A Mathematical Introduction to Lattice-Based Cryptography

Simon Campos Greenblatt*

Department of Mathematics
North Carolina State University
Raleigh, North Carolina

Spring 2021

1 Introduction

Modern cryptosystems such as RSA and Diffie-Hellman rely on the difficulty of factoring large numbers or solving discrete logarithms. However, with the advent of quantum computers, some of these difficult problems might become feasibly solvable in the future. As such, quantum-resistant cryptosystems are an important area of research in the field of information security. A promising theory that has emerged over the last several years is that of lattices. In addition to providing difficult problems for which no known quantum solutions exist, lattice-based cryptosystems lend themselves well to parallelized implementations for faster encryption and decryption. Along with the fact that these cryptosystems are relatively easy to implement, these properties put lattice-based cryptography in a forward-thinking position.

The purpose of this paper is to introduce the theory of lattices in the context of cryptography by discussing their properties, hard problems based on them, and an overview of both the GGH and NTRU cryptosystems. Concrete examples will be provided to illustrate the processes of key creation, encryption, and decryption. I will also discuss the LLL lattice reduction algorithm as a way of attacking these cryptosystems and its applications to cryptanalysis. Finally, I will touch upon digital signature schemes based on GGH and NTRU.

2 Definitions and Properties of Lattices

Definition: Let $\boldsymbol{v_1}, ..., \boldsymbol{v_n} \in \mathbb{Z}^n$ be a set of linearly independent vectors with integer entries. The *integral lattice* generated by $\boldsymbol{v_1}, ..., \boldsymbol{v_n}$ is the span with integer coefficients,

$$L = \{a_1\boldsymbol{v_1} + ... + a_n\boldsymbol{v_n} : a_1, ..., a_n \in \mathbb{Z}\} \tag{1}$$

Equivalently, an integral lattice can also be defined of as an additive subgroup of $\mathbb{Z}^m$ for some $m \geq 1$.
A *basis* for L is any set of linearly independent vectors that can generate L. The *dimension* of L is its number of basis vectors. It is often convenient to form a matrix using the basis vectors as the rows of the matrix.

Proposition 2.1: Any two bases for a lattice L are related by a matrix having integer coefficients and a determinant equal to ± 1.

This set of matrices is called the *general linear group* (over $\mathbb{Z}$) and is denoted by $GL_n(\mathbb{Z})$. It is also the group of matrices with integer entries whose inverses also have integer entries.

*Correspondence to: sccampos@ncsu.edu

Example 2.1: Consider the three-dimensional lattice $L \subset \mathbb{Z}^3$ generated by the three vectors

$$\boldsymbol{v_1} = (4, 2, 11), \qquad \boldsymbol{v_2} = (-4, 2, 0), \qquad \boldsymbol{v_3} = (-12, -25, 8).$$

We now form a matrix using these vectors as the rows of the matrix,

$$A = \begin{pmatrix} 4 & 2 & 11 \\ -4 & 2 & 0 \\ -12 & -25 & 8 \end{pmatrix}$$

Now consider the matrix in $GL_n(\mathbb{Z})$ with determinant -1,

$$U = \begin{pmatrix} -4 & -4 & 3 \\ 5 & 6 & 0 \\ 0 & 1 & 4 \end{pmatrix}$$

We can now take the product of these matrices to find another matrix.

$$B = UA = \begin{pmatrix} -36 & -91 & -20 \\ -4 & 22 & 55 \\ -52 & -98 & 32 \end{pmatrix}$$

Proposition 2.1 tells us that the rows of the matrix B are also a basis for L.

Definition: Let L be a lattice of dimension n with basis vectors $\boldsymbol{v_1}, ..., \boldsymbol{v_n}$. The *fundamental domain* of L corresponding to this basis is the set

$$\mathcal{F}(\boldsymbol{v_1}, ..., \boldsymbol{v_n}) = \{t_1\boldsymbol{v_1} + ... + t_n\boldsymbol{v_n} : 0 \leqq t_i < 1\} \tag{2}$$

Proposition 2.2: Let L be a lattice of dimension n and fundamental domain $\mathcal{F}$. Then every vector $\boldsymbol{w} \in \mathbb{R}^n$ can be written in the form

$$\boldsymbol{w} = \boldsymbol{t} + \boldsymbol{v} \text{ for a unique } \boldsymbol{t} \in \mathcal{F} \text{ and a unique } \boldsymbol{v} \in L.$$

