Lattices, Cryptography, and NTRU

An introduction to lattice theory and the NTRU cryptosystem

Ahsan Z. Zahid

A thesis presented for the degree of
Bachelor of Science

School of Science
St. Mary's College of California
Moraga, CA
May 21, 2017

1 Background

NTRU (pronounced either as an initialism or as “en-true) is a cryptosystem that utilises lattice theory. The most well-known modern-day encryption algorithm, called (Rivest-Shamir-Adleman) generally relies on the difficulty of finding the prime factorisations of large numbers. NTRU encryption relies instead on the “closest-vector problem. The acronym stands for “Number Theorists R Us.

Before studying the mathematics behind NTRU, we will examine the motivation for creating and using such a system.

1.1 The Need for Encryption

A great deal of human progress is a direct result from being able to communicate. Communication becomes difficult when receiving parties must avoid any third party intercepting the messages sent back and forth, especially when the two parties are communicating over long distances.

In the modern day, we send messages from our computers every day that may contain sensitive personal information. These messages may travel thousands of miles before reaching the recipient, and all along the way there is no guarantee of privacy. There are, of course, criminals that would love to harvest people's personal information by intercepting and reading these messages, for the sake of committing crimes. Many modern governments also have been found to be conducting surveillance on the messages travelling through their borders. Whether from criminals or from governments (foreign and domestic), protecting the privacy of information sent through the Internet becomes more important by the day.

This is why cryptography, the study of secure communications, has become an important field in the modern day. In order to secure messages, people have used techniques that obscure the message in a way that is reversible, yet

secure. A cryptosystem is a suite of algorithms that are implemented to secure communications between the sending and receiving parties.

1.2 The Situation for Cryptography

One side will "encrypt the message, and send the encrypted message. Unless an intercepting party has the "key necessary to "decrypt the message, the encrypted message is hopefully unreadable to third parties. When the intended receiver finally receives the message she can use her key to decrypt the message and read it.

There are two kinds of cryptography: symmetric and asymmetric systems. In a symmetric system, both sides use the same key, and use this one key to both encipher and decipher messages. If a third party manages to obtain this key, that third party can decrypt all the messages. Hence, the key must remain secret, or "private". There are several problems with symmetric cryptography: the parties must find some other way to exchange keys, a new key is required for each person one wishes to share a message with, and interception of a single key can ruin the security of all communication at once.

A symmetric encryption system is constantly at the mercy of the security practices of all parties holding the key. Negligence on the part of any of the key-holders can sabotage the security of messages being sent over the system. If, for example, a key-holder loses his computer or does not secure his computer against unauthorised use, a third party can attempt to retrieve the key and decipher any messages enciphered with the key, possibly without any other key-holders being aware their communication is being intercepted.

In an asymmetric, or "public key cryptosystem, there are two keys: a public key and a private key. The public key is widely accessible, and anyone who wishes to do so may use the public key to encrypt information to be sent using

the system. However, to decrypt this information one needs the corresponding private key, a separate key that only the receiving party has access to.

If two parties wish to communicate in an asymmetric cryptosystem, they must exchange public keys, and keep their private keys secret, even from each other. In the RSA cryptosystem, generating these keys and using them involves finding the products of large primes. Breaking such a cryptosystem involves factoring these large products back into their prime factors, which proves to be a difficult task. NTRU uses several parameters to generate public and private keys, as we will see later.

1.3 A Fundamental Example: Caesar Ciphers

To better understand private key ciphers, we'll briefly examine a very simple system: Caesar ciphers (also known as shift ciphers). The following cryptosystem is a symmetric one (that is, both sides need to know the same information).

A Caesar cipher encrypts text by shifting each character in the "plaintext information by a predefined number of characters in the alphabet. This generates "ciphertext, which is the encrypted information sent to the opposite party.

Let us take for example the following to be the plaintext:

Blood alone moves the wheels of history.

Let us decide also that we will shift all characters backwards by two characters. This means that any 'b' will become a 'z', and 'd' will become 'b', and any 'm' will become 'k'. This is, by the way, the same as shifting all characters forward b 24 characters.

Shifting the letters thus yields the following ciphertext:

Zjmmb yjmlc kmtcq rfc ufccjq md fgqrmpw.

Without knowledge of how to properly shift the ciphertext back into the plaintext, obtaining the information mid-transmission will not be of any use to an intercepting party. [3]

Caesar ciphers are very easily broken; a "brute-force" attack would require testing only 25 possible keys for a plaintext message written with English characters.

