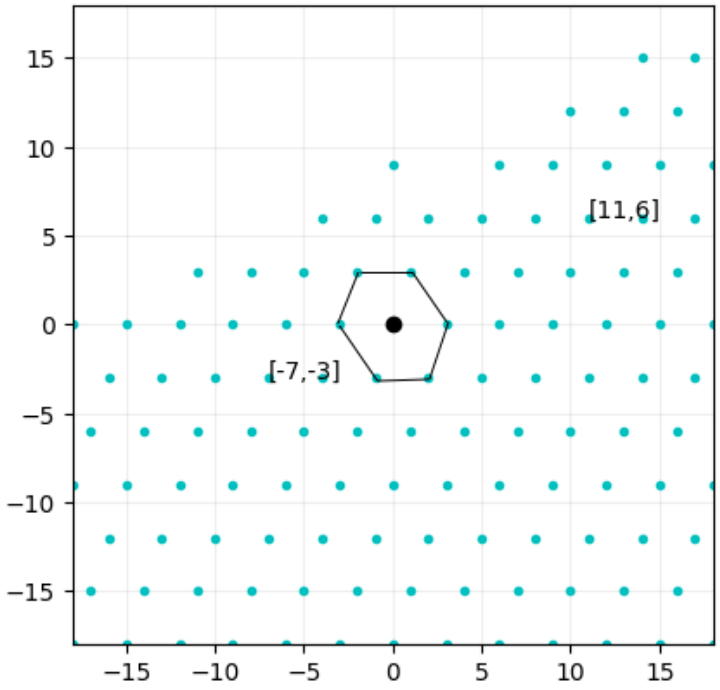

Figure 7.5.: This graph and figure fig. 7.4 share almost the same base vectors with the exception that $[-7, -3]$ is negative. Even though this vector is negative, the group of points is the same.

An interesting fact that can be noted is that **eventually all lattice spaces will become a pattern**, however hard the basis is. This is because there are static inputs and is very predictable. Any point can be written as a function $f(a, b) = a * v_1 + b * v_2$, where x and y are integers, and v_1, v_2 are the base vectors. The formula can map any point (in this case in two dimensions), and it doesn't have variables that may affect the pattern of points (like a random added value for each iteration, that would change the pattern). If there would be something like a variable C that is added to each point and depending on a, b and a secret key would change, then it would be impossible to predict a pattern without knowing the secret. It is important outline this, even by respecting all the rules when choosing the vectors almost parallel, a bad basis can still be predictable and might not present a big resistance for an attacker.

Another observation is that the nature of constructing a lattice allows, by using subtraction, absolute values of the third quadrant to be the same as in the first quadrant This property derives from the possibility of writing the vector in a negative form and add the vectors to create the lattice space (refer to fig. 7.4 and fig. 7.5). As an example, taking the vectors from figure fig. 7.2, $red = [25, 5]$ and $green = [20, 10]$, using subtraction $-[x, y]$ which equals $[-x, -y]$, both are able to be subtracted to create the lattice space and taken as $-[25, 5] = [-25, -5]$ and $-[20, 10] = [-20, -10]$. Having this negative vectors simple

addition can be done on them and will form a mirror image of Quadrant I. The following figure fig. 7.6 is a reference for a better understanding, the red plus points were not computed using the code, but by multiplying the points in first Quadrant by -1. This is an optimization for computing less information. It is easier to multiple already known points by -1 than to compute new points. This can be useful to map out a lattice for a problem, but in cryptography this is a threat.

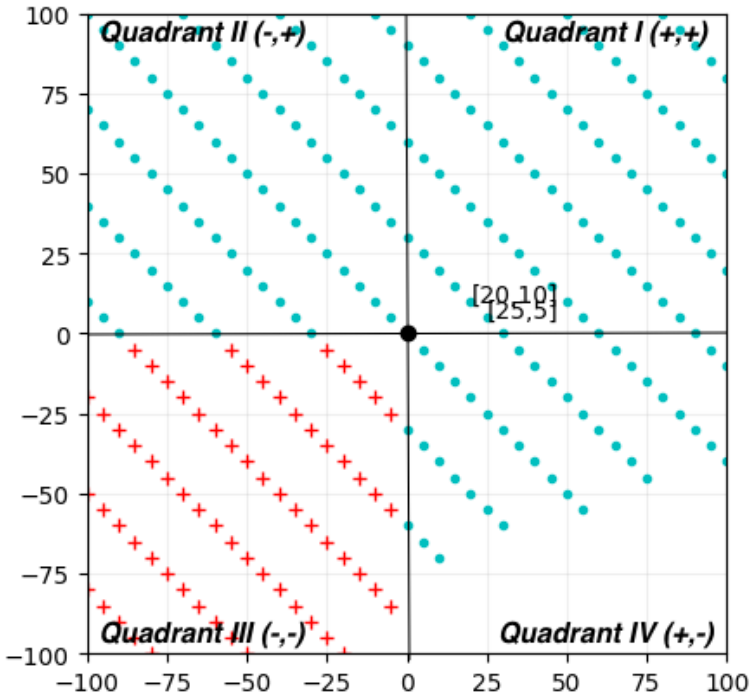

Figure 7.6.: Again the vectors $[25, 5]$ and $[20, 10]$ gives a predictable pattern of diagonal lines, which seems to be evenly spaced and not much randomness. The $redplus$ represents the points in the first Quadrant multiplied by -1. Comparing this figure with fig. 7.2 there is no difference.

For the figures fig. 7.4, fig. 7.5, fig. 7.6, the graphs are the same because of the property discussed earlier about the Quadrants. To represent the lattice, addition and subtraction of the vectors is needed. If a vector is all negative, by using subtraction to map the points, it becomes positive: $-1*[-7, -3] = [7, 3]$ (subtraction), which is also the same vector used in fig. 7.4. By using subtraction on a negative vector it is the same as addition on a positive one. That is why is possible to represent the same points using two opposing vectors that have the same absolute values. Also another pattern that can be seen in the figures are the six points around the origin. These six points that form a hexagon around a center point, will form a pattern around each other point in the lattice. Is not a rule to share this pattern with the good basis (take as example fig. 7.1 and fig. 7.2), although it is pretty common to see this pattern in the structure of the lattice. This is just a small

observation, that can also be seen more clearly in the following Voronoi diagrams. This is an interesting finding because patterns are bad in cryptography, this is how ciphers are broken.

Another important observation, is that even though fig. 7.4 and fig. 7.5 uses the bad basis, the good vectors can be found in the structure of the lattice fig. 7.7. It can also be seen that the absolute difference between two nearby points returns a good basis vector, even though the bad basis were used. Refer to fig. 7.7, bad basis $[7, 3]$ and $[11, 6]$ were used. The nearby point to $[7, 3]$ is $[10, 3]$, absolute value from subtracting these two vectors is $[3, 0]$. Another close point is $[8, 6]$, subtracting with $[7, 3]$ results in $[1, 3]$. These two resulting vectors are the good basis of this lattice (this in the case presented, not many tests were done to confirm, but even if it is present in one case, it is a nonoptimal crypto solution). Similar points are present after all because the bad vectors are just points that can be found in the good basis, the bad lattice's points are a subset of the good basis's points, at least in this case (not enough tests were conducted to prove this, but again because it was found even in one case it is suboptimal). Finding the good basis (meaning the key) inside the formula (in this case the graph of points/lattice space) that is supposed to hide it, is horrible.

