Evaluating the Robustness of a Lattice-Based Cryptosystem

Author Andrea Paige Pesigan [C1606020]
Final-Year Computer Science Undergraduate at Cardiff University

Project Supervisor Eirini Anthi
IoT and ICS Security Research Associate at Cardiff University

Project Moderator Dr. Frank Langbein
Senior Lecturer in the School of Computer Science and Informatics at Cardiff University

CM3203 One Semester Individual Project
40 credits

15 May 2020

Abstract

As part of the student's undergraduate degree in computer science, the security and practicality of a lattice-based cryptosystem was assessed. This task was accomplished by choosing an implementation of NTRUEncrypt to analyse, performing select experiments using classical computing, and researching current quantum computing attacks. As a result of the project, it was determined that NTRU will, for the foreseeable future, remain resilient against the current attack methods analysed of both classical and quantum nature, provided that the correct parameters are chosen. However, it may only be a matter of time before a quantum algorithm is designed that effectively solves the mathematical problem the security of NTRU relies upon.

Acknowledgements

I would like to express immense gratitude to my supervisor, Irene. You are someone I look up to, and I know you will continue to accomplish whatever you set out to do. Thank you for your support and friendship.

I would also like to thank my project moderator, Dr. Frank Langbein, for guidance with my project. Your passion for a variety of topics is an inspiration to others.

A special thank you goes out to Mr. Jędrzej Krauze, whose code was analysed as part of this thesis.

Lastly, I would like to thank the lecturers, the staff, and my fellow classmates at Cardiff University for their support throughout my degree and for encouragement in my project.

Contents

List of Figures

1 Introduction

Cryptography has had practical worth for millennia, from its use to securely transport war intelligence in the times of Julius Caesar, to ensuring that payment details are seen only by the intended recipients in the digital age. This subtopic within mathematics (and, in recent decades, computer science) has seen many important uses, and it is vital that it continues to be studied and improved further to prevent malicious entities from uncovering confidential information.

Every cipher created has a way to recover the key used or recover the plaintext from the ciphertext and render the encryption futile, with some methods better than others. A key used in a Caesar cipher is easily found with frequency analysis, whereby an attacker can match up the most frequently-used letters of the ciphertext to the most frequently-used letters of the language in plaintext use. DES can be broken with differential cryptanalysis. With greater computing power, smaller keys for digital ciphers can be brute-forced in less time, leading to the need for larger keys. Ideally, any given key should be infeasible to brute-force both in the present and in the far future where the technology to perform brute-forcing has improved. With a large enough key space, many widely-used ciphers are protected, at least from computers as we have known them before quantum.

Whereas private-key cryptographic methods have a great reliance on properties such as key sizes and block modes for their robustness, public-key methods have more of a reliance on the difficulty of solving certain mathematical problems. The well-known RSA public-key algorithm relies on the difficulty of factoring the products of large prime numbers. It is far easier to multiply two large prime numbers to get a product than it is to find those factors when given just the product, since the only factors of that number, aside from the number itself and the number 1, are those two large primes. Diffie-Hellman Key Exchange (DHKE) another public-key system, used primarily for secure exchange of encryption keys between parties, relies on the discrete logarithm problem. In essence, as long as the key space is sufficiently large and it is computationally infeasible to solve these mathematical problems (meaning that no algorithm has been discovered that solves such a problem in a reasonable amount of time), the cryptosystems are safe.

Enter quantum computing. Though the field is in its infancy, the promises of great computational power are enticing, and some experts expect quantum computers of sufficient computing power to model and simulate complex biological systems, to start becoming commercially available next decade [1]. The excitement is appropriate as this is a time where increases in computational power of classical computers (such as those that use silicon processors) are slowing down [2]. Creating algorithms requires a different way of thinking when it comes to quantum computing, and the hardware to take advantage of quantum mechanics is far from reliable, let alone standardised. But already there are a handful of algorithms published that have the potential to change how we must perform tasks such as cryptography. An algorithm that has this potential is Shor's algorithm, which can factor large integers

in polynomial time, rendering cryptosystems such as RSA insecure [3].

Private-key cryptosystems such as AES are said to still be secure against quantum computers, even with Grover's quantum search algorithm that reduces worst-case search time from $O(N)$ to $O(\frac{\pi\sqrt{N}}{4})$, as long as the key space is adequately large [4].

The United States standards agency National Institute of Standards and Technology (NIST) has recognised the importance of finding a public-key cryptosystem that can resist quantum computers (known as post-quantum cryptography or PQC); the organisation has initiated a process [5] to standardise one or more candidate cryptosystems. This process includes a call for submissions of post-quantum cryptosystems, evaluating the submissions, and standardising one or more finalists. At the time of writing, NIST is selecting the algorithms that will pass the second round of the PQC standardisation process [6]; the selection comes from the 26 candidate algorithms that passed the first round in early 2019.

Cryptography was chosen as the subject of this project due to its relevance to computer security. Post-quantum cryptography in particular was chosen as quantum computing is a growing and exciting field. There is clearly much to learn yet as the field of quantum computing is very young. Public-key cryptographic methods are being broken by this emerging technology, and it is in the public interest to explore post-quantum cryptography.

Lattice-based cryptography was chosen as the type of cryptography to be analysed as it is the most popular candidate for PQC. Lattice-based cryptography has garnered much interest because, whereas most cryptographic algorithms are only worst-case hard, lattice problems are proven to be average-case hard [7]. Of the 26 algorithms that made it to the second round of the NIST PQC standardisation competition, many of them are lattice-based [6], and IBM is focusing on lattice-based cryptography as its approach to PQC [8].

The aim of this project was to choose an implementation of a lattice-based cryptosystem and assess its robustness against both classical and quantum devices. Assessing its robustness involved understanding the cryptosystem and the mathematics behind what makes it secure, as well as understanding the foundations of quantum computing, its advantages over classical computing, and what can be done in the present to work with a quantum computer.

This report covers what was done throughout the course of the project to achieve the goal of assessing the robustness of the cryptosystem chosen (NTRU) by stating results and a reflection of the shortcomings of these results. The structure of the report is as follows: the report covers the background of the problem in greater detail in section 2, the methodology of choosing and analysing the cryptosystem in section 3, what results were found in section 4, and a discussion of the results to determine how robust the cryptosystem is in section 5. Some remarks on future directions of research and an overall summary are included in sections 6 and 7. In addition, a reflection of academic growth and personal development is provided in the final section 8.

As a result of the analysis performed, it appears that NTRU is secure against classical attacks provided the proper parameters are chosen, and will be resistant to quantum attacks

with current available quantum algorithms.

2 Background

2.1 NTRU History and Classical Attacks

A viable solution to PQC is lattice-based cryptography. This type of cryptography refers to schemes based on the mathematical objects known as lattices, part of the abstract algebra and geometry branches of mathematics. The lattice-based cryptosystem chosen for analysis for this project was NTRUEncrypt, hereby referred to simply as NTRU. It is worth noting that in practical usage of the term, NTRU can refer to the encryption scheme, the digital signature scheme, or the many variations of the two. This portion of the report provides a historical overview of NTRU and attacks against the cryptosystem. The next section 3, Methodology, will go into greater depth on the mathematics behind NTRU as well as why this particular cryptosystem was chosen.

NTRU stands for **N**-th degree **TRU**ncated polynomial ring. The cryptosystem was first devised around 1996 by Hoffstein, Pipher, and Silverman [9], though improvements have since been made to the original version of NTRU in speed and security. The patent for NTRU expired in 2017 [10], which has led to increased interest in the cryptosystem.

The main mathematical attacks against NTRU include lattice reduction, a chosen ciphertext attack, and a meet-in-the-middle attack. Perhaps the most widely-tested attack is lattice basis reduction (also referred to as lattice reduction or basis reduction), due to NTRU's security relying on the shortest vector problem (SVP), which lattice reduction can assist in solving. As a result of the popularity and effectiveness of this attack, the attack was chosen for exploration in this project. What exactly lattice reduction entails is better understood after covering foundational knowledge to understand lattices and NTRU.

One basis reduction method is the Block Korkin-Zolotarev (BKZ) algorithm [11], based on Gram-Schmidt orthogonalisation of vectors. Another, arguably more popular lattice basis reduction algorithm based on the number of NTRU studies performed on it, is the Lenstra-Lenstra-Lovasz (LLL) algorithm, published in 1982 [12]. LLL builds on BKZ and approximates a reduced basis in polynomial time, meaning it does not solve the SVP exactly. LLL has been found to be successful against NTRU by LatticeHacks [13] using SageMath, an open-source mathematics software system often used as an open-source alternative to software like MATLAB.

Another popular attack on NTRU is the aforementioned meet-in-the-middle attack [14], which attempts to get the private key from the public key. This attack operates by finding two parts of the private key such that certain mathematical conditions are satisfied, based on how keys are generated in NTRU. This attack can reduce the time to brute-force the private key by a square root. With careful parameter choice, however, this attack becomes infeasible.

The multiple-transmission attack, published by the original creators of NTRU [15] is another attack. Across many transmissions of the same message, an attacker can determine information about the coefficients of the private key, and paired with a brute-force attack for the rest of the coefficients, the cryptosystem can be broken. The authors then stated that practices such as some scrambling and padding of each message will prevent this attack.

In the chosen ciphertext attack [16], the secret key was determined from a few ciphertext-plaintext pairs with good probability, by taking advantage of the key generation process. It was concluded by the same authors that using a special type of padding and/or hashing in the messages should be used to mitigate this attack.

Hybrid attacks, as in [17], combine attacks such as those above. In this paper, Howgrave-Graham combined lattice reduction with a meet-in-the-middle attack. It was found that correct parameter choices will make this attack infeasible, however.

As a result of the above attacks, NTRU as originally published is not secure. Care needs to be taken to use secure padding methods, and the original parameter suggestions have since been re-announced. As stated in the NTRUEncrypt submission to the NIST PQC competition [18], $N = 443$ provides 128 bits classical security and 84 bits quantum security, and $N = 743$ provides 256 bits classical security and 159 bits quantum security, where N is a significant parameter in NTRU.

To understand the cryptosystem and lattice reduction attacks, the LatticeHacks LLL attack was reproduced for the implementation of NTRU found. Details and findings will be given in later sections. Other attacks were not deeply examined due to time constraints.

2.2 The Current Status of Quantum Computing

Quantum computation simply refers to performing computations using hardware that takes advantage of quantum mechanics. Subatomic particles have physically strange behaviour. An example of this peculiarity is demonstrated in the double-slit experiment, which has been performed in many variations with one recent case being [19]. Physicists wanted to see whether an electron behaved like a wave (implying electrons are energy) or a particle (implying electrons are matter). By firing electrons one at a time at a filter with one slit, the electrons produced a pattern on the other side of the filter that particles would make. After adding another slit, however, the physicists found that the electrons produced a pattern that looked like one made by a wave. The same wave-like behaviour occurred even after firing the electrons one at a time, making it seem as if when passing through the slits, an electron transforms from a particle to a wave, passes through both slits, and interferes with itself to create the patterns that resemble what a wave would produce. When a measuring device was placed at the slits to observe which slit the electrons passed through, the electrons again produced a pattern that a particle would produce, as if the electrons were aware they were being watched. This behaviour is still the subject of much research.

The takeaway from the double slit experiment is that in quantum mechanics, measuring a system changes the state of that system. The state of the electrons before being measured at the filter is known as *superposition*. Like Schrodinger's analogous cat being in the states of

both dead and alive before being observed, the physicists conducting the double-slit experiment initially did not know which slit the electron passed through to produce the wave-like behaviour; it seemed that the electrons passed through both, and observation (measurement) changed this behaviour.

Quantum computing makes use of superposition. The quantum computing equivalent of a bit (called a qubit) can be put into superposition, meaning that there is a probability that the qubit, when measured, will output 1, and a probability that it will output 0. But once measured, the qubit cannot be put back into the state it was in just before measuring; this is a one-way system. Another interesting and useful quantum phenomenon is *entanglement*, whereby one qubit's behaviour instantaneously influences another's. When quantum phenomena such as superposition and entanglement are used, the usefulness and advantages of quantum computing over classical become more apparent. Two qubits can represent four (2^2) numbers simultaneously, three qubits can represent eight (2^3), and so on.