In this system there is one key meant to be kept private. Both parties must have this key for the system to work. In cases where the two parties are far away, or where all lines of communication are unsecured, securely sharing the key becomes a problem. One might take a chance with writing letters or making phone calls to transfer keys, but these media are prone to surveillance. Bringing the two parties together physically may be difficult if they are far away.

2 Lattices

2.1 Lattices from Vectors

A lattice is generated using a set of vectors called basis vectors. Lattice generation is similar to taking the span of a set of vectors, but using only integer coefficients. Formally:

Definition 1. *Lattice*

Consider n vectors $b_1, b_2, \ldots, b_n \in \mathbb{R}^m$ which are linearly independent. We call these the basis vectors of our lattice. We refer to m as the dimension of the lattice, and n the rank of the lattice. A lattice is said to be full-rank when $n = m$.

We define the lattice generated by these vectors thus:

$$\mathcal{L}(b_1, \ldots b_n) = \left\{ \sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z} \right\}.$$

Should we write the bases as columns of a matrix B, we may also use the following notation:

$$\mathcal{L}(B) = \{Bx \mid x \in \mathbb{Z}^n\}.$$

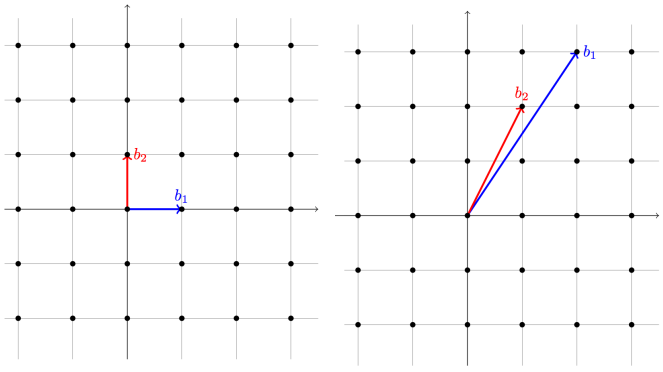
In the case of full-rank lattices, B will be a square matrix. [8]

The remainder of this paper will focus on full-rank lattices.

An important fact is that the bases of a lattice are not unique. Each lattice has many different bases, all of which generate the same lattice.

For example, the lattice generated by the bases $(0, 1)$ and $(1, 0)$ is the same as the lattice generated by the bases $(1, 2)$ and $(2, 3)$. This two-dimensional lattice consists of all integer coordinates in $\mathbb{R}^2$.

Since points in the lattice are created by taking integer linear combinations of the bases together, having different bases means that reaching the same lattice point will sometimes involve very different combinations of vectors for different bases. Getting to the same point might be easier with some bases (shorter, orthogonal ones) and much more difficult with other (long, less orthogonal) bases. This becomes crucial for cryptography, as we will see.



In the above figure, the bases on the left are perfectly orthogonal, and are

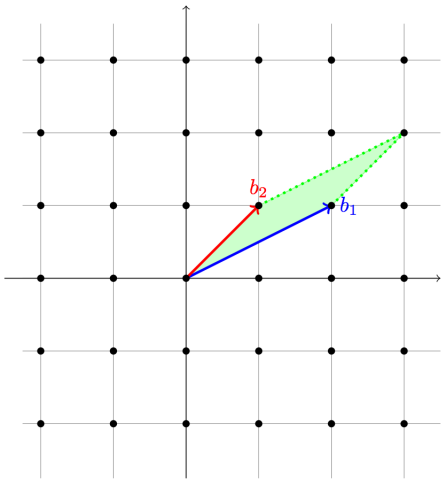
relatively short. The bases on the right, in contrast, are long and less orthogonal. Both of these bases, however, generate the same lattice.

Once we have a set of basis vectors for a lattice, we can construct a fundamental parallelepiped:

Definition 2. *Closed Fundamental Parallelepiped*

Let the n vectors $b_1, b_2, \ldots, b_n \in \mathbb{R}^m$ be linearly independent. The closed fundamental parallelepiped of these vectors is defined to be

$$\mathcal{P}(b_1, \ldots b_n) = \left\{ \sum_{i=1}^{n} x_i b_i \in \mathbb{R}, 0 \leq x_i \leq 1 \right\}.$$



2.2 Checking for Equivalent Lattices by Bases

A natural question is whether, given two matrices of bases, one can determine whether the lattices generated by each one will be the same.

To understand the relationship between lattices and basis vectors, we must examine unimodular matrices.