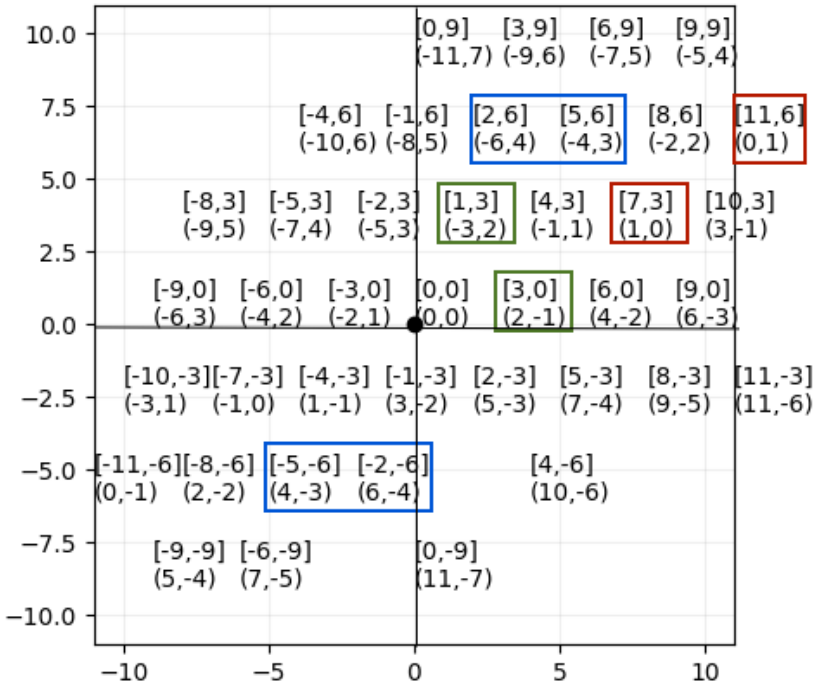

Figure 7.7.: In this figure, with square brackets is the position of the point $[x, y]$, and below each point its (a, b), the combination of the base vectors (v_1, v_2), $f(a, b) = a * v_1 + b * v_2$, needed to map each respective point. Red squares are the bad basis vectors with which this lattice was constructed. In green are the good basis vectors of the lattice, which are found in the lattice constructed using the public vectors. In blue is depicted the similar vectors in first and third Quadrant, as well as the similarity of the construction vectors.

In fig. 7.7, marked in red are the bad vectors that were used for constructing this space. Despite using the bad vectors, marked in green are the good vectors which can be observed as being part of this public lattice. Marking in blue, two randomly picked points (similar for any chosen point) from the third quadrant, which absolute values can also be observed in first quadrant. That is not where the similarities end, also the negative values of the constructing vectors (the bad vectors) can be found in third quadrant, again referencing the substitution property.

7.3. Algorithms for lattice space

The observations described in the previous section made me wonder how they can be used to map and solve the problem lattice is proposing. Humans are always looking for patterns in nature and in mathematics (e.g. fractals, golden ratio, etc). I wanted to pursue this thought further and see what can be achieved

using programming. Dynamic programming is a method of programming based on using the previously discovered answer to aid in computing the new value. This can help reduce the time of processing a lot of data with the tradeoff using more memory. Nowadays this is not a problem anymore, there is plenty of memory in every computer, but time is not something there is plenty of. For example Fibonacci sequence can be done using recursive functions, but it can be more efficient using dynamic programming. Recursive method is used to explore multiple possibilities using data passed through the same function and multiple calls. This is very good for backtracking, for example the Rat in a maze problem[71]. Multiple paths are possible, and if there is a dead end you need to backtrack to the last position. This information is kept in the call stack and is available only through the logic of the recursive calls. For this method you have to call the same function from within, so it forms a loop with the only change being the parameter passed to it. An ending condition must be set for the function otherwise it will call itself to infinity. Biggest drawback of recursive programming is that even though the information is known, it will be computed again.

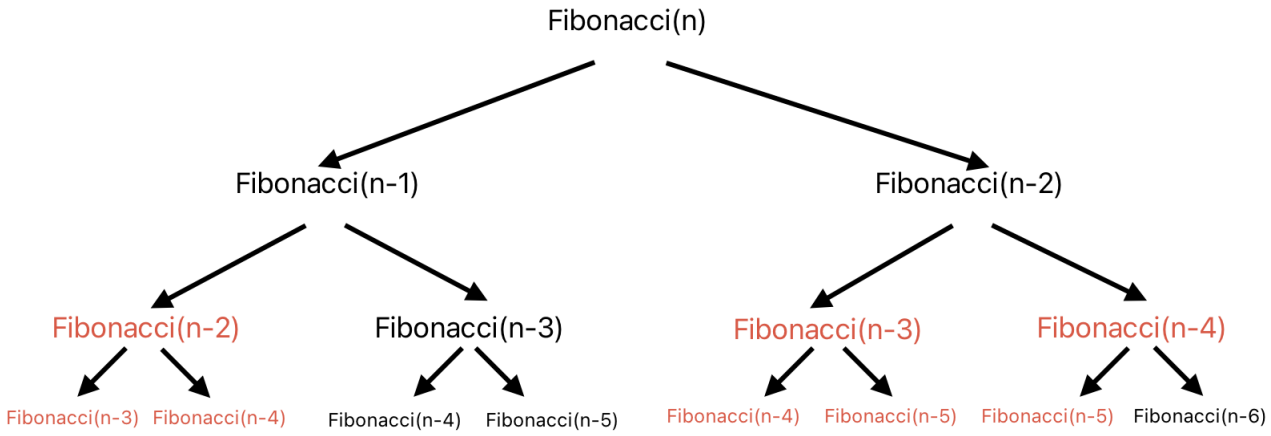

Figure 7.8.: Representation of Fibonacci sequence using recursive method. Still a viable option but provides more complexity than dynamic programming, in terms of big O notation results in $O(2^n)$ time and $O(n)$ space complexity. Recursive method provides a tree-like structure in terms of processing, with each subsequent node repeating. Red denotes the nodes that were already computed in another call, but were still computed again because of the recursive nature to call the same function, without knowledge of other data already computed.

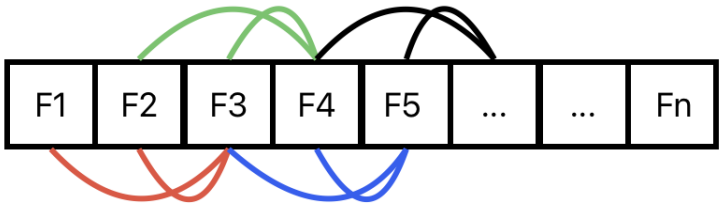

Figure 7.9.: Dynamic programming provides a simple solution to the problem described in fig. 7.8, by storing the data and giving the function knowledge of the already computed values. Now the time complexity is $O(n)$ with the memory still remaining $O(n)$. Even though the result is the same, the time it saves matters a lot for these computations, especially if the values are big.

This small difference in the approach of solving the problem yields so much optimization in time (reference fig. 5.6 for a better visualization of the time complexity). In this case dynamic programming is better, but for the rat in a maze problem, the recursive method provides a better solution, the approach really depends on the situation. In the case of lattice space, recursive method seemed to be the best option in the beginning, but after more testing, dynamic programming might also be a good option for optimizing the code. I decided to make a hybrid code between the two, using recursive call for addition and subtraction of base vectors depending on the current point, and storing the points for later use and optimization(like computing the third quadrant points with relation to the first quadrant, explained in the section 7.1).

7.3.1. Lattice Generation

In this subsection, the code for creating the lattice spaces so far will be described. Reference appendix code where each line is commented and explained briefly appendix C. The libraries used were: numpy and matplotlib.

The code used was done using recursive functions with some storing of data for efficiency. Without storing the points, same ones would be calculated, it would waste time and memory. Although computing each point would provide a more complete view compared to fig. 7.10, where it can be seen that it is more one directional if zoomed out, this would significantly increase the computational time (this can actually be optimized by bounding the points to a square, like it is done in figure fig. 7.1 where the data is cropped). The direction depends on the vectors (because of how the code was written, this is not a rule for the lattice), if they are positive or negative, and we are mapping first the direction they are pointing to. Another case depends on the order of the recursive function(in this code, addition is first, that is why it goes in the positive first Quadrant direction, but subtraction can be set first). Further research can be done for using

only dynamic programming, and using the already computed points in the array. This would eliminated the recursive problem of going back to the same point and computing another calculated point from that position again(I tried optimizing it, by marking the points that already have been computed. The code returns and does not call the same function for its neighbors).

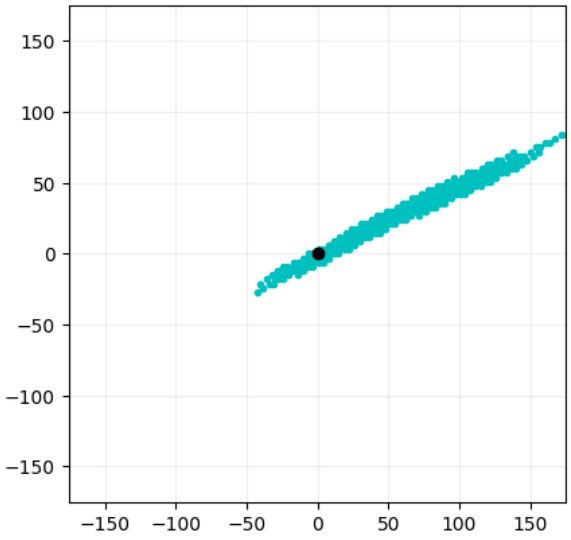

Figure 7.10.: The code used produces a more directional pattern if zoomed out. This is because of the optimizations done, while reducing the number of points, the time of computation was decreased, this takes a few seconds to compute, compared to a non-optimized algorithm where the lattice points are not stored.