Quantum computing today is in its infancy. The largest setback to making quantum computers mainstream is *decoherence*. That is, it is difficult to make a qubit maintain a state of superposition, similar to keeping a coin balanced on its side and preventing it from tipping over to heads or tails. Quantum computers can be constructed from whatever exhibits quantum behaviour, and one current physical implementation is through superconducting electronic circuits. Qubits in these computers need to be kept at extremely low temperatures, near absolute zero. Many forces, including light, sound, and heat, have the potential to make a qubit lose its state, making it very difficult to perform useful calculations with even relatively few qubits. There is a trade-off between keeping the qubits in the system resistant against information loss and making the qubits controllable. When preventing decoherence in qubits, the design must also ensure that it is easy to manipulate the state of the qubit. This task is a difficult one.

Google made headlines in the autumn of 2019 for claiming the achievement of quantum supremacy [20], meaning that the company's quantum computer (comprised of 53 qubits) demonstrated solving a problem that a classical computer cannot feasibly solve. A task that would take 10,000 years on a classical computer took Google's quantum computer a mere 200 seconds. IBM, Google's major rival in the field, was quick to express skepticism on this claim [21], saying that a powerful enough supercomputer could complete that task in 2.5 days. While a computer the size of about 50 qubits such as those announced by Google and IBM [21] is impressive given the ongoing decoherence problem, humanity is still a long way from scaling up these devices to be the thousands of qubits needed to break RSA with Shor's Algorithm [22], if key lengths are 2048 bits as is recommended by NIST for standard security [23].

It is currently difficult to say how much more powerful a quantum computer comprised of thousands of qubits has, practically, over the current classical ones, given how early it is in the stages of development for such devices. A quantum computer can perform the same computations a classical computer can, albeit through a different physical medium. However, without taking proper advantage of quantum phenomena such as entanglement and superposition, there is little gain. Although in theory 10 entangled qubits in superposition can store up to 1024 numbers at once, how these qubits are manipulated is what determines

how beneficial a quantum computer is over a classical computer. Algorithms such as that created by Shor provide that gain because they make use of these quantum properties.

A handful of tools already exist to help current software engineers begin developing quantum algorithms, such as IBM Qiskit, Microsoft Q#, and PyQuil [24]. However, these tools handle quantum computing at the circuit level, and, exactly like classical computing, layers of abstraction and compilers need to be created for more useful higher-level programming and for greater ease in creating algorithms.

As part of this project, some Qiskit programming was done to understand the advantages of quantum algorithms, but no results that were of great use towards breaking NTRU were produced.

Investigation was carried out to gain a firm foundational understanding of NTRU, quantum computing, and how quantum computing can render NTRU insecure, but to properly look at the other methods of attacks and to look further into quantum computing requires time that is out of the scope of this three- to four-month-long undergraduate project.

3 Methodology

The initial plan of the project was followed rather closely: the first week was spent searching for and deciding on an implementation of lattice-based cryptosystems to analyse, the next couple weeks were spent understanding the mathematics and how that cryptosystem works and what makes it secure, and the weeks after involved attacking the cryptosystem using classical computing methods. The last few weeks of the project before writing were dedicated to understanding the foundations of quantum computing with a focus on how it can be used to break the cryptosystem.

3.1 Choosing an Implementation

An existing implementation of a lattice-based cryptosystem was analysed rather than creating one as part of the project so that more time could be dedicated to analysis of its security and practicality. NTRU and GGH were chosen as the cryptosystems to find implementations of due to their popularity, as much has been written about these cryptosystems. NTRU is based on the shortest vector problem, while GGH is based on the closest vector problem.

GitHub, the popular code-sharing platform, was searched for existing implementations. Two implementations each of NTRU and GGH were downloaded and examined for the following criteria: documentation provided, flexibility for modification, understandability, speed, and testing performed. After examination, one candidate program stood out as the easiest to work with. The NTRU implementation in Python3 released by the GitHub user jkrauze [25] was noticeably and thoroughly documented, was readable, was structured such that functions are organised and their purposes are clear, and was mindful of using logging. The code demonstrated professionalism and care in its creation that the other three programs

simply did not have. For these reasons, this implementation was chosen as a focus of this project.

3.2 Understanding Lattices and NTRU

This subsection covers the mathematics necessary to understand NTRU, and the process performed as part of this project to obtain this knowledge.

The weeks immediately after choosing an implementation were spent understanding the mathematics behind lattices and NTRU, as this knowledge is required to recognise how attacks will work. The first task was to comprehend lattices.

To put this structure in layman's terms, lattices are a grid of points in a given number of dimensions, and those points are spaced apart regularly and stretch out in those dimensions infinitely. More formally, a lattice is a subgroup of the group $\mathbb{R}^n$, and the points in the lattice are a subspace of the real coordinate space $\mathbb{R}^n$. $\mathbb{R}$ refers to the set of real numbers and n refers to the number of dimensions. What defines which points in all of $\mathbb{R}^n$ are part of the lattice is the basis of the lattice. The basis is a set of vectors such that all linear combinations of those vectors produce that lattice. That is, all possible combinations of integer coefficients for the basis vector produce all points in the lattice, and by starting from any arbitrary point in the lattice, one can reach any other arbitrary point in the lattice with a combination of those vectors. A lattice has infinitely many bases, with shorter, orthogonal vectors producing a basis that is more manageable and easier to visualise.

Lattices are perhaps best understood visually. The following images are taken from the University of Tel Aviv autumn 2004 lectures on lattices by Oded Regev [26].

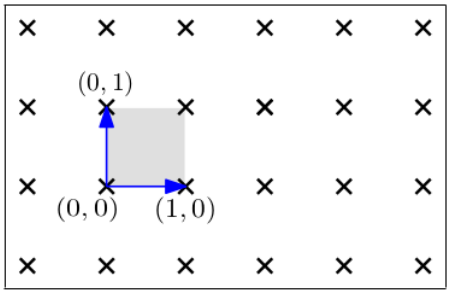

Figure 3.1: A basis in $\mathbb{Z}^2$, the set of integers in two dimensions

It becomes very difficult for humans to picture lattices beyond three dimensions.

Whereas the security of the public-key cryptosystem RSA is based on the difficulty of solving the mathematical problem of factoring the product of two large prime numbers, the security of lattice-based cryptosystems relies on the difficulty of mathematical problems with lattices. Two of these problems include the shortest vector problem (SVP) and the closest vector problem (CVP). The shortest vector problem is as such: given a lattice, find the shortest non-zero vector between two points. The closest vector is similar: given a lattice and a point p, find the point in the lattice that is closest to p. The SVP and the CVP are

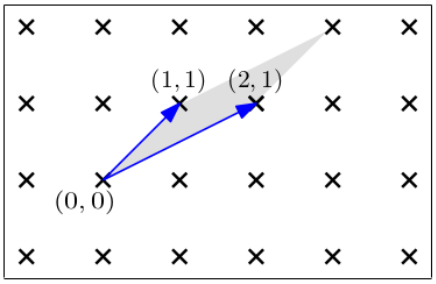

Figure 3.2: Another basis in $\mathbb{Z}^2$

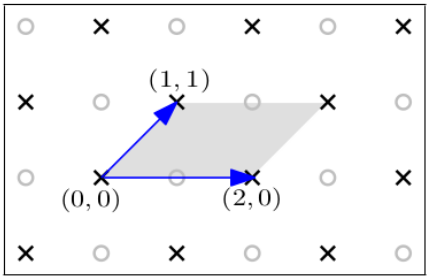

Figure 3.3: Not a basis in $\mathbb{Z}^2$

easy for both humans and computers to solve when the lattice is of a dimension of 2, but gets increasingly difficult as the number of dimensions grows. The security of NTRU is based on the SVP.

3.2.1 Mathematics for NTRU

As stated in the Background section, NTRU stands for **N**-th degree **TRU**ncated polynomial ring. The prerequisite mathematical knowledge to understand what this means and how the cryptosystem works is provided in this section.

Polynomials are mathematical expressions that consist of variables and coefficients. The operations involved in the expression are addition, subtraction, and multiplication, and the exponents of the variables are non-negative. The *degree* of a polynomial with one variable is the largest power of that polynomial, such as 6 in $x^2 + 3x^6 - 5$. In NTRU, the degree of the polynomials are defined by a value assigned the variable name N, hence "N-th degree".

The question naturally arises: how exactly are polynomials related to lattices? With NTRU, the coefficients of the polynomial form a vector in a lattice. The degree of the polynomial is equivalent to the number of dimensions in a lattice. As the vectors are linearly independent, each coefficient is equivalent to the magnitude of the vector in that dimension. The *cyclic convolution* of the coefficients provide the other basis vectors in that lattice. For example, with $N = 3$ the convolution of a vector $[3\ 1\ 4]$ in that lattice is $\begin{bmatrix} 3 & 1 & 4 \\ 4 & 3 & 1 \\ 1 & 4 & 3 \end{bmatrix}$, where the columns of the matrix provide each vector. Factoring the product of two such polynomials is similar to basis reduction in a lattice. Thus, while it is helpful to understand lattices and

basis reduction, it is more useful to focus on polynomials rather than lattices when learning about NTRU.

- A *ring* R is a set of elements with two operations (addition $+$ and multiplication $\times$) such that:
  - The set is closed under those two operations (performing one of those two operations on two elements in that set returns another element in that set)
  - The set is commutative under addition (for every a and b in R, $a + b = b + a$); commutative rings are also commutative over multiplication
  - The operations are associative (for every a, b, and c in R, then $(a+b)+c = a+(b+c)$ and $(a \times b) \times c = a \times (b \times c)$)
  - There is an identity element e under addition (for every a in R, $a+e = e+a = a$)
  - Every non-zero element has an inverse under addition (for every a in R, $a+(-a) = 0$)
  - The distributive property holds (for every a, b, and c in R, $a \times (b+c) = a \times b + a \times c$)
- $\mathbb{Z}$, the set of integers, satisfies the conditions of a ring.
- A *polynomial ring* $R[x]$ refers to all polynomials with coefficients in the ring R. $\mathbb{Z}[x]$ refers to all polynomials with coefficients being integers. The ring is formed from the set of polynomials, with coefficients in another ring (often a field such as $\mathbb{Z}_p$ where p is prime).
- The concept of fields, related to rings, will become relevant later in this section. It is a good time to note that rings, fields, and groups are all simply sets of elements, each with unique rules about how to combine elements to produce another element in the set. Performing the modulo operation on two polynomials is similar to performing the operations on standalone numbers: a mod b is the remainder after dividing a by b.
- *Truncation* in this case does not refer to the traditional sense of removing terms in the polynomial based on the power of the term (e.g. discarding x^3 in $x^3 + 5x^2 - 1$ to give $5x^2 - 1$), but rather in performing the modulo operation on the polynomial.
- NTRU operates on objects in the truncated polynomial ring R where

$$R = \mathbb{Z}[x]/(x^N - 1)$$

  . $x^N - 1$ indicates that a modulo operation is then performed on the polynomials with $x^N - 1$. As an example, if $N = 3$ and a mod$(x^3 - 1)$ operation is performed on $(x^2 + 1) \times (x^2 + 1)$, the answer is $2x^2 + x + 1$.
- Truncating using the modulo operation in NTRU is often paired with reducing polynomials in $\mathbb{Z}_p$. In $\mathbb{Z}_3$, for example, $x^3 + 3x^2 + 3x + 1$ is reduced to $x^3 + 1$.
- A *field* F is a commutative ring that also satisfies the conditions that:
  - There is an identity element under multiplication

– Every element has an inverse under multiplication (for every a in F, there is an element b also in F such that $a \times b = 1$)

Every field is a ring but not every ring is a field.

Finding the inverse of a polynomial in a field is relevant in NTRU especially in generating a public-private keypair. An example of this process is as described in [27]. The example can help in appreciating the mathematics of why NTRU is secure.

In this example, we are asked to find the inverse of x^2+1 in the polynomial ring $\mathbb{Z}_3[x]/(x^3+2x+1)$.