In other words, Proposition (2.2) states that in much the same way that a real number can be decomposed into the sum of its integer and fractional parts (i.e. $\forall$x$\in \mathbb{R}$, x $= \lfloor x \rfloor + \{x\}$), so too can any vector be decomposed into the sum of a point on a lattice and a point in its fundamental domain. Equivalently, the union of translated fundamental domains across all vectors in the lattice exactly covers $\mathbb{R}^n$. Figure 1 illustrates a 2-dimensional lattice and its translated fundamental domains.

Definition: Let L be a lattice of dimension n with fundamental domain $\mathcal{F}$. Then the n-dimensional volume of $\mathcal{F}$ is called the *determinant* of L and is denoted by $det(L)$.

Calculating the determinant of a lattice is as simple as computing the determinant of the matrix whose rows are the basis vectors for that lattice. It is important to note that all the fundamental domains of a lattice L have the same determinant (up to the sign) regardless of the chosen basis. This makes the determinant an important invariant of a lattice. In example (2.1), $det(A) = 1492$. Even though the rows of matrix B defined an alternate basis for L, $|det(B)| = 1492$. Now, if one were to try to maximize the volume of the parallelepiped outlined by the basis vectors of L, then the largest volume is obtained when the basis vectors are all orthogonal to one another. This leads to an upper bound on the determinant of a lattice.

Definition: Let L be a lattice with basis vectors $\boldsymbol{v_1}, ..., \boldsymbol{v_n}$ and fundamental domain $\mathcal{F}$. Then the *Hadamard Inequality* states that

$$det(L) = Vol(\mathcal{F}) \leq \|\boldsymbol{v_1}\|...\|\boldsymbol{v_n}\| \tag{3}$$

The closer the basis vector are to being orthogonal, the closer Hadamard's Inequality is to being an equality. This leads to an important metric for the basis vectors of a lattice.

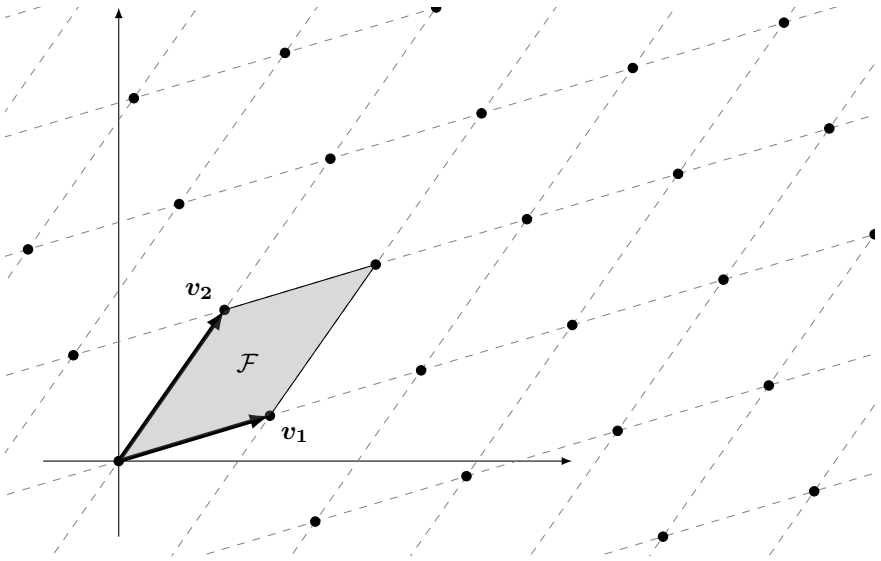

Figure 1: A lattice L and its fundamental domain $\mathcal{F}$.