Because of the patterns observed, storing this data and using the difference between points, a rule can be found for the points and the combination of vectors to get to that random point that needs to be found for SVP or CVP. Being able to find this rules could mean to find any point in the lattice instantly. But SVP and CVP is not used only by choosing a point, an error is added to that point to harden the task of finding it, as well as the combination of vectors. A Voronoi diagram is a partition of a plane into regions close to each of a given set of objects [72]. This gives the closest region to a point without interfering with another. This is great for the purpose of finding a CVP as the error point to be found has to be in the region of a lattice point. Voronoi diagram can be used for any point, even if they form a pattern or not, to map the region that is closest to a point, but there are also other methods like the distance between two points, that gives an approximation of what Voronoi is exact at. Figure fig. 7.11 shows the area where each point is covered by a Voronoi diagram. This area around each point is called a Voronoi cell. It produces a hexagonal pattern, which is similar to fig. 7.3, reproduced for each point. This is because of the pattern lattice space is forming,

and it can be detected easily using Voronoi diagram.

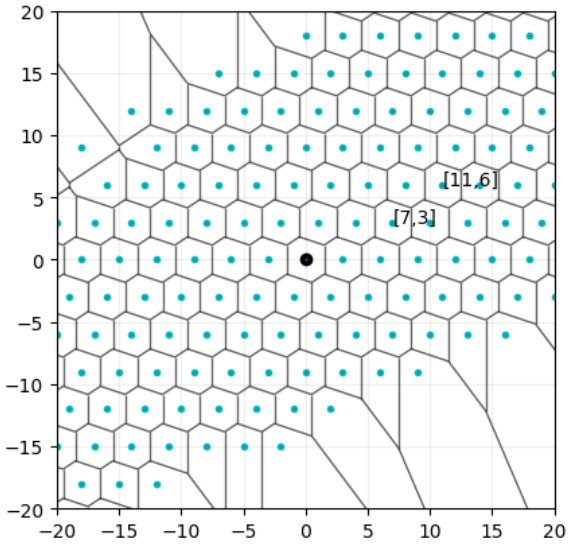

Figure 7.11.: The same public base as before, but using a Voronoi diagram on top, it can detect the region of each point(called Voronoi cell), and especially the pattern that each point is forming. This is because of the nature of lattice space. The extended cells are caused by the missing neighbor points which were not computed for optimization, so they cover rest of the graph to an infinite extent(they can be ignored).

7.3.2. SVP and CVP

Shortest Vector Path and Closest Vector Path are the problems that are used in lattice space and which provides its difficulty. In the case of CVP a point must be chosen from the infinite group of points. An error is added to the chosen point formed by the good base(the points from the good base can be found in the bad lattice) and shared. The recepient, trying to decrypt the message, receives only the bad basis and the desired point plus the error, and must find the right point as well as the combination of these vectors in order to decrypt the message. The SVP is similar, but the resulting point (or key point) is the origin of the lattice ($[0, ..., 0]$), and again the combination of vectors and the point closest to it must be found. For this task, I used the difference between two point to determine the closest point in the lattice closest to the key point. After creating the lattice and storing the points, the code goes through all the points that were created and determining the smallest distance to the error point using the following formula:
$d(p1, p2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, written in python as:

```
def distanceBetweenTwoPoints(point1, point2):
    return
        math.sqrt(math.pow(point2[0]-point1[0],2)+math.pow(point2[1]-point1[1],2))
```

Listing 7.1: Pyhton function for computing the distance between two vectorial points. Can be extended to any size of dimensions N. Here it is used for two dimensions.

For the first test, the code or attacker would only know the "Lattice vectors" and "Result point"+"Error", denoted as "Error point". The purpose of the code is to find the "Result point"/"Result combination" to be able to decrypt the data sent. "Error Point" is mapped in figure fig. 7.12 using a red plus. The predicted point by the algorithm and the "Result point" are the same. In fig. 7.13, I used Voronoi's schema to map the area of inclusion that a point in the lattice has, also for the subsequent figures. The point is actually inside the area of Voronoi's mapping which confirms the point is the right one. Voronoi schema helps confirm the answer and could also help discover the "Result Point" by itself. Searching the area of each point and checking if the "Error Point" is inside the respective point could also be a solution.

```
Lattice vectors: [7, 3] [11, 6]
Result combination: [1, 2]
Result point: [29, 15]
Error: [-0.5, -1]
```

Listing 7.2: These are the inputs and results for the figure fig. 7.12. Lattice vectors are the bad vectors use to create the public lattice. They are used to find the "Error Point" derived from the good base. "Result combination" is the predicted "Result point" found by the program, with matches with the result chosen point. "Error" shows the error added to "Result point" to make it harder to find.

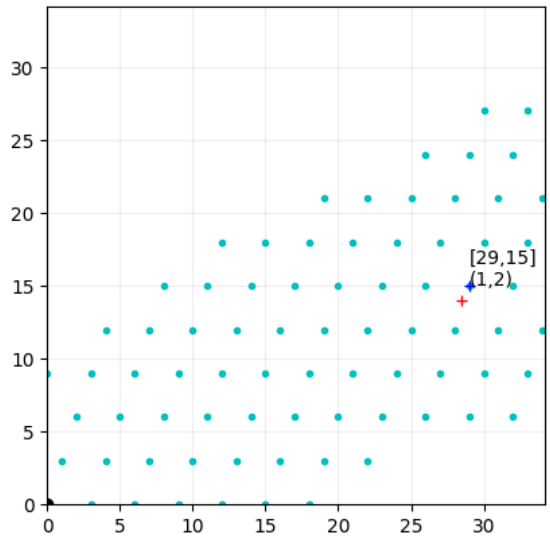

Figure 7.12.: Finding the "Result combination", using the distance between two points formula. This is done by passing through all the saved points and comparing the distance to each one, saving the shortest distance's combination. It can be seen that the predicted combination and point are the same as the result. The red plus denotes the "Error Point", and this is the input for the code. Although the code can see only the point with error, it can detect which point is the closes to it and choose it as the "Result Point".

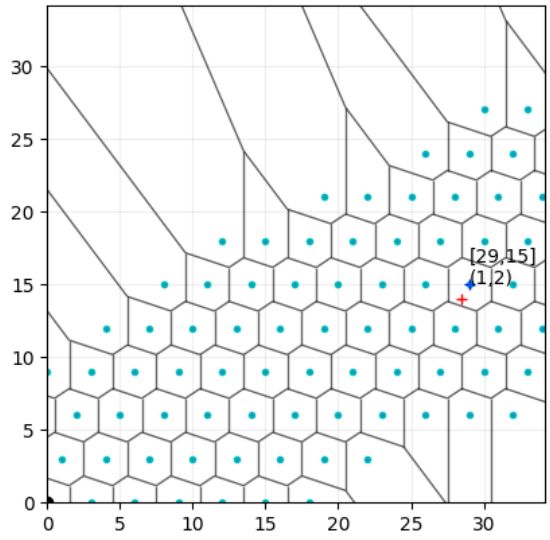

Figure 7.13.: Here the same algorithm is used as in figure fig. 7.12, and it corresponds with the Voronoi diagram, the error point is in the area of the predicted point, concluding that the prediction is right.

In the following figure fig. 7.14, the algorithm was tested using a big error, and even though the starting "Result point" and "Result combination" were the one from fig. 7.13, it changed the area and the "Result point" becomes a different one. This can affect the decryption as it damages the location of the real "Result point". Sometimes the error can be too big, if the error results in the "Error point" being outside of the area of the initial proposed "Resulting point" like in the figure fig. 7.14. This can result in an entirely different point. This affects the performance negatively because decryption is essential not only protecting against attackers, but also for the authentic recipient that should be able to decrypt it. The biggest error that can be applied can be calculated by taking the smallest midpoint between the neighboring points (this would result in a circle), although this is not the most accurate(because a circle have spaces outside the area of a point). For the most accurate maximum error, Voronoi schema has to be used like it was done so far, and the point has to be inside (similar to how RSA is testing the numbers before using them).

```
Lattice vectors: [7, 3] [11, 6]
Result combination: [1, 2]
Result point: [29, 15]
Error: [-1, -7]
Predicted point: [27, 9]
Predicted combination: [7, 2]
```

Listing 7.3: Testing the algorithm on a big error, and it can be seen that the predicted point is not the right one, because the error pushed the point to another cell.