We can write $f(x) = x^3 + 2x + 1$ and $g(x) = x^2 + 1$, and find the inverse of g in the field $F_3[x]/(f)$. In other words, we are looking for h such that $gh \equiv 1 \bmod f$ or $gh + kf \equiv 1$ for some k in $F_3[x]$. We can find h and k using Euclid's algorithm:

$$f = xg + (x + 1)$$

$$g = (x + 2)(x + 1) + 2 = x^2 + 3x + 4 = x^2 + 1$$

$$x + 1 = 2(2x) + 1$$

By working backwards (all the while recalling to apply mod3 to coefficients):

$$1 = (x + 1) - 2(2x)$$

$$= (x + 1) - (2x)(g - (x + 2)(x + 1)$$

$$= (x + 1) - (2x)(g) + (2x)(x + 2)(x + 1)$$

$$= (x + 1) - (2x)(g) + (2x^2 + 4x)(x + 1)$$

$$= (2x^2 + x + 1)(x + 1) - (2x)(g)$$

$$= (2x^2 + x + 1)(f - xg) - (2x)(g)$$

$$= (2x^2 + x + 1)(f) - (2x^2 + x + 1)(x)(g) - (2x)(g)$$

$$= (2x^2 + x + 1)(f) + (-2x^3 - x^2 - x)(g) + (-2x)(g)$$

$$= (2x^2 + x + 1)(f) + (x^3 + 2x^2 + 2x)(g) + (x)(g)$$

$$= (2x^2 + x + 1)(f) + (x^3 + 2x^2)(g)$$

As such, the sought format of $gh + kf \equiv 1$ has been found. The inverse of $g \bmod f$ is $h = x^3 + 2x^2 (\mathrm{mod} f) = 2x^2 + x + 2 (\mathrm{mod} f)$. This result can be verified by confirming $gh (\mathrm{mod} f) = 1$. If we evaluate $\frac{(x^2+1)(x^3+2x^2)}{x^3+2x+1} = \frac{x^5+2x^4+x^3+2x^2}{x^3+2x+1}$ we produce $x^2 + 2x - 1$ with remainder $-3x^2 + 1$, which when performing mod3 for the coefficients, indeed produces 1. This process is helpful in understanding how key generation operates and why it is difficult to decrypt messages encrypted using NTRU. More linear algebra was learned to understand NTRU, but the most significant elements were summarised in this section.

3.2.2 What makes NTRU secure?

Another way to ask the above question is: what is the difficult mathematical problem that NTRU relies on? Related to the difficulty in solving the SVP with lattices in general, the problem that makes NTRU secure can be summarised as factoring polynomials. It is more difficult to factor polynomials than integers (like in RSA), especially in a truncated polynomial ring. What is especially hard in NTRU is that it is difficult to factor such a polynomial into two polynomials that have small coefficients, which the private key consists of. The exact process of key generation, encryption, and decryption are described in the next subsections.

Polynomial factoring is similar to lattice reduction in this case. To summarise lattice reduction, another basis for the same lattice is found where the vectors found are shorter and usually more orthogonal. With a reduced basis, the shortest vector problem is easier to solve, and a message encrypted using the larger basis as a public key may be decrypted with its reduced form.

The explanations of key generation, encryption, and decryption follow the jkrauze implementation of NTRU.

3.2.3 How Key Generation Works

1. Three numbers N, p, and q are agreed upon by the entities taking part in encryption. These values are not secret.

   N is a prime number that will determine the degree of the polynomials making up the keys; generally speaking, the larger N is, the more secure the encryption will be but the slower key generation, decryption, and encryption will be

   p is a small number (usually 3) to which each coefficient is reduced (so coefficients in the private key will be -1, 0, or 1 if $p = 3$)

   q is a number that's a power of 2 (usually 128 or 256) to which each coefficient is reduced (so coefficients in the public key are in the range $[-\frac{q}{2}, \frac{q}{2}]$); q is much larger than p; p and q should be relatively prime

   Security Innovation (who acquired NTRU Cryptosystems, Inc., a company created by the mathematicians who devised NTRU) have determined that $N = 251$, $p = 3$, $q = 128$ are considered standard security level.

   Another parameter, d, represents the number of -1 and +1 integers each in the production of f and g in the next step of the key generation process and in r in the encryption process. In jkrauze's implementation, the d value for f and g are separate, hereby referred to as d_f and d_g respectively. In this version of NTRU, $d_f = \lfloor \frac{N}{3} \rfloor$ and $d_g = d_r = \lfloor \sqrt{q} \rfloor$.

   N, p, and q are part of both the public key and the private key.

2. Choose two random polynomials f and g satisfying the following:

- The degree of both is at most $N-1$, as the result of performing the $\bmod(x^N-1)$ operation
- The coefficients of both will be -1, 0, or 1
- In f, there is a d_f number of $+1$ coefficients and a d_f number of -1 coefficients; it is the same for g and d_g
- f has inverses under mod$p$ and under mod$q$ (if this condition is not met, generate another f until it is met)

f is part 1 out of 2 of the private key

g is used in calculating the public key in step 4

3. Calculate the inverses f_p of $f(\text{mod}p)$, and f_q of $f(\text{mod}q)$. These will be polynomials of degree $N-1$. f_p will have coefficients in the range [-$\frac{p}{2}$, $\frac{p}{2}$] and f_q will have coefficients in the range [-$\frac{q}{2}+1$, $\frac{q}{2}$]. In other words, determine f_p and f_q such that:
   - $f \times f_p = 1(modp)$
   - $f \times f_q = 1(modq)$

   f_p is part 2 out of 2 of the private key in this implementation of NTRU; other versions may omit this

   f_q is used to calculate the public key in the next step
4. Calculate another polynomial $h = p \times f_q \times g(\text{mod}q)$ from the values established

   h is the polynomial that is the public key; coefficients will be in the range $[-\frac{q}{2}+1, \frac{q}{2}]$

To summarise, choose three parameters N, p, and q (sometimes d, depending on the implementation of NTRU) each with requirements, create polynomials based on those values and some other requirements, and calculate inverses of those polynomials. It is difficult to invert h over the ring to produce f or g. One might attempt to perform the procedure as described in section 3.2.1, but it is difficult to produce an f or g that has small coefficients.

3.2.4 How Encryption Works

1. Turn the message into a polynomial m

   This is usually done by using the binary values of the data, e.g. 1101 becomes x^3+x^2+1. If the message polynomial is longer than N, the message is split up into blocks each the size of N. In the jkrauze implementation, each block is encrypted separately and each block of ciphertext is then concatenated.
2. Choose a random polynomial r to be the blinding value.
   - This polynomial will be of degree $N-1$ with coefficients in the range [-1, 1]
   - There will be d_r amount of -1 and $+1$ coefficients
3. Calculate the ciphertext $e = r \times h + m(\text{mod}q)$

h and q are taken from the public key. r and m are determined from the previous steps.

e will be a polynomial of very high degree with coefficients in the range $[-\frac{q}{2}+1, \frac{q}{2}]$. "hello" gets encrypted to a polynomial of degree 66 in jkrauze's implementation with the parameters $N = 11$, $p = 3$, and $q = 16$.

To put the encryption process in the context of lattices, a random point p in N-dimensional space is chosen, and a message vector is added to it (that vector being +1 unit in certain dimensions and 0 in others). Take that random point in the lattice spanned by the basis (both f and h) by taking a random combination of the vectors in h (i.e. perform $r \times h$) and add the message to that. Applying mod$q$ makes it difficult to invert e over the ring to produce m, even knowing h and q.

3.2.5 How Decryption Works

1. Calculate a polynomial a such that $a = f \times e (\mathrm{mod} q)$

   Coefficients of a are in the range $[-\frac{q}{2}+1, \frac{q}{2}]$

   q is publicly known and f is from the private key

2. Calculate a polynomial b such that $b = a (\mathrm{mod} p)$

   p is from the public key

3. Obtain the original message $m = f_p \times b (\mathrm{mod} p)$

   f_p is from the private key, b is taken from the previous step, p is publicly known

Finding polynomial inverses in a ring as in the example in section 3.2.1 becomes explicitly relevant in the decryption process.

3.2.6 Other Characteristics of NTRU

In cryptography and even in information security as a whole, a trade-off usually has to be made between the speed and functionality of a system, and the security of that system. It is the same with NTRU; to make NTRU more secure, the length of the keys needs to be bigger, meaning that the key space is bigger and that brute-force and other attacks are harder to achieve. However, by increasing the key length, it takes more computing time to generate keys, encrypt messages, and decrypt messages since larger numbers are dealt with in computation. With RSA, NIST recommends an RSA key size of 3072 bits to be secure past 2030. The key sizes for NTRU recommended by Security Innovation is $N = 251$, $p = 3$, $q = 128$ for standard security, and the key size in bits depends on the particular implementation of NTRU, as keys can be represented in many formats. In the implementation considered in this project, a private key f created for $N = 251$ and stored as a .npy file (a serialised numpy array object) was 906 bytes.

NTRU has been found to be faster, in terms of time elapsed encrypting and decrypting, than RSA, while providing the same level of security. It takes $O(N^2)$ operations, with some sources citing $O(N \log N)$ to encrypt and decrypt in NTRU [28], whereas RSA takes $O(N^3)$.

Although comparing cryptosystems is not directly within the scope of this project, it is helpful to know for assessing the practical uses of NTRU that it appears to perform better than a main competitor in public-key cryptography.

3.3 Experiment 1: Sage Attack Experiments

The LLL attack as shown by LatticeHacks indicates that the attack is successful depending on the values set for q and d; there is a trade-off between successes in decryption and failures in LLL attacks depending on these two values. A difference in their implementation of NTRU in Sage and jkrauze's implementation in Python is that LatticeHacks chose to keep d the same value for f, g, and r, whereas in jkrauze's implementation, d_f is $\lfloor \frac{N}{3} \rfloor$, and d_g and d_r are $\lfloor \sqrt{q} \rfloor$). To discover how successful this attack is against jkrauze's implementation chosen, this attack was applied.

The value to set for q such that (1) the attack as done by LatticeHacks is successful and (2) q is closest to the actual value recommended by jkrauze in the README for $N = 167$ (which is the old standard considered secure by Security Innovation), was determined by experimenting with various q values in the LatticeHacks version, where the parameters other than q were set to the standard security values of $N = 167$, $p = 3$, and $d = 110$. d was set to 110 since d_f in jkrauze's program would be $\lfloor \frac{167}{3} \rfloor = 55$ and d in the LatticeHacks code is the number of *both* +1 and -1 coefficients, not each. As it appeared that a q value of 2^{14} produced a successful LLL attack in the LatticeHacks Sage code, q was set to 2^{14} when conducting the experiment.

It was also found when conducting this preliminary experiment that a sufficiently low d led to successful LLL attacks.

The experiment is as follows: generate a public-private key pair as usual except with q as 2^{14} rather than 2^7, encrypt a message with the public key to produce ciphertext c, perform the SageMath LLL attack on the public key, and decrypt c with both the true private key and the key found via LLL to see if each matches the true plaintext. Results will be shown in the next section 4.

3.4 Experiment 2: Parameter Choices for Decryption Successes

Exploring combinations of N, p, and q for the best security has been examined extensively, but d has not received the same attention. To visualise the effects of d on NTRU operations as well as to understand the cryptosystem on a deeper level, an experiment was performed by having separate values of d for f, g, and r. To reiterate, a lower d value results in less -1 and +1 coefficients in the polynomial, and a higher one results in more of these coefficients. Doing this may shed light on how timing on NTRU operations is affected and which combinations make decryptions fail, as well as how these results change as N increases. The results of this experiment can help in understanding the reliability of NTRU, as a cryptosystem should be able to decrypt ciphertext consistently and without failure and, given the ability to perform an LLL attack with a low d value as shown with the LatticeHacks attack, a

cryptosystem should be resistant to mathematical attacks. Additionally, having an idea about how performance scales as N increases can help in knowing what to expect about the security and practicality of NTRU as N increases, and whether improvements the current implementation can be made by changing how d values are determined given N, p, and q.

This experiment was performed on a simplified but logically equivalent version of the jkrauze implementation to make acquiring results easier. The results were obtained by observing which combinations of d values were the minimum needed to achieve a sufficient decryption success rate, which combinations achieved decryption failures, and the approximate combinations in which the success-fail rate was about 50-50.

The procedure to get the 100 percent decryption success results is as follows: for each combination of high/med/low for two d values, start at the most demanding setting of the third d value in question (such that it is the most difficult it can possibly be to consistently be successful at decrypting), and decrease this value until no failure occurs in 25 rounds of key generation, encryption, and decryption. As soon as a decryption failure occurred, decrease the value in question.