Definition: Let L be a lattice of dimension n with basis vectors $\mathcal{B} = \{\boldsymbol{v_1}, ..., \boldsymbol{v_n}\}$. Then the *Hadamard Ratio* of the basis $\mathcal{B}$ is

$$\mathcal{H}(\mathcal{B}) = \left(\frac{det(L)}{\|\boldsymbol{v_1}\|...\|\boldsymbol{v_n}\|}\right)^{1/n} \tag{4}$$

In essence, the Hadamard Ratio gives a measurement for how orthogonal the basis vectors of a lattice are. Since the Hadamard Ratio satisfies $0 < \mathcal{H}(\mathcal{B}) \leq 1$, bases with values close to 1 are more orthogonal. The reason for taking the nth root is to make the Hadamard Ratio consistent across dimensions since the ratio will tend to be smaller the higher the dimension. As we will soon see, bases with $\mathcal{H}(\mathcal{B})$ close to 1 are considered "good" while bases with $\mathcal{H}(\mathcal{B})$ close to 0 are considered "bad". In Example (2.1), the basis described by matrix A has a Hadamard Ratio of 0.9911 whereas the basis described by matrix B has a Hadamard Ratio of 0.1296. Therefore, even though both are bases for the same lattice, A is a better basis than B.

3 Difficult Problems in Lattices

The two most important problems problems associated with lattices are those of finding the shortest nonzero vector in a lattice and finding a vector in a lattice that is closest to a given nonlattice vector. It turns out that both of these problems are $\mathcal{NP}$-hard and only become more difficult as the dimension n of the lattice grows. More formally,

The Shortest Vector Problem (SVP): Find a vector $\boldsymbol{v} \in L$ that minimizes the Euclidean norm $\|\boldsymbol{v}\|$.

The Closest Vector Problem (CVP): Given a vector $\boldsymbol{w} \in \mathbb{R}^n$ that is not in L, find a vector $\boldsymbol{v} \in L$ that minimizes the Euclidean norm $\|\boldsymbol{w} - \boldsymbol{v}\|$.

In general, CVP can oftentimes be reduced to SVP in a slightly higher dimension, thus making the difficulty of both problems fairly equivalent. There also exist variants of SVP and CVP such as the Shortest Basis Problem (SBP), the Approximate Shortest Vector Problem (apprSVP) and the Approximate Closest Vector Problem (apprCVP). In practice, the CVP and its approximate variant are most commonly used in lattice-based cryptography. What follows are some important results related to the expected values of the SVP and CVP in dimension n.

Theorem 3.1 (Hermite's Theorem): Every lattice L of dimension n contains a nonzero vector $\boldsymbol{v} \in L$ satisfying

$$\|\boldsymbol{v}\| \leq \sqrt{n}\, det(L)^{1/n} \tag{5}$$

This inequality can be improved for large n by using the formula

$$\|\boldsymbol{v}\| \lessapprox \sqrt{\frac{2n}{\pi e}} \cdot det(L)^{1/n} \tag{6}$$

This approximation improves formula (5) by a factor of $\sqrt{2/\pi e} \approx 0.4839$.

Definition: Let L be a lattice of dimension n. The *Gaussian expected shortest length* is

$$\sigma(L) = \sqrt{\frac{n}{2\pi e}} \cdot det(L)^{1/n} \tag{7}$$

In other words, the Gaussian heuristic says that the shortest nonzero vector in a randomly chosen lattice will satisfy $\|\boldsymbol{v}_{shortest}\| \approx \sigma(L)$. However, for small values of n, it's better to use the more precise formula

$$\sigma(L) = (\Gamma(1 + n/2) det(L))^{1/n} / \sqrt{\pi} \tag{8}$$

where $\Gamma(x)$ is the Gamma function.

We will find that the Gaussian heuristic is useful in quantifying the difficulty of finding the shortest vector in a lattice. In particular, if the actual shortest vector of a lattice L is significantly shorter than the expected $\sigma(L)$, then a lattice reduction algorithm such as LLL will be much more effective at finding the shortest vector. The same is true for the CVP since the closest lattice vector $\boldsymbol{v} \in L$ to a random point $\boldsymbol{w} \in \mathbb{R}^n$ will satisfy $\|\boldsymbol{w} - \boldsymbol{v}\| \approx \sigma(L)$.

Example 3.1: Consider the lattice described in Example (2.1) with dimension $n = 2$ and determinant $det(L) = 1492$. Then the Gaussian expected shortest length $\sigma(L)$ derived from formula (8) is equal to 21.79. This means that we can expect the shortest vector in L (and similarly the distance from a random nonlattice point to its closest lattice point) to be around 21.79.