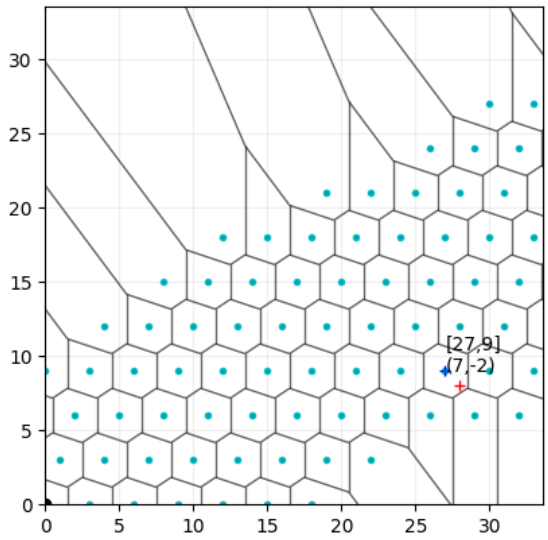

Figure 7.14.: The same "Results combination and point", but using a very big error that would affect the right answer. The predicted point is not the right one but that is because of the high error. Although if the error would be passed, the key point/"Result point" could be found. Again the lines that go to infinity are because of the missing points.

Shortest Vector Problem is very similar to Closest Vector Problem, instead of choosing a point arbitrary the resulting SVP point is the origin $[0, 0]$. Without an error added to the equation, in figure fig. 7.15 the point chosen by the algorithm to be the closest to origin is $[3, 0]$. As seen before in fig. 7.7 there is also $[-3, 0]$ in this lattice. Using the distance between two points formula, the distance would yield the same, so it cannot be determined which one is the right one. The code checks just the first time it finds the smallest distance, if it finds any other distance that is the same length as the one stored it doesn't overwrite the value. But the right point cannot be determined because we did not choose a point to be found, even from a human perspective. Actually adding an error here is more beneficial for finding the actual desired point, as seen in fig. 7.16, otherwise it cannot be determined the right "Resulting Point" because there are two points that could be it, $[-3, 0]$ and $[3, 0]$.

```
Lattice vectors: [7, 3] [11, 6]
Chosen point: [-3,0]
Result point: [0, 0]
Error: [0, 0]
Predicted point: [3,0]
```

Listing 7.4: but the desired resulting point is [-3,0]. Both points are equally distanced in regards to origin, so the algorithm can't know which one is right.]Results from SVP, the predicted point is [3,0] but the desired resulting point is [-3,0]. Both points are equally distanced in regards to origin, so the algorithm can't know which one is right.

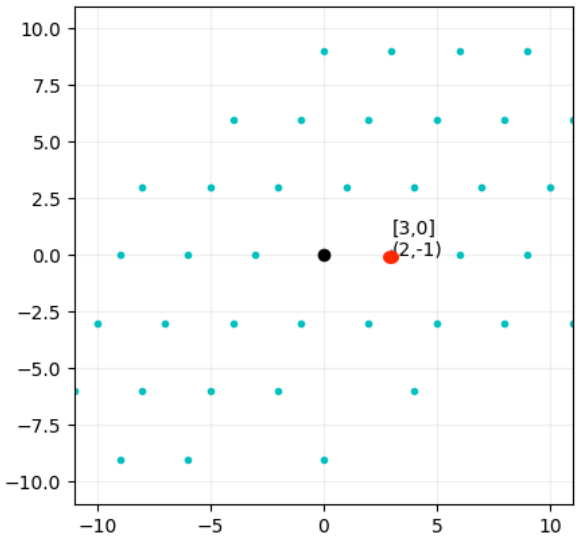

Figure 7.15.: SVP, where the origin is the point $[0, 0]$ and without an error, the predicted point can't be determined because there are two equally distanced points in regards to origin. The red dot represents the predicted point by the algorithm, which may or may not be the right "Result point" because of the equally distanced points from origin $[3, 0]$ and $[-3, 0]$.

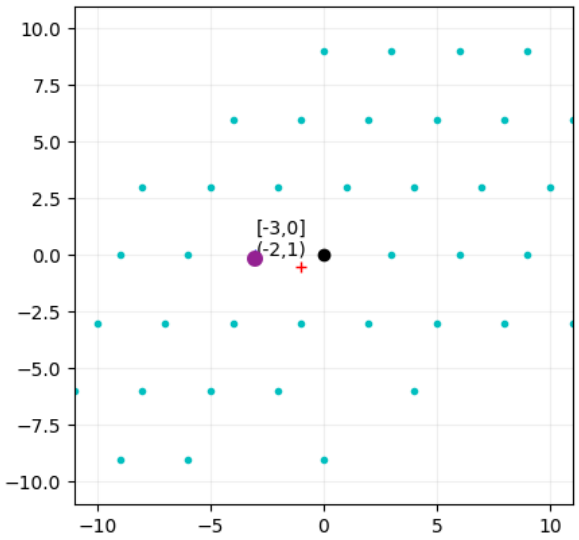

Figure 7.16.: Using a small error can be more deterministic for finding the right “Result point” and the “Predicted point” can be accurate. The purple point is the “Predicted point” as well as the resulting one, because the error point is in its area. Thanks to the error, the “Result point” can be found.

These were just some tests and a playground for lattice space. These representations and problems described are called Goldreich-Goldwasser-Halevi (GGH) which was compromised [70]. Bigger values would be chosen, more obfuscation, more dimensions and so on, but it is still important to understand the process that it stands on. Although GGH is broken, it still is the basis of the present Learning with errors and learning with errors over rings, they are consider to be part of the lattice space family also.

7.4. LWE/R

Lattice spaces [chapter 7] were the basis of lattice based algorithms, it started as mathematical problem with representation in a multi dimensional graph. Some flaws can be observed: repeatability, simplicity and scalability is limited by the number of dimensions. Although GGH is not offering the security everyone was hoping for, it allowed development of more secure algorithms like “Learning with errors”. Learning with errors (LWE) is a hard security reduction problem defined by Oded Regev in 2005 [73], and later awarded Gödel Prize for his idea, which is very effective in the future of cryptography. It is an evolution from the traditional lattice space described in section 7.1, by adding a random value to a simple equation one cannot find the right answer. Furthermore, modulus is applied on the result, this way no one can guess the answer, only by trying to brute force the answer to infinity. That is the beauty of modulus in cryptography, you

cannot know the right answer ($7mod2 = 9mod = 11mod2 = ... = 1$, the reminder of division by two). The grounds on which LWE stand are adding noise and modulus to the system of equations. Another variation of this problem is RLWE, which relays on the error being a polynomial using addition and multiplication. Taking the system below, where the solution is $S = [3, 8, 2]$ ($S = [x, y, z]$), error values can be added to the result and the equation would make no sense to an attacker, making it impossible to solve without the answer.

$$\begin{cases} 3x + 2y + 2z{-1} = 22mod7 = 0 \\ 7x - 4y + 6z{+0} = 1mod7 = 1 \\ -3x + 3y + 2z{+1} = 20mod7 = 6 \end{cases}$$

In red are displayed the errors that are added to the equation. Taking a closer look, this is already similar to the multivariate family section 6.1, but it takes another approach, which actually makes it harder to solve by adding the errors and modulo. section 6.1 relies mainly on the coefficients. Both LWE and section 6.1 can be written as a system of matrices. The first matrix represents the coefficients called **A**, the second one are the unknowns (denoted using **S** which is the private key/secret), the third one are the errors being **e**(the values are hidden from the public) and the result is written as **B** modulo.

$$\overbrace{\begin{bmatrix} 3 & 2 & 2 \\ 7 & -4 & 6 \\ -3 & 3 & 2 \end{bmatrix}}^{\text{A}} \times \overbrace{\begin{bmatrix} 3 \\ 8 \\ 2 \end{bmatrix}}^{\text{S}} + \overbrace{\begin{bmatrix} -1 \\ +0 \\ +1 \end{bmatrix}}^{\text{e}} = \overbrace{\begin{bmatrix} 0 \\ 1 \\ 6 \end{bmatrix}}^{\text{B modulo 7}}$$