The procedure to get the 0 percent results is similar to the 100 percent results, except that rather than start from the most difficult setting for decryption to succeed and making that setting easier, start from the easiest setting and increase the difficulty until 25 decryption attempts in a row result in failures. A more difficult setting at that point is not very likely to produce many successes in decryption.

The procedure to get the 50 percent results is as follows: in the cases where such results were not obtained from the previous two procedures, start at the most difficult setting for 25 decryptions in a row to succeed according to the results from the 100 percent experiments, and decrease this value until the ratio of successes and failures when decrypting is between a 10-15 ratio and 12-13 ratio.

High, mid, and low values for each d was determined for each N as such: high refers to the largest possible d value for that N, low refers to the lowest possible value for d which is 1, and mid refers to the average of the high and low values. For example with $N = 167$, the highest value that d can be is 83 since a polynomial of degree 166 can have at most 83 coefficients of the value of -1 with 83 coefficients of value $+1$. Since the lowest possible value for d is 1, the "mid" value for $N = 167$ is the average of 83 and 1, which is 42.

It would be more accurate to perform this experiment with 100 rounds instead of 25, but the computing resources needed to effectively carry out the experiment as such was not accessible, and acquiring use of a supercomputer occurred at a later point in this project.

Where patterns emerged and where a particular combination of parameters can only result in failures or successes in decryption no matter the variable in question, the appropriate value for the variable in question could be estimated, thus reducing time obtaining results.

3.5 Experiment 3: Brute-Force Estimates

It may be of interest to estimate the classical computing power needed to brute-force the private key, given the public key, in a certain amount of time. To get a more realistic estimate than relying on mathematical speculation using computational complexities, the amount of decryptions per minute was tested on a high-performance computer.

The supercomputer provided by Supercomputing Wales, which Cardiff University students are eligible to access, was utilised as the student does not have access to a high-performance computer otherwise. To access the supercomputer and run programs for this project, approval was acquired from Supercomputing Wales. The resources requested were 32GB of RAM, 8GB of home storage, and 8GB of scratch storage.

3.6 Exploring Quantum Attacks Against NTRU

Quantum computing was explored towards the end of the project. These weeks were dedicated to grasping the capabilities and advantages of quantum computing, how quantum computation works, and how to program a quantum computer. The end goal was to gain a better understanding of how secure NTRU is against quantum computers of even thousands of qubits, despite the challenges of being in the very early stages of quantum.

The learning process included taking from a variety of sources to understand the basics of quantum computing. This included completing a Udemy course by University of Massachusetts teacher and software engineering consultant Kumaresan Ramanathan [29] as an introduction to quantum computing principles and perusing the Quantum Country interactive essays [30] to dive deeper into the mathematics of the topic and to gain another perspective of it.

IBM Qiskit [31] was examined and official tutorials from IBM [32] were followed to see what can be done currently in quantum programming. According to this guidance, this how a typical quantum program is structured at the moment: import packages, initialise variables, add gates, visualise the circuit, run the circuit (either through simulation or on IBM's quantum computers accessible through the cloud), and visualise the results. Most of the procedure is similar to constructing a classical program. Visualising the results involves seeing how many runs of the program result in, if there are two qubits, 00, how many runs result in 01, and how many in 10 and 11. When running the circuit on IBM's actual quantum computers, this process is necessary at this time due to qubit decoherence, which is expected to decrease as the technology progresses.

By completing the above activities, the foundations of quantum computation were better understood.

3.6.1 Demonstration of a Quantum Algorithm

It may be helpful to demonstrate a quantum algorithm and the benefits it offers over a classical algorithm in order to understand how a quantum computer can break NTRU more effectively than a classical computer can. This exploration is not directly relevant to breaking NTRU and may be seen as optional, but it serves to enhance all mentions of quantum computing in this report by providing a clear example of quantum computation, and allows the student to evidence independent learning.

An algorithm that demonstrates this benefit is the Bernstein-Vazirani algorithm, which is explored in the Qiskit YouTube tutorial series [32], and whose Qiskit code is included in this thesis submission. The algorithm addresses the following problem: we are given a hidden sequence h of n bits. We are also given an oracle that takes as input a sequence s of n bits, performs the operation of $s \oplus h$ where $\oplus$ denotes the bitwise inner product $\bmod 2$, and returns that single bit of output. We are to find the unknown number h using the oracle. A classical computer can guess the number in n queries to the oracle, but a quantum computer can guess the number in one try. For example, if h is 6 bits long and is '101001', a classical computer can query '000001' to the oracle and receive 1, then query '000010' to receive 0, and so on until '100000' is queried. Piecing together the outputs from the oracle queries will return '101001'. The following quantum circuit demonstrates how a quantum computer would solve this instance of the problem: Quantum circuits are often represented

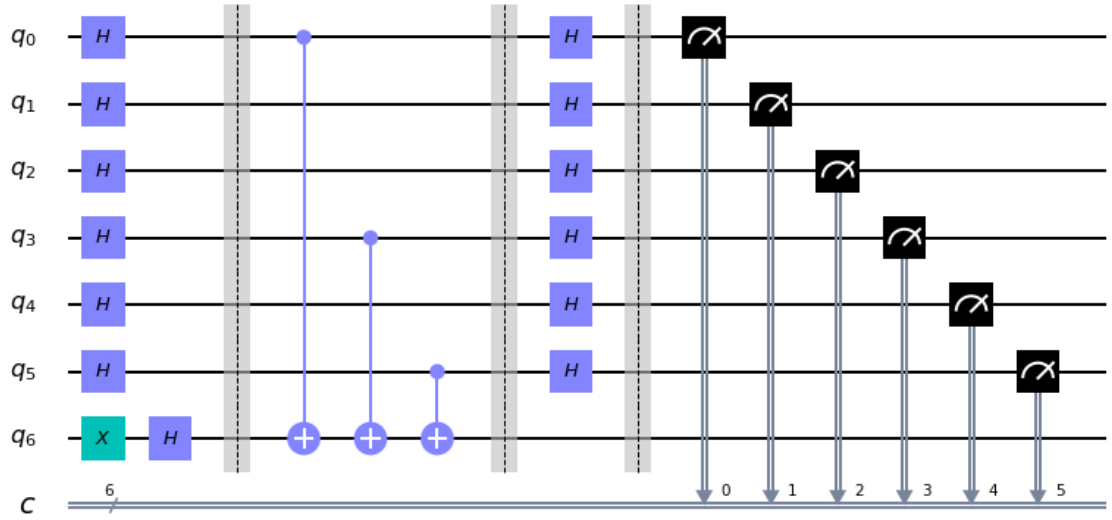

Figure 3.4: The Bernstein-Vazirani quantum circuit for an instance of the Bernstein-Vazirani problem

in a timeline format as in 3.4. In this circuit, there are seven qubits labelled q_0 to q_6. There are also six classical bits that will be used for storing the outputs of the qubits. These classical bits are denoted with C at the bottom of the diagram, with the double line and the 6 indicating that there are six such classical bits.

The purple and teal icons on the lines of the timeline indicate the application of quantum logic gates (the equivalent of AND, OR, etc. gates in classical computing) to the qubits. The NOT gate, labelled as X switches a qubit from state $|0\rangle$ to state $|1\rangle$ and vice versa. The Hadamard gate, labelled as H, puts a qubit that is in state $|0\rangle$ to the superposition state $|+\rangle$, and puts a qubit in state $|1\rangle$ in the superposition state $|-\rangle$. The same gate also reverses that state change: the H gate puts a qubit that is in state $|+\rangle$ to the superposition state $|0\rangle$, and puts a qubit in state $|-\rangle$ in the superposition state $|1\rangle$. Although it is not explicitly stated in the circuit diagram, the qubits at the beginning of the timeline are in the state $|0\rangle$.

The notation of $|\phi\rangle$, known as Dirac or Bra-ket notation, is commonly used to conveniently represent the state of a qubit. The state of one qubit can be represented as a vector with two elements: one for representing the probability of measuring the qubit to be in the quantum equivalent of 0, and another for representing the probability of measuring to 1. For example, $|0\rangle$ is the same as the vector $\begin{bmatrix}1\\0\end{bmatrix}$. There is a 100% probability of measuring the qubit to be 0, and a 0% probability of measuring the qubit to be 1. The state of a qubit in superposition can be represented as $|+\rangle$ and $|-\rangle$, or $\begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}$ and $\begin{bmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{bmatrix}$ respectively; these states have equal probability of measuring to state 0 or 1, although $|+\rangle$ and $|-\rangle$ are distinctly different states. This will become relevant in understanding why the circuit works and why quantum computing can be helpful in breaking NTRU.

The quantum gate stretching across two qubits indicates a CNOT (Controlled NOT) gate, where the solid circle signifies the control qubit and the circle with the open plus sign signifies the target qubit. A CNOT gate operates like an if statement in a classical program: if (control qubit measures as 1) then (NOT the target qubit). In this circuit, applying these CNOT gates is equivalent to querying the oracle.

The gauge-like icon at the end of the timeline for the qubits signifies a measurement, and in this particular circuit, the outputs from qubits q_0 to q_5 are stored in classical bits c_0 to c_5.

Put together, this circuit takes an n number of qubits, puts them into superposition, queries the oracle, puts the qubits back into a state of either $|0\rangle$ or $|1\rangle$, and measures the state of those qubits. It takes just one run of this circuit to guess the hidden binary number h.

Key to understanding why this circuit works is understanding what querying the oracle does to the qubits. Following the Quantum Country series of essays, it was of interest to complete the following mathematical exercise in understanding quantum logic gates: indicate how a CNOT gate functions in the case where the control qubit is in the state $|+\rangle$ and the target qubit in $|-\rangle$. In this application of the CNOT gate, the target qubit is not altered but the control gate is, which is counterintuitive. A quantum gate can be represented as a matrix, with the application of a gate equivalent to multiplying the state of the qubit(s) with that matrix.

We can rewrite a quantum state $\begin{bmatrix}\alpha\\\beta\end{bmatrix}$ as $\alpha\,|0\rangle + \beta\,|1\rangle$. If we have two single-qubit states

$\alpha|0\rangle + \beta|1\rangle$ and $\gamma|0\rangle + \delta|1\rangle$, then their combined state is $(\alpha|0\rangle + \beta|1\rangle)(\gamma|0\rangle + \delta|1\rangle) = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$, which can also be represented as $\begin{bmatrix}\alpha\gamma\\ \alpha\delta\\ \beta\gamma\\ \beta\delta\end{bmatrix}$. If the control qubit is in $|+\rangle$ and the target qubit is in $|-\rangle$ like in figure 3.5 then the above becomes $(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle) = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle$.

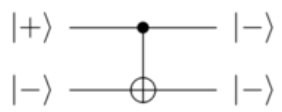

Figure 3.5: Application of a CNOT gate that changes the control qubit but not the target qubit

Applying the CNOT matrix $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$ to $\begin{bmatrix}\frac{1}{2}\\ \frac{1}{2}\\ \frac{1}{2}\\ -\frac{1}{2}\end{bmatrix}$ yields $\begin{bmatrix}\frac{1}{2}\\ \frac{1}{2}\\ -\frac{1}{2}\\ \frac{1}{2}\end{bmatrix}$.

$$\begin{bmatrix}\frac{1}{2}\\ \frac{1}{2}\\ \frac{1}{2}\\ -\frac{1}{2}\end{bmatrix} \cdot \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\\ \frac{1}{2}\\ -\frac{1}{2}\\ \frac{1}{2}\end{bmatrix}$$

The state the qubits must be in to form that combined state vector is $|-\rangle$ and $|-\rangle$ as $(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle) = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$. The control qubit is changed but the target qubit remains unchanged. This is what is happening in the Bernstein-Vazirani circuit to each qubit in the position of the hidden sequence s where there is a 1. The queries to the oracle change the qubits, which cannot be the case in a classical computer since classical bits cannot be in superposition nor entangled.

This brief introduction to quantum circuits explains how a quantum computer can perform computations in a way a classical computer cannot, and this explanation may help in appreciating how a quantum algorithm can help render NTRU insecure more efficiently than a classical computer can.

Again, there is a need for abstraction when it comes to programming with a quantum computer. It became clear that there would be limited gain from looking further at, as more mathematical knowledge and more time would be needed.