Definition 3. *A matrix $U \in \mathbb{Z}^{n \times n}$ is called unimodular if $|\det(U)| = 1$.*

So, a square matrix whose determinant's absolute value is 1 is a unimodular matrix. Since the definition is based on determinants we can make the following proposition:

Proposition 1. *If an $n \times n$ matrix U is unimodular, then U^{-1} is unimodular as well.*

Proof. Simply examine the relations of determinants to inverses of matrices: $\det(U^{-1}) = 1/det(U) = \pm 1$, and thus it will always be true that $|\det(U^{-1})| = 1$ □

And now we can determine whether bases are equivalent:

Theorem 2.1. *Consider two full-rank, $n \times n$ matrices of basis vectors $B, B' \in \mathbb{R}^{n \times n}$. $\mathcal{L}(B) = \mathcal{L}(B')$ if and only if there exists a unimodular matrix U for which $B' = BU$.*

Proof. Going forward: let $\mathcal{L}(B) = \mathcal{L}(B')$. This means that all integer combinations of the column vectors of B must equal the linear combinations of the column vectors of B'. Then it must be true that there exist some integer matrices V and V' for which $B' = BV$ and $B = B'V'$.

We put the equations together like so:

$$B' = BV = B'(V'V),$$

and multiply both sides by B'^{-1} :

$$V'V = I_n.$$

I_n, here, represents the identity vector in n dimensions.

Finally, since determinants are multiplicative, and all integers involved are integer matrices, we know that $\det(V')det(V) = 1$. And thus we have shown

that:

$$\det(V) = \det(V') = \pm 1.$$

Next, going backwards: Let U be a unimodular matrix such that $B' = BU$. Note that U is an integer matrix. That means that all the column vectors of B' can be written as linear combinations of the vectors in B. Thus,

$$\mathcal{L}(B') \subseteq \mathcal{L}(B).$$

We repeat this logic with the statement that $B = B'U^{-1}$, which leads us to the statement

$$\mathcal{L}(B) \subseteq \mathcal{L}(B').$$

Putting these two together, we see that $\mathcal{L}(B) = \mathcal{L}(B')$. □

2.3 Equivalent Lattice Definitions

In addition to the above definition of lattices through vectors, there exists an equivalent definition involving additive subgroups.

Definition 4. *Subgroup Definition*

$\mathcal{L}$ is a lattice if it is a discrete additive subgroup of $\mathbb{R}^n$.

Discrete, here, means:

$$\exists \epsilon \in \mathbb{R}, \epsilon > 0 \mid \forall x \neq y \in \mathcal{L}, ||x - y|| \geq \epsilon.$$

This means that there is some positive, real ϵ such that no two elements in $\mathcal{L}$ can be less than ϵ apart. Effectively, this creates a minimum distance between lattice elements.

While additive means:

$$\forall x, y \in \mathcal{L}, x - y \in L.$$

2.4 Finding Short Vectors

We are particularly interested, for the purposes of cryptography, in finding the shortest non-zero vectors in a lattice. Formally, we refer to successively shortest vectors in our lattice (that is, the ith shortest vector) as $\lambda_i(\mathcal{L})$. Thus, the shortest possible vector length in lattice $\mathcal{L}$ is $\lambda_1(\mathcal{L})$, often abbreviated to λ_1 if there is no confusion about the lattice in which it exists. More formally:

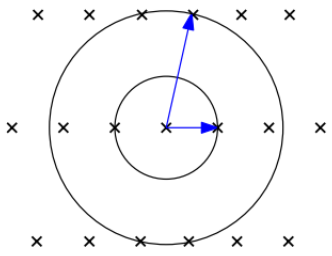
Definition 5. *Successive Minima [2]. For lattice $\mathcal{L}$, the ith shortest vector is defined as:*

$$\lambda(\mathcal{L}) = \inf\{r \mid dim(span(\mathcal{L} \cap B(0, r))) \geq i\}$$

for $B(0, r) = \{x \in \mathbb{R}^m \mid ||x|| \leq r\}$, or a closed ball of radius r around 0.

Essentially, $\lambda(\mathcal{L})$ will be the smallest possible value in a set of all r where a closed ball of radius r at the origin contains at least i linearly independent lattice points.

Here, λ_1 and λ_2 are illustrated for a lattice:



Note that λ_1 contains at least one point in the ball formed by its radius

around a lattice point, and λ_2 contains at least two linearly lattice points.

Next, we will examine an important theorem that provides an upper bound on $\lambda_1(\mathcal{L})$. Before we can do so, we need the following preliminary theorems and definitions.