This system would be denoted as $\mathbb{Z}_7^3$, where 7 is the modulus and 3 the number of unknowns and secret length. The only data that is passed for the public key are the coefficients matrix A and the result B, which would be passed as $K_{public} = \{([3, 2, 2], 0), ([7, -4, 6], 1), ([-3, 3, 2], 6)\}$ (the tuple is (vector A, "B" value of that vector)). The errors and secret values allows the decryption of the message, but without these variables there is almost no chance to finding the right combination of parameters. After securing the payload, using the half of the modulo number (in this case 7), it can deduced if the received bit is 0 or 1. Example: Half (rounded down) of 7 is 3, K_p must be calculated from the tuples. For $bit0$ => $([3, 2, 2], 0), ([7, -4, 6], 1)$ which would result in $Enc_0 = ([10, -2, 8], 1mod7 = 1)$. The half of modulo

7 rounded down must be added for the $bit1$ => $\{([3,2,2],0),([-3,3,2],6)\}$ the encryption function is $Enc_1 = ([0,5,4],0+6+[7/2]mod7 = 6+3mod7 = 2)$. Now for decryption the secret must be inserted, $Enc_0 = 10*3-2*8+8*2 = 30mod7 = 2$ and $Enc_1 = 0*3+5*8+4*2 = 48mod7 = 6$. Then the B value must be subtracted from this, $bit0$ is $1-2 = -1+7 = 6$ (B from here $Enc_0 = ([10,-2,8],1mod7 = 1(B))$, and the result after substituting the secret $Enc_0 = 10*3-2*8+8*2 = 30mod7 = 2$) and for $bit1$ is $2-6 = -4mod7 = 3$, in case of negative numbers the modulus must be added (in this case 7) . The resulting bit is calculated depending if it's applying an 0 (range $[6,7,0,1]$) the bit is 0 and closer to $[7/2]$ (range $[2,3,4,5]$) the bit is 1. After calculating the these numbers, depending on the result we determine if the answer is 0 or 1 by comparing to the range of modulo 7, for 0 (range $[6,7,0,1]$) the result must be closer to 0 and for bit 1 its closer to $[7/2]$ (range $[2,3,4,5]$).

Using this obfuscation method by applying noise and errors can greatly secure the system and make it almost impossible to decrypt. In reality these numbers are much bigger, equations are more complex, with multiple equations into one system, and is great because of the small size and its simplicity (compared to lets say elliptic curves or code-based). But for large blocks of data, say n-sized bits, the functions Enc_0 and Enc_1 must be called multiple times to for just a 16-bit data block. RLWE was adapted to solve this issue, because it can send set of bits with one equation.

LWE over rings(RLWE) goes one step further and adds the noise to the equations using polynomials. Using a generic polynomial $a(x)$ such as $a(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-2}x^{n-2} + a_{n-1}x^{n-1}$, there is an infinite possibility of resulting polynomials by addition (subtraction is implicit using negative values) and multiplication (division is implicit by using $\frac{1}{a(x)}$, where $a(x)$ is a polynomial). As an example taking $a(x) = -5+2x+4x^2$ and adding the polynomial $e(x) = 10-8x+6x^2-x^3$ (error) results in $a(x) = 5-6x+10x^2-x^3$ which becomes unrecognizable from the original function. The first difference that can be seen is that RLWE is using powers of x instead of multiple unknown variables. This causes the secret s as well as the error e to change from a group of unknowns to a polynomial. So now secret s is also of type $s(x) = s_0 + s_1x + s_2x^2 + \ldots + s_{n-2}x^{n-2} + s_{n-1}x^{n-1}$ which gets multiplies with every polynomial in the system.

$$\overbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \ddots & & \\ & & & \\ a_{n1} & \dots & & a_{nn} \end{bmatrix}}^{\text{A}} \times \overbrace{\begin{bmatrix} s_1 \\ \vdots \\ \\ s_n \end{bmatrix}}^{\text{s}} + \overbrace{\begin{bmatrix} e_1(x) \\ \vdots \\ \\ e_n(x) \end{bmatrix}}^{\text{e}} = \overbrace{\begin{bmatrix} b_1(x) \\ \vdots \\ \\ b_n(x) \end{bmatrix}}^{\text{b}}$$

The polynomial all are part of $F_q[x]$, meaning the group of possible values and this defines also the key length, and the result is of type $b_i(x) = (a_i(x) \cdot s(x)) + e_i(x)$, as it can be seen also from the matrix formula above. In this problem, again only the matrix A and b is known and used as public key. It should be computationally improbable to recover the polynomial $s(x)$ from just those two matrices.

There are two versions of the problem proposed for the learning with errors over rings system, these have different applications and are used for different purposes:

- **Search**: Proposes to find the unknown polynomial $s(x)$ given the list of polynomial pairs (the matrices) $(a_i(x), b_i(x))$.
- **Decision**: Given a list of polynomial pairs $(a_i(x), b_i(x))$ similar to **Search** version, but it must be determined whether the $b_i(x)$ polynomials were constructed as $b_i(x) = (a_i(x) \cdot s(x)) + e_i(x)$ or were generated randomly from $\mathbf{F}_q[x]$ with coefficients from all of F_q.

In this paper [67] Oded Regev the creator of LWE, together with Vadim Lyubashevsky and Chris Peikert, analyze the security reduction of LWE to be as hard as worst-case lattice problems. "Unfortunately, LWE applications are rather inefficient due to an inherent quadratic overhead in the use of LWE. Because of this impediment, ring LWE provided an advancement that includes the first truly practical lattice-based public-key cryptosystem with an efficient security reduction; moreover, many of the other applications of **LWE can be made much more efficient through the use of ring-LWE**".

7.5. Practical Use (NIST Finalists)

This section presents the algorithms where lattice family was proposed in NIST's competition. As of writing this thesis, the finalists were announced and they are the following: FIPS 203, Module-Lattice-Based Key-Encapsulation Mechanism Standard is based on CRYSTALS-KYBER submission which is based on ML-KEM [74]. This is used for securing data against post quantum attacks as well as traditional computing. It

uses Module LWE+R with base ring $Z[x]/(3329, x^{256}+1)$, as described earlier, this is learning with errors over rings which focuses on adding error on a polynomial equation. This $Z[x]/(3329, x^{256}+1)$, meaning the modulus is 3329 and $x^{256}+1$ there are a number of 256 possible powers, meaning the power of x is in range $[0, .., 255]$.

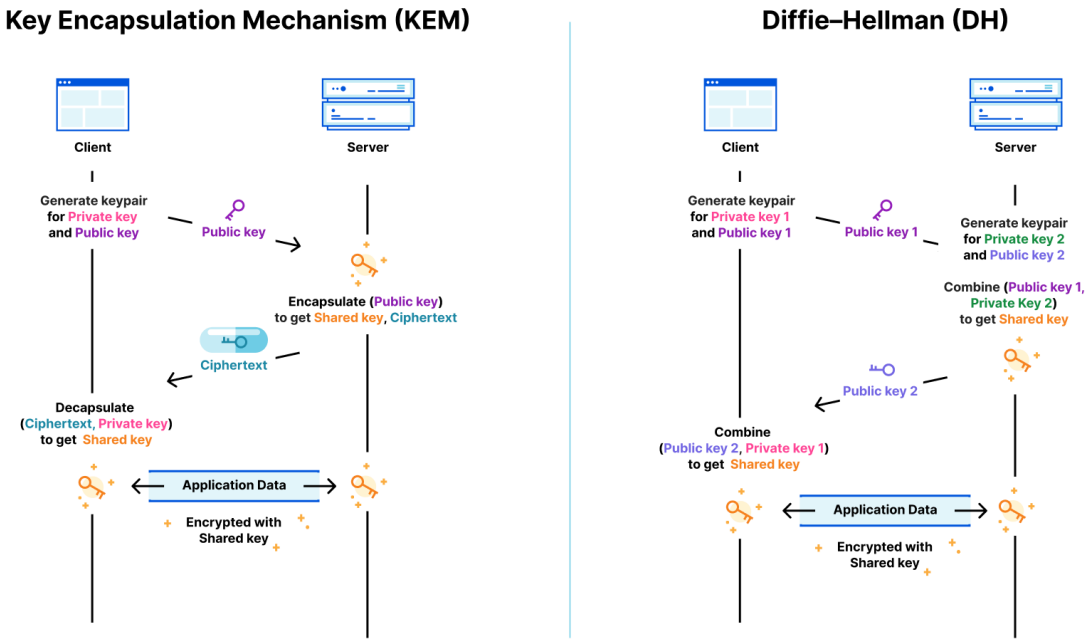

Figure 7.17.: Visualisation of Key Encapsulation Mechanism(KEM) vs Deffie-Hellman(DH) [75]. The difference here is the sharing of a Ciphertext instead of the public key, because a public key can be used by a quantum computer to discover the secret

FIPS 204, Module-Lattice-Based Digital Signature Standard is used for signatures and is based on CRYSTALS-Dilithium submission, which is based on ML-DSA [76][77]. ML-DSA is based on the hardness of lattice problems over lattice space and is focused on digital signatures. This uses the described power of lattice space to confirm digital signatures, similar to the other proposed algorithm using hashes. These provide the best time and memory size for the security provided, which nowadays is more important than size but still important. This is because of the small information provided needed to decrypt the data, as discussed, only some parameters or formulas are being provided, which is very small to store.