Following guidance from the project moderator, it was understood that time would be best spent looking at what methods within attacks can be replaced or sped up by current quantum algorithms. The end of the project was spent on this task, and the findings are explored in the next sections.

The approach taken to this project could be summarised as choosing a version of lattice-

based cryptography to analyse, understanding the lattice-based cryptosystem, performing some experiments with that implementation, learning the basics of quantum computation from software engineering and cryptographic perspectives, and investigating how robust that lattice-based cryptosystem will be in the foreseeable future of quantum computing.

4 Results

To discover how robust this application of a lattice-based cryptosystem is, three experiments were run, and current quantum algorithms were examined against known attacks against NTRU. The results of these undertakings are displayed in this section.

4.1 Experiment 1: Sage Attack Experiments

The attack was successful against the implementation of NTRU chosen, though with a large q. This result matches that found by the LatticeHacks team. A public-private keypair with parameters $N = 167$, $p = 3$, $q = 2^{14}$ was generated, the LLL attack in SageMath as written by LatticeHacks was applied to the public key h to produce $h\prime$, and a message encrypted by h was able to be decrypted successfully by both the true private key f and the fake private key $h\prime$.

4.2 Experiment 2: Parameter Choices for Decryption Successes

These experiments were run on a 2013 MacBook Air running Ubuntu 18.04, with 4GB 1600 MHz DDR3 RAM and a 1.3 GHz Dual-Core Intel Core i5. The results of running the program are included in the appendix, with appropriate graphs included in this section for $N = 167$. The full results across varying N values, including timing of key generation, encryption, and decryption, are included in the appendix A.1 A.2 A.3.

N	d_f	d_g	d_r	Key Generation Time (sec)	Encryption Time (sec)	Decryption Time (sec)
167	55	11	11	5.688	0.270	0.266
251	83	11	11	12.942	0.509	0.596
347	115	11	11	25.310	1.098	1.125

Actual average operation times for $N = 167, 251, 347$

The d value that affects timing of key generation the most appears to be d_f. d_r seems to have the greatest effect on decryption error and security since, as shown in 4.3, d_g can be of a low, medium, or high value, but depending on d_r decryptions can become reliably successful or reliably unsuccessful, no matter what d_f is.

As N increases, timings for all operations increase, which is to be expected since a higher dimension of the lattice requires more computations.

This experiment was conducted to understand how different parameters work with NTRU, and viewing these results with different combinations of values for those parameters can indicate how secure the suggested standard parameters are. These results are explored in 5.

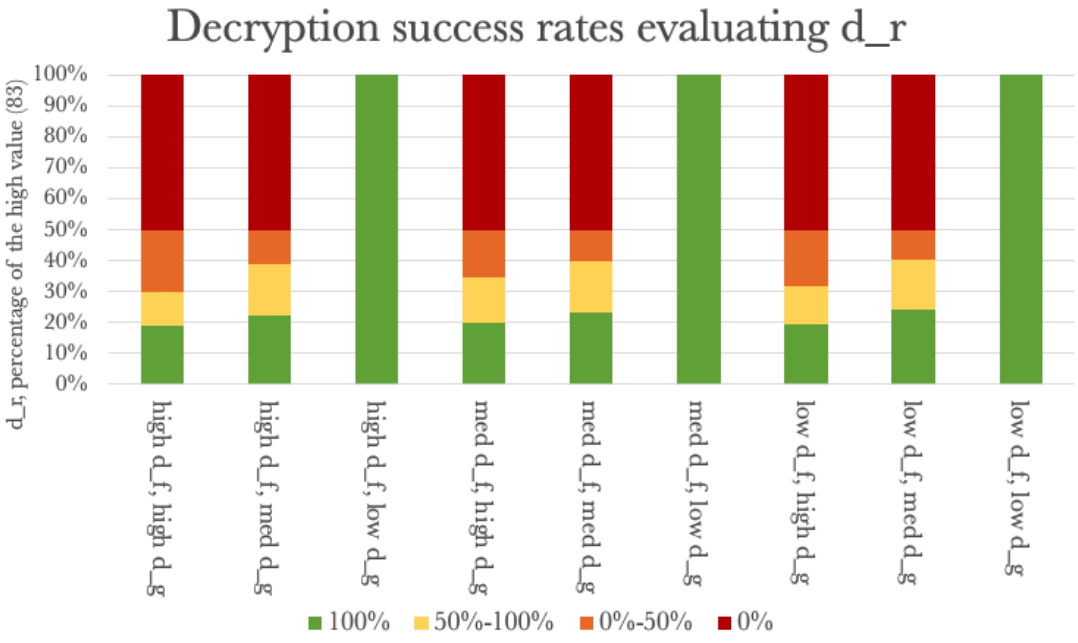

Figure 4.1: Evaluating d_r

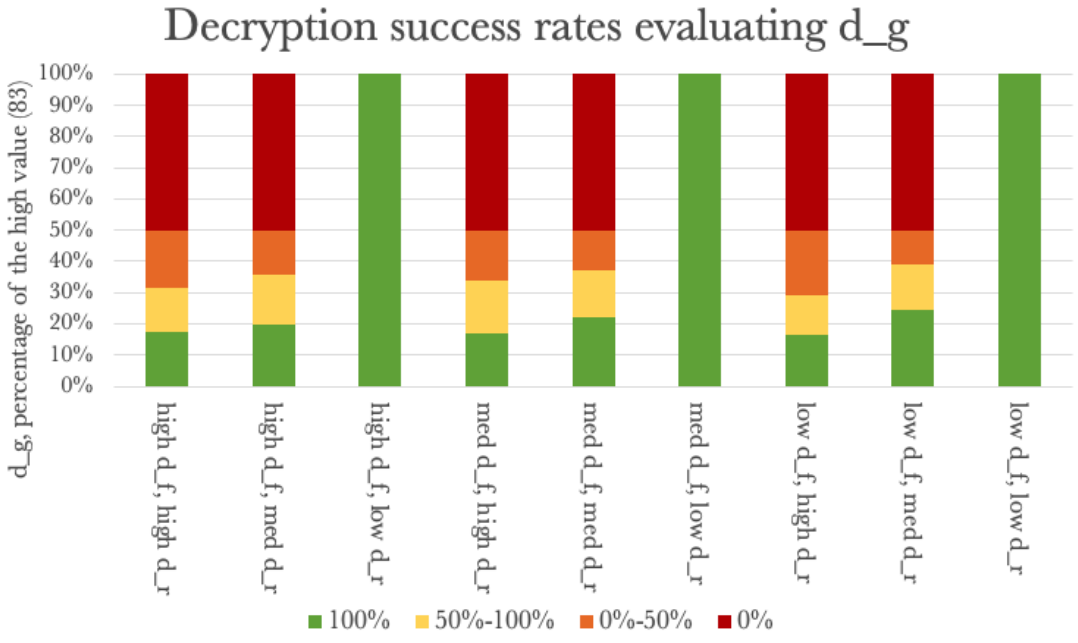

Figure 4.2: Evaluating d_g

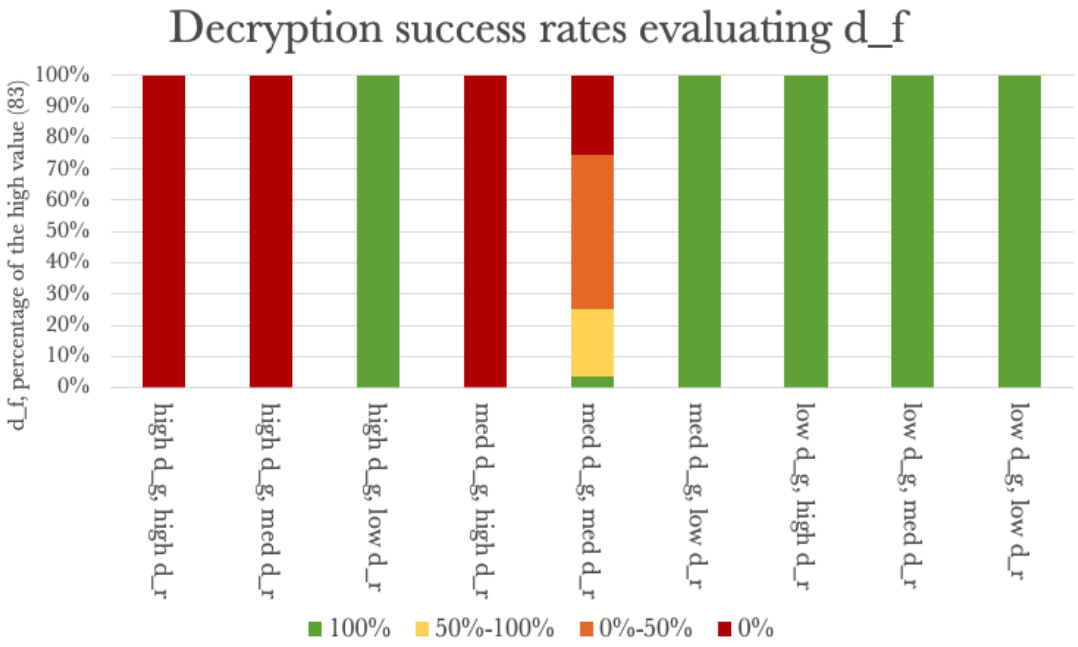

Figure 4.3: Evaluating d_f

4.3 Experiment 3: Brute-Force Estimates

The results of number of decryptions per minute, one way of verifying that an NTRU key is the corresponding private key given a public key, are shown.

The graph suggests a exponential decrease in decryptions per minute, which makes sense as decryption time increases exponentially as N increases.

The number of possible keys for this implementation of NTRU is all permutations of coefficients where the number of -1 and +1 coefficients each are $\lfloor \frac{N}{3} \rfloor$. For example, for $N = 4$ (not possible for actual use since it is not a prime number) the number of keys possible is the number of permutations of (0, 0, 1, -1), which is 24, divided by the individual permutations of 0, -1, and 1, which in this case is 2 for the individual permutations of 0. This division is performed to account for the insignificance of order for individual coefficients. In other words, (0a, 0b, 1, -1) and (0b, 0a, 1, -1) both refer to (0, 0, 1, -1) or, in polynomial form, $x - 1$, so (0, 0, 1, -1) should not be counted twice. The number of possible keys given this practice for varying N is shown in the table below.

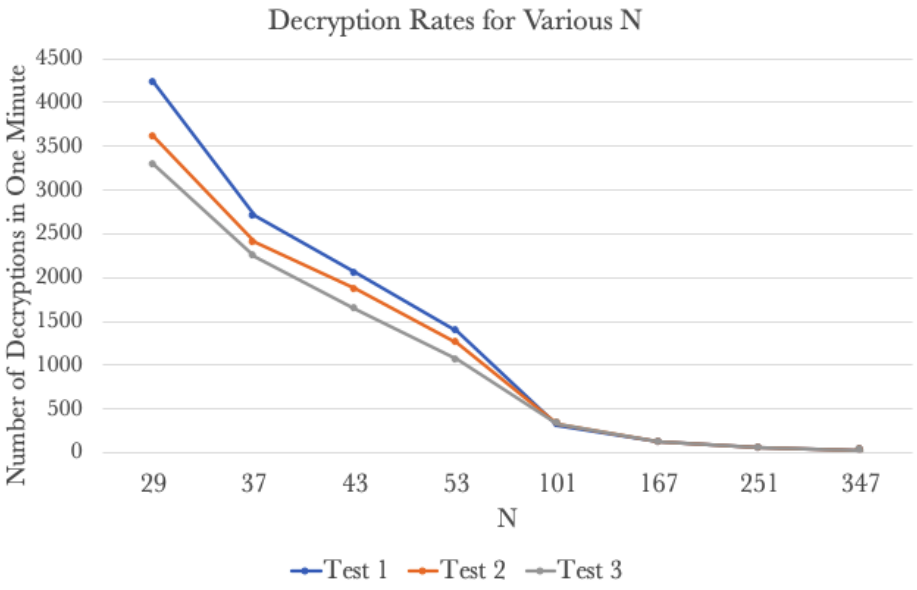

Figure 4.4: Decryptions per minute for varying values of N

N	Number of Possible Keys
4	12
5	20
6	90
11	9240
13	90090
17	4900896
19	46558512
23	2.80×10^{9}
29	1.68×10^{12}
53	2.78×10^{23}
101	1.21×10^{46}
167	2.30×10^{77}
211	1.83×10^{98}
251	1.85×10^{117}
347	8.56×10^{162}

Number of possible keys for varying N, in jkrauze's implementation of NTRU

The key space grows quickly. For the standard security parameters of $N = 251$, $p = 3$, and $q = 128$ it appears infeasible to reliably brute-force the private key even with many devices that have the computing power used in the brute-force experiments. The mathematics to estimate how many such computers it would take to brute-force an $N = 167$ key is given in the discussions section 5.