Theorem 2.2. *Blichfeldt's Theorem [2]*

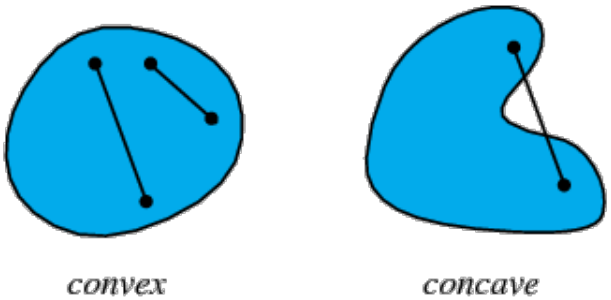
If we have a full rank lattice $\mathcal{L}$ and measurable set $S \subseteq \mathbb{R}^n$ such that $vol(S) > \det(\mathcal{L})$, $\exists x, y \in S \colon x - y \in \mathcal{L}$.

Essentially, Blichfeldt's Theorem tells us that for a set S with volume greater than $\det(\mathcal{L})$, (or the fundamental parallelepiped of the basis vectors of $\mathcal{L}$) the set can be translated in $\mathcal{R}^n$ such that there are two vectors x and y in S whose difference is a vector in $\mathcal{L}$. If the volume of a measurable set has a volume large enough, it can be positioned in a way that guarantees lattice points will exist within it.

Definition 6. *Convex Set. A set S is called convex if:*

$$\forall x \neq y \in S, \forall \alpha \in [0,1], \alpha x + (1-\alpha)y \in S.$$

In other words, for a convex set, should we choose two distinct points from the set, all points on the line connecting these points must also be in the set.



In the above figure, any two points in the convex set can be connected by

a line within the set. In the concave set however, a line between two points in the set contains points outside the set.

Definition 7. *Centrally Symmetric Set. A set S is centrally symmetric if:*

$$\forall x \in S, -x \in S.$$

Simply, a centrally symmetric set is one where every element contains its additive inverse.

Theorem 2.3. *Minowski's Convex Body Theorem*

For any full-rank lattice $\mathcal{L}$ and a convex, centrally symmetric set S for which $vol(S) > 2^n \det(\mathcal{L})$, S contains a non-zero lattice point.

This theorem states that if a convex, centrally symmetric body has a volume large enough, it must contain a lattice point.

Now that we have several useful pieces of information in place, Minowski's First Theorem becomes clear.

Theorem 2.4. *Minowski's First Theorem [7]*

For any full-rank lattice, $\mathcal{L}$, we have

$$\lambda_1(\mathcal{L}) \leq \sqrt{n} \times \det(\mathcal{L})^{1/n}.$$

Proof. Minowski's First Theorem [7]

Let $\mathcal{B}(x, r)$ be the n-dimensional open ball of radius r, with center at x.

Then, the n-dimensional cube of side length $\frac{2r}{\sqrt{n}}$ will be inscribed within this ball. Since the ball must contain at least this, we have

$$vol(\mathcal{B}(0, r)) \geq (\frac{2r}{\sqrt{n}})^n.$$

So, if we let $r = \lambda_1(\mathcal{L})$), we get

$$vol(\mathcal{B}(0, \lambda_1(\mathcal{L})) \geq (\frac{2\lambda_1(\mathcal{L})}{\sqrt{n}})^n.$$

Given Minowski's Convex Body Theorem, and the fact that since the the ball is open, it cannot contain any non-zero lattice points we gain an upper bound to this equation:

$$(\frac{2\lambda_1(\mathcal{L})}{\sqrt{n}})^n \leq vol(\mathcal{B}(0, \lambda_1(\mathcal{L})) \leq 2^n \det(\mathcal{L}).$$

All of this is rearranged as $\lambda_1(\mathcal{L}) \leq \sqrt{n} \times \det(\mathcal{L})^{1/n}$

□

We have now found an upper limit to the length of the shortest vector in any given lattice.

Though Minowski's First Theorem provides bounds for the shortest vector, it does not help produce any sort of process to actually find such a vector. In fact, as far as we know, there does not exist an algorithm to efficiently find the shortest vector in a given lattice. For this reason, finding the shortest vector in a lattice is an excellent problem for cryptography. [4]

In summary, it is easy to construct a lattice using short vectors, but difficult to take a pre-existing lattice and find the shortest vector.

3 Lattice Problems

3.1 Shortest Vector Problem

The shortest vector problem is the problem of determining the shortest nonzero vector in lattice $\mathcal{L}$. There are a few variations to the problem, but the simplest version is as follows:

Definition 8. *Shortest Vector Problem. Let $B \in \mathbb{Z}^{m \times n}$ be the basis matrix for a lattice $\mathcal{L}$. Find a vector $v \in \mathcal{L}(B)$ such that $||v|| = \lambda_1(\mathcal{L}(B))$.*

A variation of this problem used in cryptography includes an approximation factor, γ. The problem is to find a nonzero vector whose length is within γ times the shortest possible vector. Thus, you need not find precisely the shortest vector but rather one that is short within the shortest possible scaled by γ.