8. Results

This thesis proposed to study the new approaches to cryptography and the families that the most used algorithms in NIST rounds are based on. An overview of quantum computers and why they are capable of decrypting existent data was presented, as well as understanding how these algorithms work and what they do. Towards the end of the thesis, the lattice family chapter 7 was explored, starting with GGH which was the original algorithm. Learning with errors and RLWE, which are lattice-based algorithms, shown promise and they got elected as being future-proof. The initial purpose of the thesis was to study Lattice algorithms and how they will be the chosen standard, in the meantime NIST confirmed these ideas.

Some things can e conducted about lattice spaces:

- Because of the limited number of vectors, lattices will form a pattern. Using Voronoi diagram the area of each point can be mapped and the area of influence fr each point can be seen.
- Depending on the vectors, $QuadrantIII$ and $QuadrantI$ will be the same because of addition and subtraction of base vectors. This can reduce computational time because only one Quadrant can be calculated and then reproduced, even to the combination of vectors to form the respective mapped point!
- Recursive programing was used, with storage of the points to be efficient (dynamic programming), and this provided a drastic reduction in time. Further research in this area may provide a better solution using dynamic programming for much easier prediction, or in the base case scenario even finding a formula for this computation.
- Both SVP and CVP are very similar, adding an error and choosing a point (in the case of CVP is the origin).

These are very interesting findings because it can provide a better view and understanding of lattice spaces, and they are not talked about. The overview of each crypto family may help everyone not fear quantum computers, a big presumption is that they are unstoppable, but the truth is they are still just computers, they need an algorithm to work. They are not a magical box which solves problems by itself, it still needs instructions. Their most impressive characteristic is superposition, which enables finding the right answer

faster, but not instantaneously. Evolution helps us and the fear comes from not knowing what is happening. This thesis proposed to present and understand what will happen in the future and help the reader not fear the unexpected. Humans will always find solutions to problems, that is our nature.

9. Discussion

The initial goal was to analyze lattice based algorithms but that cannot be done without also understanding the reason we need it. Post quantum families and why quantum computing can be a danger for the traditional encryption based on Deffie-Hellman and AES must be understood beforehand, to compare each approach and choose the best one. Before starting the thesis I did research about quantum algorithms, the lattice-based family stood out because there were a lot of uses within NIST rounds. After reading about it and seeing the different approach compared to the other families it peaked my interest. Two were already well established, the symmetric based algorithms like AES and elliptic curves/asymmetric algorithms like RSA. Elliptic curves are very interesting in their different approach, using isogeny between two curves to create the private/public key. Hash based were already for a long time in cryptocurrency schemas (and also programming). Multivariate cryptography was already known and it provided some similarity to learning with errors. Error correcting codes(ECC) actually are very interesting and another paper may be necessary to explore them closer.

I believe every point was touched on this thesis and it was an extensive research into lattices and other crypto families. The code made it much easier to represent and map out the graphs and provided helpful perspective into them. Shor's and Grover's algorithms were discussed for a better understanding how they work and why they prove to be a danger for the traditional encryption. Quantum computers play a big role for these algorithms and without their superposition propriety they would not be able to provide the efficiency they are predicted to do. We still haven't seen it in action for a real RSA sized key but technology is evolving very fast, as Moore's law says.

This thesis also helped me have a better understanding of everything that was talked about, and it gave me a lot of knowledge I wouldn't have been able to learn without this push. I also believe further research should be done in the area of lattice-based algorithm, as well as ECC because they seem very interesting. Programming can provide a lot of ways to visualize the data needed, like mapping a graph or writing an error correcting code. Dynamic programming can prove to be a powerful tool for lattice problems like SVP and CVP which are considered to be NP-hard. More research is needed in the governance area, like

implementation of these algorithms and requirements that must be met to be sure a company is secure. Security of clients' data should be a priority for them.

Research is a powerful tool to find weaknesses as well as strengths in encryption algorithms, which is needed to provide the security we all desire.

10. Conclusion

Overall, Lattice-based family proved to be a very interesting algorithm out of all the families in PQC, that was the reason of this thesis, to explore further it's properties and understand the competition, and why it was such a high interest over the NIST rounds. Quantum computers are a very interesting tool that can be a danger for future algorithms, but they can also improve functionality in other areas like databases(Grover's algorithm can be used to quickly find data in a database). The other PQC families are very interesting and I was happy to explore them, even though documentation is scarce for them. Even though when this thesis started, a lot of candidates were proposed, lattice based algorithms rose on top between competitors and this made me more interested in it and made me think it will be elected. When the thesis was done, the results were announced, and they speak for themselves. The two lattice-based algorithms are very promising and present resilience against quantum computers. Lattice-based algorithm has a lot to live up to, for many years RSA and AES provided our security of data, now it will be it's turn to provide us with this security for the years to come.

10.1. Future Work

A deep dive was conducted for lattice-space and also understanding of the other PQC families, and algorithms that breaks today's encryption using quantum computers like RSA and AES. Although a lot was discussed, this is a very important topic because it provides security for our data, nations and private information. Further research should be done for lattice-spaces, especially because it was chosen as a standard. Once elected more approaches should be taken into proving that it is secure, and actually into trying to compromise it, before a malicious actor does. Nobody knows over the years what will be the next big algorithm or computer to break it, so constant research should be done. Another intriguing family was the error correcting code, which seemed very interesting in my research and I think it would provide a good topic to document. The other families and algorithms based on them should be explored deeper and I believe there are many more security reductions problems that could be used to provide this security.

Bibliography

[1] *Quantinuum*. [Online]. Available: https://www.quantinuum.com/press-releases/quantinuum-launches-industry-first-trapped-ion-56-qubit-quantum-computer-that-challenges-the-worlds-best-supercomputers, (accessed: 10.11.2024).

[2] *Meet willow, our state-of-the-art quantum chip*. [Online]. Available: https://blog.google/technology/research/google-willow-quantum-chip/, (accessed: 29.04.2025).

[3] *Announcing Approval of Three Federal Information Processing Standards (FIPS) for Post-Quantum Cryptography*. [Online]. Available: https://csrc.nist.gov/News/2024/postquantum-cryptography-fips-approved, (accessed: 23.10.2024).

[4] Chris Peikert, "A decade of lattice cryptography," *Foundations and Trendső in Theoretical Computer Science*, vol. 10, no. 4, Michele Mosca, Ed., pp. 283–424, 2016, Chris Peikert (2016), "A Decade of Lattice Cryptography", Foundations and Trendső in Theoretical Computer Science: Vol. 10: No. 4, pp 283-424. http://dx.doi.org/10.1561/0400000074. DOI: http://dx.doi.org/10.1561/0400000074.

[5] Chris Peikert, "Lattice cryptography for the internet," Michele Mosca, Ed., pp. 197–219, 2014.

[6] Fábio Borges, Paulo Ricardo Reis, and Diogo Pereira, "A comparison of security and its performance for key agreements in post-quantum cryptography," *IEEE Access*, vol. 8, pp. 142 413–142 422, 2020. DOI: 10.1109/ACCESS.2020.3013250.

[7] B Gan L Yokubov, "A performance comparison of post-quantum algorithms in blockchain," *Dept of Electronic and Electrical Engineering Research Papers*, 2022. DOI: https://doi.org/10.31585/jbba-6-1-(1)2023.

[8] Rami Elkhatib, Brian Koziel, and Reza Azarderakhsh, "Faster isogenies for post-quantum cryptography: Sike," Steven D. Galbraith, Ed., pp. 49–72, 2022.

[9] Wouter Castryck and Thomas Decru, "An Efficient Key Recovery Attack on SIDH," Carmit Hazay and Martijn Stam, Eds., pp. 423–447, 2023.

[10] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis, "Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH," CoNEXT '20, pp. 149–156, 2020. DOI: 10.1145/3386367.3431305. [Online]. Available: https://doi.org/10.1145/3386367.3431305.