4.4 Quantum Algorithms

By comparing known attacks to current quantum algorithms as listed in Quantum Algorithm Zoo (a comprehensive catalogue of quantum algorithms) by Dr. Stephen Jordan [33], it appeared that the best direction for research to undermine the security of NTRU is to design a quantum algorithm that performs lattice reduction more efficiently. This will be discussed in greater deal next section.

In this section, the results of exploring NTRU as proposed in the previous section were displayed and explained. The next section goes deeper into discussing the significance of these results.

5 Discussion and Evaluation

In this section, discussions are made about what the experiments performed indicate about the security of this implementation of NTRU against classical methods. Additionally, conjectures are made regarding security against quantum attacks.

5.1 Classical Attacks

As suggested by LatticeHacks [13], a trade-off was indeed found between rate of failures in decryption and rate of successes of lattice reduction methods. By selecting the correct values of N, p, q, and, as explored in the parameter choices experiment, d, this attack can be avoided. This may not be the same for other lattice-reduction methods that have a smaller approximation factor (i.e. are more precise), however.

The parameters suggested by jkrauze of $N = 167$, $p = 3$, $q = 128$, which automatically produces $d_f = \left\lfloor \frac{N}{3} \right\rfloor = 55$, $d_g = \left\lfloor \sqrt{q} \right\rfloor = 11$, and $d_r = \left\lfloor \sqrt{q} \right\rfloor = 11$ in this adaptation of NTRU, is suitable for a balance between resistance to the LLL attack and decryptions. However, it may be better to manually set d_f, d_g, and d_r to have more control over this balance for varying N. For example, with $N = 11$, $p = 3$, $q = 128$, which were used when performing tests on smaller dimensions early in the project, run-time errors would occur as $\left\lfloor \sqrt{q} \right\rfloor = \left\lfloor \sqrt{128} \right\rfloor = 11$ and even for $q = 64$, $\left\lfloor \sqrt{q} \right\rfloor = \left\lfloor \sqrt{64} \right\rfloor = 8$ are not valid d values since $N = 11 < 8 \times 2 < 11 \times 2$.

If an attacker has knowledge of the d values, this drastically reduces how many possible keys there are compared to not knowing these values, and makes brute-forcing easier. Although it is still infeasible to perform this attack for large N as shown in the next subsection, it is additional security to have a slightly more random element in how many +1, 0, and -1 coefficients there are in the private key.

As stated previously, some considerations in obtaining the results include not testing with enough rounds and not testing a larger combination of d values (as opposed to low,

medium, and high for each). It is also worth noting that the operations of encryption and decryption for all combinations of N and d values were done on a message that was the same length. Some further observations on the security of this implementation of NTRU can be made by performing the tests for varying message lengths. However, the obtained results may be sufficient to say that the current parameter choices are safe.

LLL works with a large q because a large q allows for approximation, where smaller q values require more exact approaches to the SVP. As suggested by the Sage LLL attack, decreasing q and increasing N leads to increased security in NTRU. However, decreasing q means a higher chance of decryption failures. LatticeHacks suggests to choose parameters such that $\frac{q}{2} > 4d$. Security Innovation's submission of NTRU to NIST [18] has a proposed parameter set of $N = 743$, $q = 2048$, and $d = 247$. This is the same d value that jkrauze's implementation would produce for that N (as $\left\lfloor \frac{743}{3} \right\rfloor = 247$), and $\frac{2048}{2} = 1024$ is indeed greater than $4 \times 247 = 988$. As suggested by the Sage attack and the d experiments and as suggested by LatticeHacks these are good choices for achieving the balance between providing security against the LLL attack and minimising decryption failures.

It makes sense to separate d_f from d_g and d_r since d_f has the largest influence over key generation time among these three values.

As N increases, d_f increases proportionally. For d_g and d_r, one should increase q accordingly if $\left\lfloor \sqrt{q} \right\rfloor$ is to be used for these two values. It may be advisable to make d_g and d_r dependent on N as well rather than q so that the implementation can work on smaller dimensions and a sufficiently large q, or even make the d values settable by the parties taking part in encryption for more control over the security and decryption failures. The exact effects on security will have to be looked at further, though these are some directions to take with exploring parameter choices for this implementation of NTRU.

When it comes to brute-forcing the private key, the number of possible keys when $N = 167$ is 2.30×10^{77} and the number of decryptions per minute when $N = 167$ for the resources of the supercomputer requested is about 125. The number of minutes in a year is 525,600, meaning that the number of decryptions per year is 65,700,000. How many such devices are needed to brute-force an $N = 167$ NTRU key in a year? This can be put as:

$$[\#ofdecryptionstotal] = [\#ofdecryptionsperyear] \times [\#ofyears] \times [\#numberofcomputers]$$

$$2.30 \times 10^{77} = 65,700,000 \times 1\ times[\#numberofcomputers]$$

$$[\#ofcomputers] \approx 3.50 \times 10^{69}$$

This number of devices required is impractical. For this reason, NTRU with $N = 167$ is suitably secure against this type of brute-force attack.

It should be noted that, in addition to decrypting a particular ciphertext when given the plaintext, a potential f can be verified as the actual f by seeing if $f\prime \times h (\mathrm{mod} q)$ has small integer coefficients.

The classical experiments allowed the student to demonstrate programming skills as required for this module and learning about NTRU on a deeper level, while also demonstrating the scientific method and analytical thinking. A full brute-force attack for a small N to get a more accurate timing estimate and more advanced attacks would have been carried out with more time.

5.2 Quantum Algorithms

An instinctive approach to breaking NTRU using quantum computation is to brute-force the private key using quick search. This might involve representing all possible private keys for a given N, p, and q as qubits in superposition, and calls to an oracle as the process of seeing if decrypting a message with that potential key produces comprehensible plaintext, or if $f\prime \times h(\mathrm{mod} q)$ has integer coefficients in the range [-1, 1]. For $N = 167$, the number of qubits in superposition needed to represent all possible combinations of coefficients for f is 257, as $\lceil \log_2(2.30 \times 10^{77}) \rceil = 257$; this number may not seem incredibly large since 257 classical bits can be used to represent a very small amount of data, but only if the decoherence problem is solved and these qubits are manipulated smartly can this be an advantage.

The next method to try would be to speed up classical SVP methods or parts of those approaches using quantum computation. There is a conceptual quantum version of the LLL algorithm published [34] but the number of qubits required makes this approach infeasible. Grover's algorithm has been explored in speeding up sieving and saturation approaches to the SVP [35]. These are approaches that remove possible solutions from an initially full list or add possible solutions to an initially empty list respectively. However, this solution in itself does not provide the speed-up required for polynomial-time results. However, this same approach may indicate further speed-ups of classical SVP solvers.

Search will not be enough to completely undermine NTRU in the same way Shor's algorithm does RSA. If search is the only quantum attack, then for both NTRU and RSA, increasing the key size will be sufficient. The question to answer is whether or not the SVP can be solved in polynomial time. As a result, the best approach for breaking NTRU with quantum computation is a lattice basis reduction algorithm, ideally using exact methods rather than approximations for the shortest vector.

Oded Regev has made significant contributions to this topic. Many of the papers addressing the security of lattice-based cryptosystems have cited his 2003 paper [36]. In the paper he has shown that the SVP can be solved efficiently by a quantum computer if there is a quantum algorithm that solves the hidden subgroup problem for the dihedral group. In Regev's solution, the lattice space is split into subspaces such that in each subspace there are only two points. However, the problem remains difficult to solve in the nearly two decades since this publication.

The hidden subgroup problem (HSP) is a generalisation of problems such as the SVP; in the case of lattice-based cryptography, the non-abelian variant of the problem is relevant as opposed to the abelian since dihedral groups of orders greater than 3 are non-abelian. Solving the HSP for non-abelian groups would solve the SVP, meaning that a given plaintext encrypted with a given h in NTRU can be decrypted by performing the algorithm on that h

to find a suitable f, no matter the q value, which differs from LLL's approximation methods.

Now the question to answer becomes: is there a quantum algorithm that can solve the dihedral HSP efficiently? If such an algorithm is created, then the foundations of lattice-based cryptosystems such as NTRU are compromised.

Kuperberg found a time $2^{O(\sqrt{\log N})}$ algorithm for finding a hidden subgroup of the dihedral group [37], and Regev improved on this to use polynomial space [38]. The classical complexity for the same problem is $2^{O(\sqrt{N})}$.

If there are efficient quantum algorithms for the HSP, that implies there are efficient quantum algorithms for the SVP. Given a group of neighbouring points in a lattice, getting from one point to another takes more steps using the vectors in h than f, since f is a basis of the same lattice of h, but with shorter vectors. There can be many points produced by f contained in the area produced by h. This provides an intuitive view of how the HSP relates to the SVP.

On the other hand, there is no need to worry about NTRU being safe against quantum computers in the thousands of qubits if physical implementations of these computers cannot be scaled up to have a useful amount of fault-tolerant qubits in the first place. All this speculation is for naught if quantum computers cannot be effectively scaled up from the current ~ 50 qubits or cannot maintain the state of their qubits.

An efficient solution does not seem attainable in the near future but it does not seem unrealistic that some great minds will eventually create a method that fully realises the ideas proposed by Regev and uses quantum computation to solve the SVP efficiently.

5.3 Strengths and Weaknesses of this Approach

The work carried out indicates that this implementation of NTRU is safe from brute-force attacks and lattice reduction using LLL if parameters are chosen correctly. These results match those found in other published works. Other attacks were not explored due to time constraints; what can be done further is discussed in section 6.

Due to quantum computing being early in development, a substantial weakness of this approach is that quantum attacks cannot be fully carried out. Experimentation on NTRU with quantum algorithms can only be done using theory and mathematical conjectures at this point in time since no quantum algorithms currently exist to properly experiment with quantum attacks against NTRU. It could be interesting to attempt to implement something like Regev's approach using quantum simulators and current quantum programming languages such as Q# or using Qiskit, using small lattice dimensions and keeping black-box operations hidden.

In the initial plan of the project, it was stated that an attempt would be made to recover the private key from the public key. Through lattice reduction, this attack was carried out, since a suitable private key was obtained from applying this operation on the public key. An improved attempt, such as more precise approximation, at this attack could be made with

use of quantum algorithms.

6 Future Work

This section addresses what can to be done to improve the security and practicality of NTRU.

6.1 Classical Attacks

For this particular implementation of NTRU, it will be worth ensuring that the padding method used in encryption adheres to current standards, such as NAEP [39]. In addition to padding improvement, the speed of key generation, encryption, and decryption leaves more to be desired. More efficient key generation, for example, could be improved upon using the method as described in [40], which takes advantage of field norms, or that of [41], which computes the inverse of a polynomial in the ring more efficiently.

It may be worth examining how random the polynomial generator (which utilises the numpy random function) truly is, since if an attacker can guess the seed of this function and knows the values of N, p, and q, he or she can potentially find f. The random polynomial generator suggested by Security Innovation [42] may provide a better solution, though whether or not this is the case is what could be explored.

Side-channel attacks such as those in [43] are another avenue to explore. Cryptosystems should be designed to resist attacks that take advantage of power consumption, timing information, or electromagnetic leaks.

The classical attacks covered in the background section 2 can be explored against the chosen implementation of NTRU, though many of the attacks mentioned can be mitigated with appropriate parameter choices and padding schemes.

6.2 Quantum Attacks

As quantum computers become more fault-tolerant, as more layers of abstraction are created, and as more quantum algorithms are developed, we can get a better idea of how truly robust NTRU is against quantum computers. The most direct course of action to take when creating quantum algorithms against lattice-based cryptography is to create one that solves the dihedral hidden subgroup problem, though other directions include designing quantum algorithms to speed up search and other areas of classical lattice reduction methods and improving on lattice-reduction methods quantum or otherwise. According to [42], the best attacks on NTRUEncrypt tend to utilise both lattice reduction and combinatorial search. As an alternative to looking solely at solving the DHSP or speeding up classical lattice reduction, this hybrid approach can be taken.