Definition 9. *SVP_γ. Let $B \in \mathbb{Z}^{m \times n}$ be the basis matrix for a lattice $\mathcal{L}$. For some $\gamma \geq 1$ find $v \in \mathcal{L}(B)$ such that $v \neq 0$ and $||v|| \leq \gamma \times \lambda_1(\mathcal{L}(B))$.*

For lattice cryptography problems, we can also take γ to be a polynomial in the dimension of the lattice, rather than just scaling the shortest or closest vector. Solving problems like these require at least exponential time as the dimensions increase, making them quite difficult to solve in high dimensions. [3]

3.2 Closest Vector Problem

The closest vector problem is the problem of finding the vector in $\mathcal{L}$ closest to a given vector w not in the lattice. Formally:

Given k linearly independent vectors $B = [b_1, \ldots, b_k]$ in n-dimensional space and a vector w called the "target vector, the Closest Vector Problem is to find an integer linear combination of the vectors in B, Bx, with a minimal distance from the target. That is, we wish to minimise $||Bx - y||$ over all $x \in \mathbb{Z}^k$.

Styling this problem like SVP, we can define it formally:

Definition 10. *CVP_γ. Given a basis matrix $B \in \mathbb{Z}^{m \times n}$, an approximation factor $\gamma \geq 1$, and a target vector $t \in \mathbb{Z}^m$, find a vector $v \in \mathcal{L}(B)$ so that $||v - t|| \leq \gamma \times dist(t, \mathcal{L}(B))$.*

3.3 Application

The task of finding shortest and closest vectors is not too difficult in lower dimensions. In higher dimensions, this task becomes very difficult. Though there are some algorithms for this purpose, there are none that can operate faster than $O(1)^n$. Thus, creating lattices is easy, but finding shortest and closest vectors without very good bases seems very difficult. This makes these two problems a good foundation for cryptographic algorithms. If a problem can be formulated to be a lattice problem, the same difficulty applies. NTRUEncrypt, in fact, poses algebraic problems with polynomial rings whose solutions can be thought of as vectors in a lattice.

3.4 LLL

A significant character in the history of lattice based cryptography is an algorithm known as Lenstra–Lenstra–Lovász, or the LLL algorithm, named for its creators. The LLL algorithm is used to reduce a lattice to its orthogonal basis vectors, given a set of "bad bases. This algorithm consists of two parts:

- Start with a non-basis working vector and subtract integer multiples of the current basis vectors
- Decide if the working vector should become the next basis vector, or whether it should replace the previous basis vector.

Whether or not to replace the previous basis vector is decided based on the Lovász Condition, or whether the working vector is big enough to be the next basis.

LLL can be used to attack RSA encryption, and initially deterred work on lattice based cryptography.[1] NTRU is resistant against the LLL algorithm. [4]

4 Good and Bad Bases

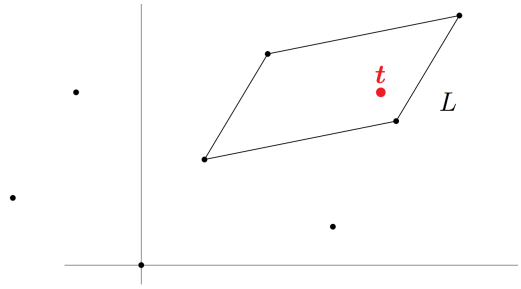
Having seen that any lattice has an infinite number of possible bases, we can ascertain a difference between these bases with regard to the Closest Vector in particular.

4.1 Good Bases

If the basis of a lattice consists of relatively short, relatively orthogonal bases, the fundamental parallelepiped will be more compact. As a result, there is a good chance that, given a target vector for a CVP problem, the parallelepiped will contain the vector closest to the target.

The fundamental parallelepiped is translatable by adding a lattice vector to it. That is, adding a lattice vector to each point in the parallellepiped will result in another parallellepiped in the lattice. This way, any vector in the lattice lies inside some translation of the fundamental parallelepiped. If the parallelipied is compact, or "smaller" (that is, the bases are relatively short and relatively orthogonal), an arbitrary vector t in $\mathbb{R}^n$ is likely to be close to one or more of the corners of a parallelepiped, which are simply lattice points. So, smaller parallelepipeds will consist of vectors in the lattice closer to t.

For example, in the following figure, the vector closest to the target is actually one of the vectors that makes up the parallelepiped.