[11] Ritik Bavdekar, Eashan Jayant Chopde, Ashutosh Bhatia, Kamlesh Tiwari, Sandeep Joshua Daniel, and Atul, "Post quantum cryptography: Techniques, challenges, standardization, and directions for future research," 2022. arXiv: 2202.02826 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2202.02826.

[12] Tiago M. Fernández-Caramès and Paula Fraga-Lamas, "Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks," *IEEE Access*, vol. 8, pp. 21 091–21 116, 2020. DOI: 10.1109/ACCESS.2020.2968985.

[13] Guillaume Hanrot and Damien Stehlé, "Improved analysis of kannan's shortest lattice vector algorithm," Alfred Menezes, Ed., pp. 170–186, 2007.

[14] MATZOV, "Report on the Security of LWE: Improved Dual Lattice Attack," Apr. 2022. DOI: 10.5281/zenodo.6412487. [Online]. Available: https://doi.org/10.5281/zenodo.6412487.

[15] Mostafa Taha and Thomas Eisenbarth, "Implementation attacks on post-quantum cryptographic schemes," 2015. [Online]. Available: https://eprint.iacr.org/2015/1083.

[16] Mohamed Henini and Marcelo Oliveira Rodrigues, "Quantum materials, devices, and applications," 2023. [Online]. Available: https://doi.org/10.1016/C2019-0-01977-2.

[17] F. Bloch, "Nuclear induction," *Phys. Rev.*, vol. 70, pp. 460–474, 7-8 Oct. 1946. DOI: 10.1103/PhysRev.70.460. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.70.460.

[18] *Quantum computing*. [Online]. Available: https://qoqms.phys.strath.ac.uk/research_qc.html, (accessed: 29.07.2024).

[19] P. A. M. Dirac, "A new notation for quantum mechanics," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 35, no. 3, pp. 416–418, 1939. DOI: 10.1017/S0305004100021162.

[20] Brian C. Hall, "Quantum theory for mathematicians," [Online]. Available: https://doi.org/10.1007/978-1-4614-7116-5.

[21] Anuj Dawar, *Quantum computing*. [Online]. Available: https://www.cl.cam.ac.uk/teaching/0910/QuantComp/notes.pdf, (accessed: 26.07.2024).

[22] P. van Oorschot A. Menezes and CRC Press S. Vanstone, *Chapter from the handbook of applied cryptography*, 1996. [Online]. Available: https://cacr.uwaterloo.ca/hac/about/chap8.pdf, (accessed: 30.07.2025).

[23] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976. DOI: 10.1109/TIT.1976.1055638.

[24] *A method for obtaining digital signatures and public-key cryptosystems*. [Online]. Available: https://web.williams.edu/Mathematics/lg5/302/RSA.pdf, (accessed: 10.11.2024).

[25] Samuel Horsley, "Ko kinon epato eno . or, the sieve of eratosthenes. being an account of his method of finding all the prime numbers, by the rev. samuel horsley, f. r. s.," *Philosophical Transactions (1683-1775)*, vol. 62, pp. 327–347, 1772, ISSN: 02607085. [Online]. Available: http://www.jstor.org/stable/106053 (visited on 04/30/2025).

[26] *How to generate Large Prime numbers for RSA Algorithm*. [Online]. Available: https://www.geeksforgeeks.org/how-to-generate-large-prime-numbers-for-rsa-algorithm/, (accessed: 26.07.2024).

[27] F. Arnault, "Rabin-miller primality test: Composite numbers which pass it," DOI: 10.1090/S0025-5718-1995-1260124-2. [Online]. Available: https://www.ams.org/journals/mcom/1995-64-209/S0025-5718-1995-1260124-2/S0025-5718-1995-1260124-2.pdf.

[28] Gerhard Jaeschke, "On strong pseudoprimes to several bases," DOI: 10.1090/S0025-5718-1993-1192971-8. [Online]. Available: https://doi.org/10.1090/S0025-5718-1993-1192971-8.

[29] J. L. Selfridge Carl Pomerance and Samuel S. Wagstaff, "The pseudoprimes to 25*10^9($paraphrasing$)," DOI: 10.1090/S0025-5718-1980-0572872-7. [Online]. Available: https://doi.org/10.1090/S0025-5718-1980-0572872-7.

[30] Peter W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997. DOI: 10.1137/S0097539795293172. [Online]. Available: https://doi.org/10.1137/S0097539795293172.

[31] *Euclidean algorithms (basic and extended)*. [Online]. Available: https://www.geeksforgeeks.org/euclidean-algorithms-basic-and-extended/, (accessed: 26.07.2024).

[32] *Shor's algorithm*. [Online]. Available: https://uwillnvrknow.github.io/deCryptMe/pages/shor.html, (accessed: 25.08.2024).

[33] P.W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," pp. 124–134, 1994. DOI: 10.1109/SFCS.1994.365700.

[34] Peter W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997. DOI: 10.1137/S0097539795293172. eprint: https://doi.org/10.1137/S0097539795293172. [Online]. Available: https://doi.org/10.1137/S0097539795293172.

[35] *Shor's algorithm (classically)*. [Online]. Available: https://courses.physics.illinois.edu/phys498cmp/sp2022/QC/Shor-Classical.html, (accessed: 26.07.2024).

[36] *IBM Condor*. [Online]. Available: https://www.ibm.com/quantum/blog/quantum-roadmap-2033, (accessed: 10.11.2024).

[37] Gordon E. Moore, "With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65,000 components on a single silicon chip," [Online]. Available: http://cva.stanford.edu/classes/cs99s/papers/moore-crammingmorecomponents.pdf.

[38] H. Delfs and H. Knebl, *Introduction to Cryptography: Principles and Applications* (Information Security and Cryptography). Springer, 2007, ISBN: 9783540492436. [Online]. Available: https://books.google.at/books?id=Nnvhz_VqAS4C.

[39] "Advanced Encryption Standard (AES)," [Online]. Available: https://doi.org/10.6028/NIST.FIPS.197-upd1, (accessed: 10.11.2024).

[40] "Information theory and entropy," pp. 51–82, 2008. DOI: 10.1007/978-0-387-74075-1_3. [Online]. Available: https://doi.org/10.1007/978-0-387-74075-1_3.

[41] Lov K. Grover, "A fast quantum mechanical algorithm for database search," STOC '96, pp. 212–219, 1996. DOI: 10.1145/237814.237866. [Online]. Available: https://doi.org/10.1145/237814.237866.

[42] *All About XOR*. [Online]. Available: https://accu.org/journals/overload/20/109/lewin_1915/, (accessed: 10.11.2024).

[43] Nicolas; et al. Courtois, *Solving underdefined systems of multivariate equations*. [Online]. Available: http://www.goubin.fr/papers/CG02.pdf, (accessed: 30.04.2025).

[44] J. Russell and R. Cohn, "Lavarand," 2012. [Online]. Available: https://books.google.at/books?id=K7p2MAEACAAJ.

[45] Jintai Ding and Dieter Schmidt, "Rainbow, a new multivariable polynomial signature scheme," John Ioannidis, Angelos Keromytis, and Moti Yung, Eds., pp. 164–175, 2005.

[46] Yuntao Wang, Yasuhiko Ikematsu, Koichiro Akiyama, and Tsuyoshi Takagi, "Cryptanalysis of GiophantusTM Schemes against Hybrid Attack," APKC '20, pp. 28–35, 2020. DOI: 10.1145/3384940.3388958. [Online]. Available: https://doi.org/10.1145/3384940.3388958.

[47] Ralph Charles Merkle, "Secrecy, authentication, and public key systems," [Online]. Available: https://www.ralphmerkle.com/papers/Thesis1979.pdf, (accessed: 30.04.2025).

[48] Moti Yungz Moni Naory, "Universal oneway hash functions and their cryptographic applications," [Online]. Available: https://www.wisdom.weizmann.ac.il/~naor/PAPERS/uowhf.pdf, (accessed: 30.04.2025).

[49] *The SPHINCS+ Signature Framework*. [Online]. Available: https://sphincs.org/data/sphincs+-paper.pdf.

[50] Richard W. Hamming, "Quantum materials, devices, and applications," undated; ca. 1982, Bell Syst. Tech. J. 29:147-60, 1950. [Online]. Available: https://hdl.handle.net/10945/46756.

[51] R. J. McEliece, "A Public-Key Cryptosystem Based On Algebraic Coding Theory," *Deep Space Network Progress Report*, vol. 44, pp. 114–116, Jan. 1978.