Regev showed that solving the unique-SVP (a variant of the SVP relevant to NTRU) is attainable if there is an algorithm that solves the dihedral HSP, and Kuperberg found a speed up for this problem. Their work is a step towards solving the SVP in direct relation to NTRU and indicates that it may be a matter of when rather than if a quantum algorithm

will be created that solves the SVP efficiently enough to render NTRU vulnerable even with larger keys, like Shor's does with RSA.

After the finalist(s) for the NIST PQC competition are announced, similar research on attacks like those mentioned in this paper will continue to be performed on those cryptosystems. It is likely that this research will not stop until the underlying hard problem of the cryptosystem is solved, breaking the system entirely.

Quantum cryptography (utilising quantum mechanical properties to perform cryptographic tasks, as opposed to *post*-quantum cryptography which is about finding a classical cryptosystem that is robust against quantum attacks) such as the BB84 quantum key distribution scheme [44] may be the best option for a cryptosystem that resists quantum attacks. It may take a quantum-based cryptosystem to provide the security needed to prevent quantum-based attacks.

It would be most productive to attack the version of NTRU that Security Innovation submitted to NIST for the PQC standardisation project. This version and the suggested parameters are what are thought to be the most secure of this cryptosystem, and attacks on this version will more effectively reveal what to improve on NTRU next. There are several routes to attack NTRU and in particular the implementation chosen, though it will be some time before its security can be properly tested against quantum computers.

7 Conclusion

This paper is an exploration of a lattice-based cryptosystem and how resistant it is to both classical and quantum attacks. For classical computing, this has been done through replicating LLL lattice reduction, experimenting with parameters, and examining the feasibility of brute-force attacks on the chosen implementation of the NTRU lattice-based cryptosystem.

Lattice-based cryptosystems are significant as they are a candidate for post-quantum cryptosystems, which are thought to be secure against quantum computers of even thousands of fault-tolerant qubits, unlike current popular public-key cryptosystems such as RSA. By assessing and confirming the robustness of such a cryptosystem, it can be assured that if quantum computers have sufficient capabilities that encrypted data remains confidential.

Based on the current status of published attacks against NTRU, the cryptosystem remains safe from classical and quantum attacks provided that parameters are carefully chosen; it is infeasible for either classical or quantum attacks to break the cryptosystem. For quantum computing, current research directions were examined to estimate how likely it is that a quantum algorithm will be devised that will efficiently undermine the security of NTRU, or what would need to be done for that to be the case. It was determined that NTRU remains safe against current attacks and technologies, but it does not seem unreasonable that a quantum algorithm will eventually be designed that will compromise the cryptosystem in the way Shor's algorithm does RSA.

The objective of this project was to evaluate the robustness of a lattice-based cryptosys-

tem. As was pledged in the initial plan, this project was able to provide an overview of the cryptosystem, an overview of quantum computing, and an investigation of how resistant it is to various attacks.

8 Reflection

Writing this report has made me realise what a broad amount of work I have done over the past few months. I learned some advanced mathematics in the process of understanding how NTRU works, I researched what research has been done to attack NTRU, I learned quantum computing fundamentals, and I conducted multiple experiments to explore NTRU and its security. I have also come to appreciate the usefulness of LaTeX. My primary goals of this report were to present my work clearly and to demonstrate that I have indeed learned. It is my hope that I was able to demonstrate independent learning, application of the scientific method, perusal of academic publications, and writing in an academic context.

The most significant personal development I experienced from this project is that I am more confident taking charge of my own learning and performing independent research. I felt responsible for producing an interesting project and applying what I have learned throughout university to expand on my knowledge. This project also tested my organisational and task prioritisation skills in a way that was relatively new to me; although throughout my academics I have exercised these abilities, this is the first instance I have had to do so for such a large work. I planned so that I could keep my progress in accordance with the time plan I set at the beginning of the project, and I regularly checked with my supervisor that I was taking the right direction with what precisely I was doing for this project. Throughout the project I kept extensive notes of what I did and what I learned, why I was doing it, and what the next steps were.

I was able to apply material taught in previous modules I have taken as part of my undergraduate degree. For example, I utilised mathematical knowledge such as matrix multiplication (used in quantum computing to model the modification of the state of a qubit) taught in my first year, data processing and visualisation methods from the aptly-named second-year module, complexity theory from the algorithms and data structures module, and optimality theory (which the SVP is about) from combinatorial optimisation. Perhaps what is most significant and most relevant is the material taught in the third-year security module. The module made me curious about exploring cryptography further and prepared me for this project. As a result of the module, I was more familiar with public-key cryptography and I discovered the importance of finding quantum-resistant cryptographic solutions as the security of current public-key methods is threatened by quantum algorithms.

At one point in my project, a lecturer gave me advice that stuck with me: with projects that do not have clear deliverables (such as research-based papers) it can be easy to be disheartened during the course of the project. This was certainly the case in this project, and it was the most persistent difficulty I experienced throughout the months. I thoroughly enjoyed learning about more advanced mathematics and about quantum computing, despite my fears that I would fall out of love with mathematics over the course of the project.

Furthermore, it was *necessary* to perform plenty of preliminary research given the nature of the project. However, it seemed that I was taking in information without producing new conclusions of my own. It took me a while to become comfortable with the fact that when performing research and exploring topics that are new to me and when performing experiments as a result of this learning experience, that not all results will be interesting or productive. My optimism fluctuated between enjoying the learning process and feeling that I was not doing enough despite my efforts. It was only as I was approaching the end of the project that I saw that what I was doing has indeed led to learning and personal development. As someone who sets high standards for herself, I learned to be more compassionate towards myself and understand that the learning in itself can be enough. Although I struggled throughout this project from feeling that I was not accomplishing much, I can say looking back at my efforts over the past few months that I am satisfied that I have grown as a student.

Apart from struggling with meeting my own standards, another challenge I faced in writing this report is writing for an audience with varying knowledge about the topics presented. I aimed to discuss these topics on the level of other undergraduate computer science students, all of whom will have different levels of expertise in topics of cryptography and quantum computing. By tailoring the writing to a diversity of levels of understanding, I have more thoroughly understood the material I have been learning about as part of my project.

Although I followed my initial plan closely, I did not contribute as much to the final report over the course of the project as I had hoped. I made detailed notes and put together a few sentences every couple of weeks but I never did anything substantial. What I would do differently in my project is focus less on topics that are not very productive towards my final work such as learning about Galois fields, and redirect that attention towards looking more deeply at the current quantum algorithms that can assist in attacking NTRU. Another thing I would do that would have made this project progress further is read the post-quantum cryptography textbook by Bernstein [45] and follow the Oded Regev lattice lectures [26].

It is worth noting that I worked on this project during the COVID-19 global crisis. About halfway through the project, the United Kingdom went into lockdown to prevent the spread of coronavirus. Although my project could be done from nearly any place with a stable internet connection, I often found it difficult to work from home. My project is not properly represented without a mention that the assignment was completed under lifestyle-changing circumstances.

With regards to applying what I have gained from this project to my career after graduation, I am more prepared for further study or at least to apply a more scientific-minded way of thinking in industry. I am more comfortable with topics such as linear algebra, quantum computing, supercomputing, and, of course, lattice-based cryptography. I had not seriously considered a research-oriented career before this project, but after completing this project and with encouragement from my supervisor, I *would* like to pursue such a career, whether it be in academia, industry, or a combination of both.

I aspire to make some original and substantial contributions to the field of cybersecurity at some point in my life. This project has shown me that although I often doubt myself, I do

enjoy the process of independent learning and I have the curiosity and drive that is needed to make those contributions.

9 References

[1] B. Connolly, "Quantum computers will be commercially available in 20 years: scientist," *CIO*, 2012. [Online]. Available: https://www.cio.com/article/3493265/quantum-computers-will-be-commercially-available-in-20-years-scientist.html

[2] R. Merritt, "Moore's law dead by 2022, expert says," *EE Times*, 2013. [Online]. Available: https://www.eetimes.com/moores-law-dead-by-2022-expert-says/#

[3] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[4] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, "The impact of quantum computing on present cryptography," *CoRR*, vol. abs/1804.00200, 2018. [Online]. Available: http://arxiv.org/abs/1804.00200

[5] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "Report on post-quantum cryptography," *National Institute of Standards and Technology Interagency or Internal Reports*, 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf

[6] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, "Status report on the first round of the NIST post-quantum cryptography standardization process," *National Institute of Standards and Technology Interagency or Internal Reports*, 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf

[7] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '97. New York, NY, USA: Association for Computing Machinery, 1997, p. 284–293. [Online]. Available: https://doi.org/10.1145/258533.258604

[8] IBM. (2019) 5 in 5: Lattice cryptography. [Online]. Available: https://www.research.ibm.com/5-in-5/lattice-cryptography/

[9] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, J. P. Buhler, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 267–288. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.8422&rep=rep1&type=pdf

[10] ——, "Public key cryptosystem method and apparatus," U.S. Patent US6 081 597A, 2000. [Online]. Available: https://patents.google.com/patent/US6081597A/en

[11] Y. Chen and P. Q. Nguyen, "Bkz 2.0: Better lattice security estimates," in *Advances in Cryptology – ASIACRYPT 2011*, D. H. Lee and X. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–20.

[12] H. Lenstra, A. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients." *Mathematische Annalen*, vol. 261, pp. 515–534, 1982. [Online]. Available: http://eudml.org/doc/182903

[13] D. J. Bernstein, N. Heninger, and T. Lange. (2018) Latticehacks. [Online]. Available: https://latticehacks.cr.yp.to/ntru.html

[14] N. Howgrave-Graham, J. H. Silverman, and W. Whyte, "A meet-in-the-middle attack on an NTRU private key," 07 2003.

[15] J. Hoffstein and J. H. Silverman, *Implementation Notes for NTRU PKCS Multiple Transmissions*, 1998. [Online]. Available: https://www.onboardsecurity.com/products/ntru-crypto/ntru-resources

[16] E. Jaulmes and A. Joux, "A chosen-ciphertext attack against NTRU," vol. 1880, 08 2000, pp. 20–35.

[17] N. Howgrave-Graham, "A hybrid lattice-reduction and meet-in-the-middle attack against NTRU," 08 2007, pp. 150–169.

[18] OnBoardSecurity, "NIST post quantum crypto submission," 2018. [Online]. Available: https://www.onboardsecurity.com/nist-post-quantum-crypto-submission

[19] S. Eibenberger, S. Gerlich, M. Arndt, M. Mayor, and J. Tüxen, "Matter–wave interference of particles selected from a molecular library with masses exceeding 10000 amu," *Physical Chemistry Chemical Physics*, vol. 15, no. 35, p. 14696, 2013. [Online]. Available: http://dx.doi.org/10.1039/C3CP51500A

[20] I. Sample, "Google claims it has achieved 'quantum supremacy' – but IBM disagrees," *The Guardian*, 2019. [Online]. Available: https://www.theguardian.com/technology/2019/oct/23/google-claims-it-has-achieved-quantum-supremacy-but-ibm-disagrees

[21] E. Pednault, J. Gunnels, D. Maslov, and J. Gambetta. (2019) On "quantum supremacy". [Online]. Available: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/

[22] T. Moses, *Quantum Computing and Cryptography: Their impact on cryptographic practice*, 2009. [Online]. Available: https://www.entrust.com/wp-content/uploads/2013/05/WP_QuantumCrypto_Jan09.pdf

[23] E. Barker and Q. Dang, "Recommendation for key management: Application-specific key management guidance," *NIST Special Publication 800-57 Part 3 Revision 1*, 2015. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-57pt3r1

[24] R. LaRose, "Overview and comparison of gate level quantum software platforms," *Quantum*, vol. 3, p. 130, Mar. 2019. [Online]. Available: https://doi.org/10.22331/q-2019-03-25-130

[25] J. Krauze, "ntru: Simple python implementation of NTRUEncrypt cryptosystem," 2018. [Online]. Available: https://github.com/jkrauze/ntru

[26] O. Regev, "Lattices in computer science," 2004. [Online]. Available: https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/

[27] D. Chouinard, "Finding inverse of polynomial in a field," Mathematics Stack Exchange, 2012. [Online]. Available: https://math.stackexchange.com/q/124300

[28] C. Narasimham and J. Pradhan, "Performance analysis of public key cryptographic systems RSA and NTRU," 01 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.8669&rep=rep1&type=pdf

[29] K. Ramanathan, "QC101 quantum computing quantum physics for beginners," 2020. [Online]. Available: https://www.udemy.com/course/qc101-introduction-to-quantum-computing-quantum-physics-for-beginners/

[30] A. Matuschak and M. A. Nielsen, "Quantum computing for the very curious," 2019. [Online]. Available: https://quantum.country/qcvc

[31] IBM, "Qiskit," 2020. [Online]. Available: https://qiskit.org/

[32] Qiskit, "Coding with qiskit," 2019. [Online]. Available: https://www.youtube.com/playlist?list=PLOFEBzvs-Vvp2xg9-POLJhQwtVktlYGbY

[33] S. Jordan, "Quantum algorithm zoo," 2019. [Online]. Available: https://quantumalgorithmzoo.org/

[34] M. Tiepelt and A. Szepieniec, "Quantum LLL with an application to mersenne number cryptosystems," Cryptology ePrint Archive, Report 2019/1027, 2019. [Online]. Available: https://eprint.iacr.org/2019/1027

[35] T. Laarhoven, M. Mosca, and J. van de Pol, "Solving the shortest vector problem in lattices faster using quantum search," in *Post-Quantum Cryptography*, P. Gaborit, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 83–101.