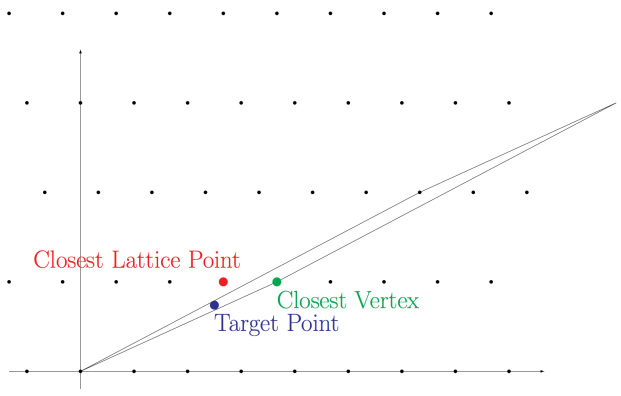


When this is the case and we have such "good bases, there is a very simple way to find the closest lattice vector. The target vector can be expressed as a linear combination of the basis vectors, albeit with real coefficients (rather than integer coefficients, like the lattice points). If the parallelepiped around the target contains the closest vector, simple round the coefficients of t to integers to arrive at a lattice point.

For example, suppose the full-rank lattice $\mathcal{L}$ is generated by bases b_1 and b_2. Suppose our target vector t can be expressed as $1.2b_1 + b_2$. If the bases are sufficiently short, we can round the coefficients of t to find vector $v = b_1 + b_2$ in the lattice, the lattice vector closest to the target. The same holds for lattices in higher dimensions: for a vector $t_2 = 1.65b_1 + 4.5b_2 + 0.68b_3$, we simply round to find $v_2 = 2b_1 + 5b_2 + 1b_3$.

4.2 Bad Bases

Conversely, where the basis vectors for the same lattice are longer or not generally orthogonal, an arbitrary vector may not be very close to any of the vectors of the parallelepiped drawn around it by translating the fundamental parallelepiped, so the rounding method is not useful. We consider these to be "bad bases.



An important consequence of this, combined with Theorem 2.1 above is that good bases can be transformed into bad ones. Let B be a good basis for a lattice. Should we multiply B by a unimodular integer matrix U, we know that $B' = BU$ will also be a basis for the same lattice. Thus, if B is a good basis for the lattice, we can transform it into a bad basis, B', by multiplying it by a unimodular integer matrix.

4.3 Lattice Cryptography

Both bases generate the same lattice. An interesting application of this phenomenon is to think of the good basis vectors as a private key, which makes solving CVP simple for a given target vector. The bad basis vectors provide a public key, which does not aid in solving CVP, but does provide a way to generate the lattice and place a target vector in the lattice. This basic idea has been used to create a number of asymmetric lattice-based cryptosystems. This is also, in a broad sense, how NTRUEncrypt works. [6]

The message is encoded as a vector m. A random point in the lattice is chosen. Encryption means adding this short vector to this point on the lattice to form a vector e. Thus the "plaintext vector m has turned into the "ciphertext" vector e. Then, multiply the lattice's good basis B by a unimodular matrix U to get a bad basis.

To decrypt the message, find the lattice point closest to the ciphertext vector eU^{-1}, and subtract it from the ciphertext vector. This results in the original, plaintext vector.

It is important to note that encryption involves choosing a random point in the lattice to which the plaintext is attached. This means that encryption in NTRU is probabilistic, and the encrypted forms of messages are not unique. The same message, encrypted multiple times in NTRU will yield different encrypted

forms.

5 NTRU

5.1 A Brief History

NTRU is a cryptosystem made up of two algorithms: NTRUEncrypt, and NTRUSign, used for encryption and digital signatures, respectively. It was originally developed in 1996 by three mathematicians: Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman. Eventually a company, NTRU Cryptosystems, Inc. was formed around the algorithms, and the team obtained a patent on the cryptosystem. The system continues to evolve as people release new versions and variations intended to make it more secure against attacks. Security Innovation, Inc purchased NTRU Cryptosystems, Inc in 2009 and now licenses NTRU for commercial use. NTRU remains open source and free to use for those who wish to use it non-commercially. One of NTRU's main claims to fame over systems like RSA is that NTRU is not known to be vulnerable against attacks mounted by quantum computers, and is thus being studied for its application in a post-quantum-computer world. [6]

5.2 Necessary Background

The following definitions make concrete certain algebraic structures used for encryption and decryption with NTRUEncrypt.