4 Babai's Algorithm

Let $L \subset \mathbb{R}^n$ be a lattice with basis $\boldsymbol{v_1}, ..., \boldsymbol{v_n} \in \mathbb{Z}^n$ with the added property that each of its basis vectors are pairwise orthogonal to each other (i.e. $\boldsymbol{v_i} \cdot \boldsymbol{v_j} = 0$ for all $i \neq j$). Also, let $\boldsymbol{w} \in \mathbb{R}^n$ be a point not on L. Then finding the vector $\boldsymbol{v} \in L$ that is closest to $\boldsymbol{w}$ is easy enough. We first write $\boldsymbol{w}$ as a linear combination of the basis vectors of L (by solving a system of linear equations) as follows:

$$\boldsymbol{w} = t_1 \boldsymbol{v_1} + ... + t_n \boldsymbol{v_n} \text{ with } t_1, ..., t_n \in \mathbb{R}^n \tag{9}$$

Then the closest vector $\boldsymbol{v} = a_1 \boldsymbol{v_1} + ... + a_n \boldsymbol{v_n} \in L$ (with $a_1, ..., a_n \in \mathbb{Z}^n$) will be the one where each a_i is the closest integer to the corresponding t_i. Since we know that the translates of $\mathcal{F}$ by the elements of L cover all of $\mathbb{R}^n$, any $\boldsymbol{w} \in \mathbb{R}^n$ will be located in a unique translate $\mathcal{F} + \boldsymbol{v}$ by an element $\boldsymbol{v} \in L$. In essence, by rounding the coefficients to the closest integer, we are selecting the vertex of the parallelepiped $\mathcal{F} + \boldsymbol{v}$ that is closest to $\boldsymbol{w}$ as a tentative solution to the CVP. This is easy enough since by equation (2), the coefficients $t_1, ..., t_n$ will be in $[0, 1)$ and we need only replace a t_i by 0 if it is less than $\frac{1}{2}$ or replace it by 1 if it is greater than or equal to $\frac{1}{2}$.

Unfortunately, this result is due only to the fact that the basis vectors for L are pairwise orthogonal to each other. This means that this procedure is not guaranteed to work for any arbitrary basis of L. In fact, this issue becomes much worse when the basis vectors aren't very orthogonal and as the dimension of L increases. Both of these factors result in a parallelepiped that is so elongated that the closest vertex will be quite far from the target point. However, if the basis vectors are reasonably orthogonal to each other, we can expect

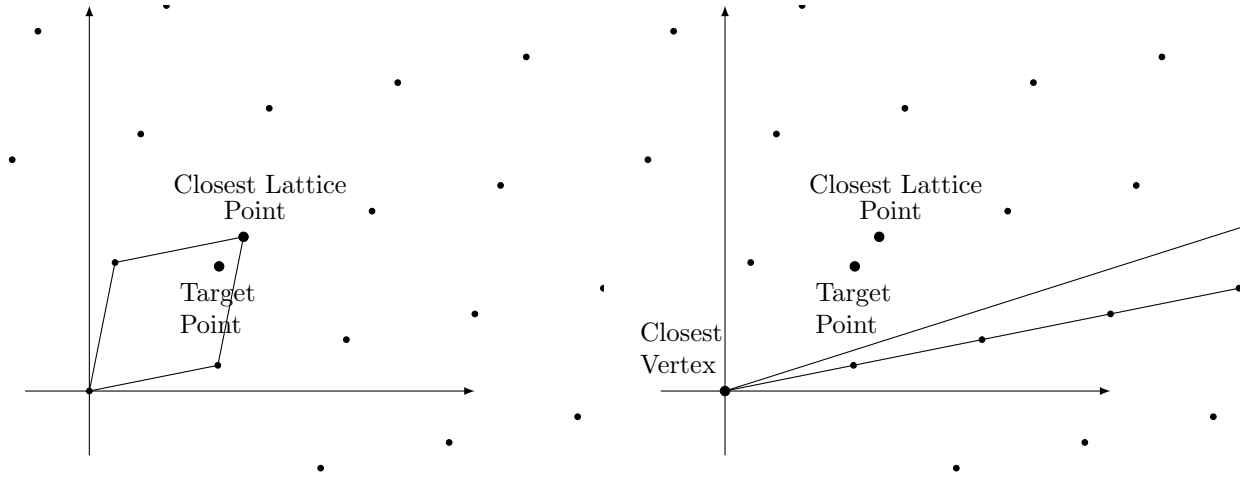

Figure 2: Babai's Algorithm produces different results with different bases.