[36] O. Regev, "Quantum computation and lattice problems," *CoRR*, vol. cs.DS/0304005, 2003. [Online]. Available: http://arxiv.org/abs/cs/0304005

[37] G. Kuperberg, "A subexponential-time quantum algorithm for the dihedral hidden subgroup problem," *SIAM J. Comput.*, vol. 35, no. 1, p. 170–188, Jul. 2005. [Online]. Available: https://doi.org/10.1137/S0097539703436345

[38] O. Regev, "A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space," 2004. [Online]. Available: https://arxiv.org/abs/quant-ph/0406151

[39] N. Howgrave-Graham, J. H. Silverman, A. Singer, and W. Whyte, "Naep: Provable security in the presence of decryption failures," Cryptology ePrint Archive, Report 2003/172, 2003. [Online]. Available: https://eprint.iacr.org/2003/172

[40] T. Pornin and T. Prest, "More efficient algorithms for the ntru key generation using the field norm," in *Public-Key Cryptography – PKC 2019*, D. Lin and K. Sako, Eds. Cham: Springer International Publishing, 2019, pp. 504–533. [Online]. Available: https://eprint.iacr.org/2019/015.pdf

[41] J. H. Silverman, *Almost Inverses and Fast NTRU Key Creation*, 1999. [Online]. Available: https://www.onboardsecurity.com/products/ntru-crypto/ntru-resources

[42] J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, W. Whyte, and Z. Zhang, "Choosing parameters for NTRUEncrypt," in *Topics in Cryptology – CT-RSA 2017*, H. Handschuh, Ed. Cham: Springer International Publishing, 2017, pp. 3–18. [Online]. Available: https://eprint.iacr.org/2015/708.pdf

[43] S. An, S. Kim, S. Jin, H. Kim, and H. Kim, "Single trace side channel analysis on NTRU implementation," *Applied Sciences*, vol. 8, no. 11, p. 2014, Oct 2018. [Online]. Available: http://dx.doi.org/10.3390/app8112014

[44] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *International Conference on Computers, Systems, & Signal Processing*, vol. 1, pp. 175–179, 1984. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2014.05.025

[45] D. J. Bernstein, *Post-Quantum Cryptography*, D. J. Bernstein, J. Buchmann, and E. Dahmen, Eds. Springer Berlin Heidelberg, 2009. [Online]. Available: https://doi.org/10.1007/978-3-540-88702-7_1

A Appendix

A.1 d Experiment Results for 100% Decryption Success Rates

$p = 3$, $q = 128$ for all tests. The message used for encryption is one of polynomial degree 70. The operation times are the average over 25 rounds. Tests that resulted only in decryption failures are marked in red, those that resulted only in successes are marked in green, and those that resulted primarily in failures throughout the full spectrum of possible d values are marked in yellow. These tests were not repeated for 50% and 0% success rates as they would produce the same results.

N	d_f	d_g	d_r	Key Generation Time (sec)	Encryption Time (sec)	Decryption Time (sec)
167	83	83	16	5.394	0.265	0.267
167	83	42	36	5.360	0.268	0.261
167	83	1	83	5.299	0.275	0.265
167	42	83	18	5.620	0.266	0.261
167	42	42	36	5.682	0.273	0.263
167	42	1	83	5.598	0.275	0.262
167	1	83	17	0.475	0.237	0.046
167	1	42	38	0.401	0.240	0.043
167	1	1	83	0.322	0.126	0.042
167	83	16	83	5.358	0.274	0.263
167	83	32	42	5.323	0.267	0.258
167	83	83	1	5.378	0.202	0.263
167	42	15	83	5.629	0.272	0.258
167	42	33	42	5.708	0.274	0.266
167	42	83	1	5.687	0.170	0.264
167	1	18	83	0.369	0.240	0.045
167	1	38	42	0.411	0.243	0.045
167	1	83	1	0.470	0.138	0.043
167	1	83	83	0.489	0.261	0.054
167	1	83	42	0.474	0.251	0.045
167	83	83	1	5.412	0.196	0.261
167	1	42	83	0.383	0.249	0.042
167	4	42	42	4.994	0.270	0.260
167	83	42	1	5.417	0.172	0.262
167	83	1	1	5.270	0.177	0.263

167	83	1	42	5.320	0.274	0.264
167	83	1	1	5.306	0.166	0.264
251	125	125	16	12.242	0.597	0.59
251	125	63	28	12.222	0.606	0.592
251	125	1	125	12.027	0.615	0.587
251	63	125	16	12.967	0.594	0.593
251	63	63	34	12.925	0.608	0.58
251	63	1	125	12.686	0.608	0.586
251	1	125	17	0.806	0.525	0.062
251	1	63	33	0.728	0.524	0.067
251	1	1	125	0.603	0.306	0.069
251	125	13	125	12.107	0.608	0.585
251	125	31	63	12.172	0.606	0.581
251	125	125	1	12.164	0.354	0.581
251	63	15	125	12.779	0.604	0.583
251	63	31	63	12.901	0.606	0.584
251	63	125	1	12.890	0.408	0.582
251	1	16	125	0.690	0.533	0.074
251	1	35	63	0.592	0.526	0.057
251	1	125	1	0.784	0.296	0.059
251	1	125	125	0.924	0.582	0.105
251	1	125	63	0.725	0.565	0.064
251	125	125	1	12.182	0.420	0.588
251	1	63	125	0.733	0.563	0.078
251	1	63	63	0.794	0.546	0.074
251	125	63	1	12.171	0.410	0.581
251	125	1	125	11.965	0.608	0.582
251	125	1	63	11.973	0.605	0.582
251	125	1	1	11.842	0.433	0.578
347	173	173	14	23.482	1.104	1.110
347	173	87	29	23.341	1.127	1.102
347	173	1	173	23.056	1.153	1.113
347	87	173	15	25.170	1.103	1.110
347	87	87	31	25.125	1.133	1.112
347	87	1	173	24.770	1.150	1.107
347	1	173	16	1.683	0.987	0.098
347	1	87	35	1.235	0.994	0.098
347	1	1	173	0.965	0.529	0.094
347	173	15	173	23.454	1.156	1.113
347	173	32	87	23.306	1.146	1.105
347	173	173	1	23.299	0.704	1.094
347	87	13	173	24.831	1.140	1.101
347	87	32	87	24.914	1.140	1.103
347	87	173	1	24.932	0.604	1.099

347	1	14	173	0.963	0.980	0.088
347	1	35	87	1.127	0.988	0.099
347	1	173	1	1.480	0.507	0.092
347	1	173	173	1.278	1.100	0.165
347	1	173	87	1.404	1.073	0.138
347	173	173	1	23.308	0.724	1.095
347	1	87	173	0.957	1.070	0.103
347	1	87	87	1.103	1.052	0.100
347	173	87	1	23.231	0.716	1.096
347	173	1	173	23.515	1.176	1.132
347	173	1	87	22.859	1.137	1.099
347	173	1	1	22.747	0.705	1.105

A.2 d Experiment Results for 50% Decryption Success Rates

N	d_f	d_g	d_r	Key Generation Time (sec)	Encryption Time (sec)	Decryption Time (sec)
167	83	83	25	5.354	0.268	0.260
167	83	42	62	5.384	0.274	0.263
167	42	83	31	5.820	0.276	0.269
167	42	42	62	5.652	0.272	0.262
167	1	83	28	0.515	0.253	0.049
167	1	42	64	0.415	0.249	0.047
167	83	29	83	5.482	0.279	0.268
167	83	58	42	5.402	0.270	0.263
167	42	30	83	5.767	0.277	0.262
167	42	56	42	5.670	0.270	0.264
167	1	32	83	0.367	0.251	0.040
167	1	61	42	0.459	0.246	0.048
251	125	125	26	11.960	0.589	0.574
251	125	63	48	12.111	0.594	0.578
251	63	125	27	12.764	0.587	0.575
251	63	63	48	12.785	0.605	0.579
251	1	125	28	1.005	0.535	0.074
251	1	63	54	0.766	0.526	0.065
251	125	24	125	12.070	0.607	0.582
251	125	49	63	12.038	0.603	0.579
251	63	25	125	12.736	0.598	0.579
251	63	44	63	12.874	0.608	0.581
251	1	26	125	0.695	0.542	0.073

251	1	54	63	0.737	0.544	0.071
347	173	173	24	23.261	1.109	1.096
347	173	87	46	23.435	1.149	1.106
347	87	173	24	25.200	1.153	1.103
347	87	87	46	25.309	1.147	1.107
347	1	173	24	1.668	1.092	0.105
347	1	87	46	1.130	1.039	0.087
347	173	25	173	23.242	1.146	1.104
347	173	47	87	23.512	1.156	1.115
347	87	24	173	25.190	1.161	1.114
347	87	45	87	25.279	1.157	1.113
347	1	26	173	1.102	1.064	0.101
347	1	52	87	1.197	1.034	0.104

A.3 d Experiment Results for 0% Decryption Success Rates

N	d_f	d_g	d_r	Key Generation Time (sec)	Encryption Time (sec)	Decryption Time (sec)
167	83	83	42	5.368	0.271	0.262
167	83	42	80	5.373	0.273	0.260
167	42	83	45	5.809	0.279	0.268
167	42	42	78	5.635	0.276	0.265
167	1	83	44	0.494	0.259	0.049
167	1	42	79	0.362	0.250	0.038
167	83	46	83	5.477	0.279	0.262
167	83	81	42	5.359	0.269	0.259
167	42	44	83	5.754	0.279	0.268
167	42	75	42	5.666	0.270	0.261
167	1	55	83	0.434	0.258	0.049
167	1	78	42	0.407	0.251	0.040
251	125	125	41	12.069	0.595	0.579
251	125	63	65	12.077	0.603	0.581
251	63	125	40	12.776	0.595	0.573
251	63	63	68	12.860	0.604	0.584
251	1	125	40	0.742	0.557	0.063
251	1	63	68	0.713	0.554	0.066
251	125	42	125	12.156	0.610	0.584
251	125	70	63	12.060	0.599	0.583
251	63	44	125	12.880	0.609	0.585
251	63	76	63	12.866	0.602	0.582
251	1	47	125	0.673	0.559	0.071
251	1	69	63	0.746	0.551	0.071

251	79	63	63	12.862	0.598	0.579
347	173	173	37	23.139	1.123	1.093
347	173	87	63	23.449	1.136	1.113
347	87	173	33	25.010	1.131	1.100
347	87	87	68	25.152	1.142	1.104
347	1	173	36	1.587	1.051	0.107
347	1	87	67	1.128	1.034	0.099
347	173	34	173	23.197	1.147	1.107
347	173	64	87	23.294	1.144	1.102
347	87	34	173	25.404	1.162	1.120
347	87	63	87	25.506	1.159	1.118
347	1	35	173	1.063	1.057	0.098
347	1	59	87	1.145	1.040	0.092