Definition 11. *A ring R is a set S paired with two binary operations, $+$ and $*$ which satisfy the following properties:*

Property 1. Additive associativity: $\forall a, b, c \in S, (a + b) + c = a + (b + c)$

Property 2. Additive commutativity: $\forall a, b \in S, a + b = b + a$

Property 3. Additive identity: There is an element called 0 such that $\forall a \in S, 0 + a = a + 0 = a$

Property 4. Additive inverse: $\forall a \in S, \exists -a \mid a + (-a) = -a + a = 0$

Property 5. Distributivity: $\forall a, b, c \in S, a * (b + c) = a * b + b * c$, *and* $\forall a, b, c \in S, (b + c) * a = b * a + c * a$

Property 6. Multiplicative associativity: $\forall a, b, c \in S, a * (b * c) = (a * b) * c$

Definition 12. *Polynomial Ring. For a given ring* R, $R[x]$ *is the ring of polynomials in* x *with coefficients in* R.

$$R[x] = \left\{ \sum_{i=0}^{\infty} a_i x^i \mid a_i \in R \right\}.$$

Definition 13. *Ideal. Given a ring* R, *an ideal* I *of* R *is a subset that forms an additive group such that for any* $x \in R, y \in I$, *we have* $xy, yx \in I$.

We see an example with the ring of integers, $\mathbb{Z}$, and the subset of even integers, which forms an ideal of $\mathbb{Z}$.

Definition 14. *Quotient Ring. Given a ring* R *and an ideal* I *of that ring, the quotient* R/I *forms another ring,* Q, *called a quotient ring.*

Alternatively, a quotient ring R/I *is a set of equivalence classes such that for* $x, y \in Q, x = y$ *if* $x - y \in I$.

5.3 Necessary Structures

Though CVP and SVP are the problems underlying NTRUEncrypt, on the surface NTRUEncrypt involves convolution polynomial rings over the integers. This makes it so that encryption and decryption can be represented as numerical operations (which computers are adept at performing) rather than needing to work with vectors directly.

NTRUEncrypt involves certain parameters that must be chosen for the algorithm to work.

Initially, we choose N, p, and q such that N is prime and p is not divisible by q.

First, we define three polynomial quotient rings:

$$R = \frac{\mathbb{Z}[x]}{x^N - 1}, R_p = \frac{\mathbb{Z}_p[x]}{x^N - 1}, R_q = \frac{\mathbb{Z}_q[x]}{x^N - 1},$$

where $\mathbb{Z}_p = \mathbb{Z}/p$ and $\mathbb{Z}_q = \mathbb{Z}/q$. Multiplication in the rings will be referred to as *.

These are quotient rings formed by rings of polynomials whose coefficients are in $\mathbb{Z}$, $\mathbb{Z}_p$, and $\mathbb{Z}_q$, respectively, whose quotients are taken with a divisor of $x^N - 1$. This means we can reduce polynomials modulo $x^N - 1$.

Addition and multiplication of polynomials in a ring, say, R_p satisfy the following:

$$(a(x) + b(x)) \pmod p = (a(x) \pmod p) + (b(x) \pmod p)$$

$$(a(x) * b(x)) \pmod p = (a(x) \pmod p) * (b(x) \pmod p)$$

An element a(x) in the ring R_p contains a multiplicative inverse if the following holds true in $Z_p[x]$:

$$gcd(a(x), x^N - 1) = 1.$$

5.4 Parameter Selection

We mentioned earlier that N must be a prime number. The two moduli p and q should be relatively prime. These parameters are used to generate a set of polynomials used later on in the encryption process.

It is common to use parameters like $p = 2$ or 3, and $q = 2^i$ for $8 < i < 11$. Fix d_f, g_g and d_Φ as positive integers. We will use these to define a function to generate a set of polynomials from which a private key, will be selected.

$L(d_1, d_2) = \{f \in R \mid d_1$ coefficients of f are equal to 1, d_2 coefficients are equal to -1, and all other coefficients are $0\}$

We use this to generate $L_f = L(d_f + 1, d_f)$, $L_g = L(d_g, d_g)$, and finally $L_\Phi = L(d_\Phi, d_\Phi)$. Finally, we define $L_m = \{m \in R$ such that m has coefficients in the interval $(\frac{-p}{2}, \frac{p}{2}]\}$. It is from these sets that we draw the polynomials used to generate public and private keys. [5]

5.5 Key Generation

The sender of a message begins by generating keys. Start by choosing random polynomials $f \in L_f$ and $g \in L_g$, under the condition that f has inverses mod p and mod q.

Compute the inverses of f mod p and mod q, which we call F_p and F_q respectively. This means that $F_p * f = 1 \pmod{p}$ and $F_q * f = 1 \pmod{q}$.

Furthermore, find $h = F_q * g \pmod{q}$. Now, f is the private key while h is the public key. One who wishes to receive messages can publish h, which a sender can use to encrypt a message. The recipient can then decrypt using the private key f.