this procedure to successfully solve the CVP. Figure 2 illustrates this attempt. More formally, we introduce Babai's Closest Vector Algorithm:

Theorem 4.1 (Babai's Algorithm): Let L be a lattice with basis $\boldsymbol{v_1}, ..., \boldsymbol{v_n}$ and let $\boldsymbol{w} \in \mathbb{R}^n$ be an arbitrary vector. If the basis vectors are sufficiently orthogonal to one another (i.e. with a Hadamard ratio close to 1), then the following algorithm solves the CVP:

Write $\boldsymbol{w} = t_1\boldsymbol{v_1} + ... + t_n\boldsymbol{v_n}$ with $t_1, ..., t_n \in \mathbb{R}^n$.
Set $a_i = \lfloor t_i \rceil$ for $i = 1, ..., n$.
Return the vector $\boldsymbol{v} = a_1\boldsymbol{v_1} + ... + a_n\boldsymbol{v_n}$.

Figure 3: Babai's Algorithm

Example 4.1: Again, consider the lattice first described in Example (2.1). We are going to use Babai's Algorithm to find a vector in L that is closest to the vector $\boldsymbol{w} = (834, 741, 532)$. We first express $\boldsymbol{w}$ as a linear combination of $\boldsymbol{v_1}$, $\boldsymbol{v_2}$, and $\boldsymbol{v_3}$ using real coordinates. That is to say, we need to find $t_1, t_2, t_3 \in \mathbb{R}$ such that

$$\boldsymbol{w} = t_1\boldsymbol{v_1} + t_2\boldsymbol{v_2} + t_3\boldsymbol{v_3}.$$

This gives the following three linear equations:

$$834 = 4t_1 - 4t_2 - 12t_3, \qquad 741 = 2t_1 + 2t_2 - 25t_3, \qquad 532 = 11t_1 + 0t_2 + 8t_3$$

Or in matrix notation,

$$(834, 741, 532) = (t_1, t_2, t_3)\begin{pmatrix} 4 & 2 & 11 \\ -4 & 2 & 0 \\ -12 & -25 & 8 \end{pmatrix}$$

Solving this system of linear equations gives $t_1 \approx 69.05, t_2 \approx -54.11$, and $t_3 \approx -28.45$. We now round these coefficients to the nearest integer and compute

$$\boldsymbol{v} = \lfloor t_1 \rceil\boldsymbol{v_1} + \lfloor t_2 \rceil\boldsymbol{v_2} + \lfloor t_3 \rceil\boldsymbol{v_3} = 69(4, 2, 11) - 54(-4, 2, 0) - 28(-12, -25, 8) = (828, 730, 535)$$

Then $\boldsymbol{v}$ is in L and $\boldsymbol{v}$ should be close to $\boldsymbol{w}$. Indeed we find that $\|\boldsymbol{v} - \boldsymbol{w}\| \approx 12.89$, much closer than the expected 21.79 from Example (3.1). This is to be expected, since the vectors in the given basis are fairly orthogonal to each other as seen by the fact that the Hadamard Ratio of matrix A was 0.9911.

We now try to solve the same closest vector problem in the same lattice but this time using the basis described by matrix B. The system of linear equations

$$(834, 741, 532) = (t_1, t_2, t_3)\begin{pmatrix} -36 & -91 & -20 \\ -4 & 22 & 55 \\ -52 & -98 & 32 \end{pmatrix}$$

has the solution $(t_1, t_2, t_3) \approx (-2597.28, -2064.01, 1940.85)$ so we set

$$\boldsymbol{v'} = -2597(-36, -91, -20) - 2064(-4, 22, 55) + 1941(-52, -98, 32) = (816, 701, 532)$$

Then $\boldsymbol{v'} \in L$ but $\boldsymbol{v'}$ is not particularly close to $\boldsymbol{w}$, since $\|\boldsymbol{v'} - \boldsymbol{w}\| \approx 43.86$, which is greater than the Gaussian expected shortest distance for this lattice. Again, this is shown by the fact that the Hadamard ratio of the basis given by matrix B is 0.1296. Keep in mind that this disparity becomes much greater in higher dimensions.