[52] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems of Control and Information Theory. Problemy Upravlenija I Teorii Informacii. 15: 159-166.*, 1986.

[53] George W. Sardinas August Albert; Patterson, "A necessary and sufficient condition for the unique decomposition of coded messages," 1953.

[54] Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede, "Timing attacks on error correcting codes in post-quantum schemes," 2019, cryptoeprint:2019/292. [Online]. Available: https://eprint.iacr.org/2019/292.

[55] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962. DOI: 10.1109/TIT.1962.1057777.

[56] Weijie Wang, Yujie Lu, Charalampos Papamanthou, and Fan Zhang, "The locality of memory checking," 2023. [Online]. Available: https://eprint.iacr.org/2023/1358.

[57] Spencer E. Dickson, "A torsion theory for abelian categories," DOI: 10.1090/S0002-9947-1966-0191935-0. [Online]. Available: https://doi.org/10.1090/S0002-9947-1966-0191935-0.

[58] Nathan Jacobson, *Basic Algebra I: Second Edition*, ISBN: 0486135225, 9780486135229. [Online]. Available: https://books.google.ro/books?id=JHFpv0tKiBAC&redir_esc=y, (accessed: 29.04.2025).

[59] Andrew Sutherland, *18.783 elliptic curves lecture 1*. [Online]. Available: https://math.mit.edu/classes/18.783/2017/Lecture1.pdf, (accessed: 27.04.2025).

[60] ISARA Corporation Victoria de Quehen Security Researcher, *Math paths to quantum-safe security: Isogeny-based cryptography*. [Online]. Available: https://www.isara.com/blog-posts/isogeny-based-cryptography.html, (accessed: 29.04.2025).

[61] *The Case for Elliptic Curve Cryptography*. [Online]. Available: https://web.archive.org/web/20090117023500/http://www.nsa.gov/business/programs/elliptic_curve.shtml, (accessed: 27.04.2025).

[62] *Kerberos 5 release 1.21.3*. [Online]. Available: https://web.mit.edu/kerberos/krb5-1.21/, (accessed: 26.04.2025).

[63] *About NIST*. [Online]. Available: https://www.nist.gov/about-nist, (accessed: 02.10.2024).

[64] *Call for proposals*. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals, (accessed: 04.10.2024).

[65] *Post-Quantum Cryptography Standardization*. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization, (accessed: 10.11.2024).

[66] Andreas Hülsing, "WOTS+ – shorter signatures for hash-based signature schemes," 2017. DOI: 10.1007/978-3-642-38553-7. [Online]. Available: https://eprint.iacr.org/2017/965.

[67] Vadim Lyubashevsky, Chris Peikert, and Oded Regev, "On ideal lattices and learning with errors over rings," 2012. [Online]. Available: https://eprint.iacr.org/2012/230.

[68] *Matplotlib: Visualization with python*. [Online]. Available: https://matplotlib.org, (accessed: 26.10.2024).

[69] *Numpy*. [Online]. Available: https://numpy.org, (accessed: 26.10.2024).

[70] *Nguyen, phon. cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto '97*. [Online]. Available: https://static.aminer.org/pdf/PDF/000/120/306/cryptanalysis_of_the_goldreich_goldwasser_halevi_cryptosystem_from_crypto.pdf, (accessed: 22.02.2025).

[71] *Rat in a maze*. [Online]. Available: https://www.geeksforgeeks.org/rat-in-a-maze/, (accessed: 29.10.2024).

[72] P.A. Burrough, R.A. McDonnell, and C.D. Lloyd, *Principles of Geographical Information Systems*. OUP Oxford, 2015, ISBN: 9780198742845. [Online]. Available: https://books.google.at/books?id=kvoJCAAAQBAJ.

[73] Oded Regev, "On lattices, learning with errors, random linear codes, and cryptography," 2005. DOI: 10.1145/1060590.1060603. [Online]. Available: https://doi.org/10.1145/1060590.1060603.

[74] *ML-KEM*. [Online]. Available: https://openquantumsafe.org/liboqs/algorithms/kem/ml-kem.html#primary-source, (accessed: 23.10.2024).

[75] *KEM vs Diffie-Hellman - Cloudfare*. [Online]. Available: https://blog.cloudflare.com/content/images/2022/10/image3.png, (accessed: 2.11.2024).

[76] *ML-DSA*. [Online]. Available: https://openquantumsafe.org/liboqs/algorithms/sig/ml-dsa.html, (accessed: 23.10.2024).

[77] *Dilithium*. [Online]. Available: https://pq-crystals.org/dilithium/, (accessed: 03.11.2024).

A. Sieve of Eratosthenes

```
import math
# An implementation of Sieve of Eratosthenes
# It is an algorithm for finding prime numbers up to a value N
N=25

#Boolean array, similar to frequency arrays, used to check the numbers
    that are not prime
primes=[1 for i in range (N+1)]

#Starting from 2
num=2

#This is an optimization for shortening the time (also a prime number can
    be check by dividing numbers up to sqrt(N) to optimze the time)
while num <= math.sqrt(N):
    #If num is already not a prime is not worth searching for other primes
        that are multiple of it
    if primes[num]==1:
        #Using a step of num, we filter the numbers that are not prime
            (because they are a multiple of the num)
        for i in range (pow(num,2),N+1,num):
            primes[i]=0
    # After filtering num moves to the next number, and if it was not
        already filtered, it means it is a prime.
    num+=1

#Printing the result:
print(f"Primes numbers up to {N} are:")
```

```
   for prime in range(2,N+1):
      if primes[prime]==1:
         print(prime,end=" ")
```

B. GCD

```
#Greatest common divisor function based on the Euclidean algorithm
def gcd(a, b,iter=0):
print(f"Iteration {iter}:a={a} b={b}")
if a == 0:
   return b

return gcd(b % a, a,iter+1)
```

C. Lattice generation code

```
import numpy as np
import matplotlib.pyplot as plt

#The basis for the lattice
base1=[0,5]
base2=[5,0]

#The number of iterations desired. This will impact time but will yeild
    more points.
maxIterations=100

#Storing of the Lattice points
latticePoints = []

#Recursive method to generate a point using the vector bases
#Count is used to store also the combination of vectors for the point
def nextPoint(x=0, y=0,iter=0,count1=0,count2=0):
    #Can be uncommented to disable points in third quadrant
    #and be computed just by multiplying with -1 points in first
        quadrant.(x>0 and y>0)
    #if x<0 and y<0:
        #return

    #Stop condition for recursive method
    if iter>maxIterations:
        return
```

```
    #Another stop condition for optimizing, will provide better view of
        points if commented, but slower.
    if [x,y,count1,count2] in latticePoints:
        return

    #Appending the point to storage and calling the function again with the
        other base vector.
    latticePoints.append([x,y,count1,count2])
    nextPoint(x+base1[0],y+base1[1],iter+1,count1+1,count2)
    nextPoint(x+base2[0],y+base2[1],iter+1,count1,count2+1)
    nextPoint(x-base1[0],y-base1[1],iter+1,count1-1,count2)
    nextPoint(x-base2[0],y-base2[1],iter+1,count1,count2-1)


if __name__=="__main__":
    # Call initial function
    nextPoint()

    #Test for more optimizing, unfinished, more research is needed.
    #for pt in latticePoints.copy():
        #latticePoints.append([pt[1],-pt[0]])


    #Get absolute maxes for axis ranges to center origin
    #This is optional, it can be set like on the following line
    #maxes=1.02*np.max(latticePoints)
    maxes=20

    #Map the lattice points
    for x,y,c1,c2 in latticePoints:
        # Don't draw points that will not be seen in the graph
        if abs(x) > maxes or abs(y) >maxes:
            continue
        #Optimization for third quadrant
        # if x<0 and y<0:
        #    continue
```

```
        # if x>0 and y>0:
        #    plt.plot(-x,-y,'r+')

        #Plot the point
        plt.plot(x,y,'.c')
        #Write the position
        plt.text(x,y,f"[{x},{y}]\n({c1},{c2})")

    plt.plot(0,0,'ok') #<-- plot a black point at the origin
    plt.axis('equal') #<-- set the axes to the same scale

    #for setting the same scale on Ox and Oy
    ax = plt.gca()
    ax.set_aspect('equal', adjustable='box')

    plt.xlim([-maxes,maxes]) #<-- set the x axis limits
    plt.ylim([-maxes,maxes]) #<-- set the y axis limits
    plt.grid(visible=True, which='major',alpha=0.2,zorder=1) #<-- plot grid
        lines

    #Display the graph
    plt.show()
```