5.6 Encryption

To encrypt a message, first choose a message $m \in L_m$ whose coefficients are reduced modulo p. Next, choose a random polynomial $\Phi \in L_\Phi$. The ciphertext e is generated by computing:

$$e = p\Phi * h + m \pmod{q}.$$

5.7 Decryption

Given ciphertext e encrypted using the process above, decryption is computed by multiplying the ciphertext by the private key, then the inverse of f, modulo p, as follows:

$$a = f * e \pmod{q}$$

$$b = F_p * a \pmod{p}.$$

As we will show, b is in fact our original plaintext message m.

Proof. Given a ciphertext polynomial e, define, $a = f * e \pmod{q}$. Expanding the ciphertext shows that $a = f * (p\Phi * h + m) \pmod{q}$, which reduces to $a = pf * \Phi * h + f * m \pmod{q}$.

Through the parameter selection process, we are ensured that the coefficients of a cannot be any further reduced modulo q, so that $a = pf * \Phi * h + m$ exactly. We then compute $b = F_p * a \pmod{p}$.

Since we know what a is, we can substitute for it to get $b = F_p * (pf * \Phi * h + f * m)$.

Since a term in our equations has p as a coefficient, we reduce it to reach $b = F_p * f * m \pmod{p}$. Recall that f and F_p are inverses, modulo p, so ultimately,

$$b = m \pmod{p}.$$

And thus we can be sure that b is the original plaintext message m. □

You may notice that for decryption to work, the coefficients of $pf*\Phi*h+f*m$ are small, specifically in the interval $(\frac{-q}{2}, \frac{q}{2}]$. In cases where the coefficients lie outside this interval, decryption will fail.

Thus to avoid decryption failure, the parameters must be chosen such that the coefficients fall in this interval. [5]

5.8 NTRUEncrypt as a CVP

We will now briefly examine how the above process can be turned into a lattice problem.

For a given h (the public key), we can consider a Convolution Modular Lattice, $\mathcal{L}_h$:

$$L_h = \{(a, b) \in \mathbb{Z}^{2N} : a \times h \equiv b \pmod{q}\}.$$

Recall that the ciphertext is defined thus:

$$e = r * h + m.$$

We can use the ciphertext to construct the a vector:

$$(0, e) = (0, r * h + m \pmod{q}).$$

A bit of vector manipulation yields us the following

$$(0, e) = (r, r * h \pmod{q}) + (-r, m).$$

In this equation we see clearly that the ciphertext is a vector, $(-r, m)$ added to another vector, $(r, r * h) \pmod{q}$ which, in fact, exists within the lattice L_h.

This allows us to see decryption as a closest vector problem looking for the vector closest to $(0, e)$ such that we can subtract it from $0, e$ to arrive at $-r, m$, which gives us access to the original plaintext m. [4]

6 Conclusion

Lattices are discrete structures formed by taking integer linear combinations of a set of linearly independent basis vectors. A single lattice can be generated by an infinite number of sets of basis vectors. This allows us to propose the problems of finding the shortest vector in a lattice, or a vector closest to a target non-lattice point. Such problems are simple in lower dimensions, but increase in difficulty exponentially. This makes these problems fit for use in cryptography. Indeed, the NTRU cryptosystem harnesses the closest vector problem to encrypt and decrypt data. Though the algorithms represent lattices, on the surface NTRU processes elements in convolution polynomial rings to generate pairs of public and private keys for use in encryption and decryption. As time goes on, NTRU is improved, and aims to provide effective, efficient cryptography for a post-quantum-computer world.

References

[1] Alexander May. *Using LLL-Reduction for Solving RSA and Factorization Problems*, pages 315–348. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[2] Oded Regev. Lattices in computer science: Lecture 1, 2004.

[3] J.H. Silverman, Jill Pipher, and Jeffrey Hoffstein. *An Introduction to Mathematical Cryptography / by J.H. Silverman, Jill Pipher, Jeffrey Hoffstein.* Undergraduate Texts in Mathematics. New York, NY : Springer New York, 2008., 2008.

[4] Joseph Silverman. Ntru and lattice-based crypto: Past, present, and future, 2015.

[5] Sonika Singh and Sahadeo Padhye. Generalisations of ntru cryptosystem. *Security and Communication Networks*, (18):6315, 2016.

[6] Nigel P. Smart. *Cryptography Made Simple*, chapter 5. Springer International Publishing, 2016.

[7] Vinod Vaikuntanathan. Csc 2414: Lattices in computer science lecture 2, 2011.

[8] Vinod Vaikuntanathan. Lattices in computer science lecture 1, 2011.