5 The GGH Public Key Cryptosystem

The GGH cryptosystem is a straightforward application of the ideas discussed in the previous sections. Alice's private key is a good basis $\mathcal{B}_{good}$ for a lattice L and her public key is a bad basis $\mathcal{B}_{bad}$. Bob's message is a vector $\boldsymbol{m}$, which he uses to form a linear combination $\sum m_i \boldsymbol{v_i}$ of the vectors in $\mathcal{B}_{bad}$. He then perturbs the sum by adding a random vector $\boldsymbol{r}$. Since Alice knows a good basis for L, she can use Babai's algorithm to find $\boldsymbol{v}$, and then expresses $\boldsymbol{v}$ in terms of the bad basis to recover $\boldsymbol{m}$. Since Eve only knows $\mathcal{B}_{bad}$, she is unable to solve the CVP in L. It is important to notice that GGH is a probabilistic cryptosystem since the same plaintext can be encrypted into different ciphertexts depending on the choice of $\boldsymbol{r}$. One must be careful not to leak information by reusing the same perturbation vector or plaintext. The following table describes the process in more detail.

Key Creation
Alice begins by choosing a set of linearly independent vectors $\boldsymbol{v_1}, ..., \boldsymbol{v_n} \in \mathbb{Z}^n$ that are reasonably orthogonal to one another. One way to do this is to fix a parameter d and chose the coordinates of $\boldsymbol{v_1}, ..., \boldsymbol{v_n}$ randomly between $-d$ and d. Let V be the n-by-n matrix whose rows are the vectors $\boldsymbol{v_1}, ..., \boldsymbol{v_n}$. This will be her private key. Let L be the lattice generated by these vectors. Alice must now create a bad basis to be her public key. She can do this multiplying V by a matrix $U \in GL_n(\mathbb{Z})$. One way to generate U is by multiplying together a large number of randomly chosen elementary matrices. She computes $W = UV$. The resulting row vectors $\boldsymbol{w_1}, ..., \boldsymbol{w_n}$ of W are the bad basis for L and also Alice's public key.
Encryption
In order for Bob to send a message to Alice, he first selects a small vector $\boldsymbol{m}$ with integer coefficients as his plaintext. He also chooses a small random perturbation vector $\boldsymbol{r}$. He could choose the coordinates of $\boldsymbol{r}$ randomly between $-\delta$ and δ, where δ is a fixed public parameter. He then computes the cipher text $\boldsymbol{e}$ using the following formula. $$\boldsymbol{e} = \boldsymbol{m}W + \boldsymbol{r} = \sum_{i=1}^{n} m_i \boldsymbol{w}_i + \boldsymbol{r}$$ Notice that $\boldsymbol{e}$ is not a lattice point but it is close to the lattice point $\boldsymbol{m}W$, since $\boldsymbol{r}$ is small.
Decryption
Alice uses Babai's Algorithm as described in Theorem (4.1) on Bob's ciphertext $\boldsymbol{e}$ with the good basis $\boldsymbol{v_1}, ..., \boldsymbol{v_n}$. Since $\boldsymbol{r}$ is small and she is using a good basis, the lattice point that she finds is $\boldsymbol{m}W$. Finally, she multiplies by W^{-1} to recover the original message $\boldsymbol{m}$.

Table 1: The GGH Public Key Cryptosystem

Example 5.1: Let Alice's private basis be $\boldsymbol{v_1} = (4, 13)$ and $\boldsymbol{v_2} = (-57, -45)$. She then generates the matrix $U \in GL_n(\mathbb{Z})$

$$\begin{pmatrix} -1118 & -525 \\ 707 & 332 \end{pmatrix}$$

to generate her private basis $\boldsymbol{w_1} = (25453, 9091)$ and $\boldsymbol{w_2} = (-16096, -5749)$. In this case, the determinant of L is 561. Note that the Hadamard Ratio of the private basis is 0.7536 whereas the Hadamard Ratio of the public basis is 0.0011. Suppose that Bob sends the encrypted message $\boldsymbol{e} = (155340, 55483)$. Alice uses Babai's Algorithm to find the closest lattice vector to $\boldsymbol{e}$. She must first solve the following system of linear equations.

$$(155340, 55483) = (t_1, t_2)\begin{pmatrix} 4 & 13 \\ -57 & -45 \end{pmatrix}$$

which has solutions $t_1 \approx -6823.12$ and $t_2 \approx -3204.08$. Rounding these coefficients to the nearest integer and multiplying by the vectors in the private basis gives (155336, 55481) as the closest vector to $\boldsymbol{e}$ in L. This tells us that Bob's random perturation vector $\boldsymbol{r}$ is equal to (4, 2). We now multiply by W^{-1} to recover the plaintext (8, 3).
Notice that if Eve applies Babai's Algorithm with the public key, the plain text she recovers is (-8, -23) which is not the same as the one recovered by Alice.

One of the advantages to using GGH is that it only requires $\mathcal{O}(k^2)$ operations as opposed to RSA, ECC, or Elgamal which require $\mathcal{O}(k^3)$ operations. However, this does come at the cost of key size since key size is typically $\mathcal{O}(n^2 log\ n)$. For dimension $n > 400$ (the point at which the underlying CVP becomes intractable) the size of the public key is approximately 128kB. Compare this to the maximum 0.5kB key size used by RSA.
It is important to mention that many variations of GGH have been proposed throughout the years, each with differences in the initial parameters. These variations attempt to optimize the size of the perturbation vector relative to vectors in the lattice so that Babai's Algorithm can guarantee to remove this vector and recover the correct plaintext. However, some of the more secure variations still require hours to generate a public key and several minutes to decrypt a message. As it stands, GGH is still an open area of research but an important proof-of-concept for lattice-based cryptosystems.

6 The LLL Lattice Reduction Algorithm

So far, the GGH cryptosystem relies on Babai's algorithm failing to solve the CVP with a bad basis. In this section, I describe the LLL algorithm which attempts to solve the apprCVP in low dimensions. In short, the LLL algorithm uses the Gram-Schmidt orthogonalization process to try to "straighten" out the vectors of a bad basis and increase its Hadamard ratio. When complete, the LLL algorithm outputs a better basis whose vectors are much shorter then those of the input basis.

We begin by exploring the idea of alternately subtracting multiples of one basis vector to another as a way of reducing their size. Let $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ be vectors in $\mathbb{Z}^n$ with $\|\boldsymbol{v_1}\| < \|\boldsymbol{v_2}\|$. Ideally, we would project $\boldsymbol{v_2}$ onto the orthogonal complement of $\boldsymbol{v_1}$ since this would make the resulting vector orthogonal to $\boldsymbol{v_1}$. Specifically, the new vector $\boldsymbol{v_2^*}$ would be

$$\boldsymbol{v_2^*} = \boldsymbol{v_2} - \frac{\boldsymbol{v_1} \cdot \boldsymbol{v_2}}{\|\boldsymbol{v_1}\|^2}\boldsymbol{v_1}$$

However, this does not guarantee that that $\boldsymbol{v_2^*}$ will be in the lattice L. In reality, we are only allowed to subtract integer multiples of $\boldsymbol{v_1}$ from $\boldsymbol{v_2}$. This means that the best we can do is round and replace $\boldsymbol{v_2}$ with the vector

$$\boldsymbol{v_2^*} = \boldsymbol{v_2} - m\boldsymbol{v_1} \qquad \text{where} \qquad m = \left\lfloor \frac{\boldsymbol{v_1} \cdot \boldsymbol{v_2}}{\|\boldsymbol{v_1}\|^2} \right\rceil$$

If $\boldsymbol{v_2}$ is still longer than $\boldsymbol{v_1}$, we stop. Otherwise we swap $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ and repeat the process. In higher dimensions, we can extend this process by subtracting from $\boldsymbol{v_k}$ integer multiples of the previous vectors $\boldsymbol{v_1}, ..., \boldsymbol{v_{k-1}}$ so as to make $\boldsymbol{v_k}$ smaller. Thus, the goal of LLL is to produce a list of short vectors in increasing order of length. It is interesting to note that this size reduction depends on the order in which the vectors are fed

into the algorithm. Before I describe the algorithm in full, we need to introduce the idea of a supplementary Gram-Schmidt orthogonal basis.

Suppose that we are given a basis $\mathcal{B} = \{\boldsymbol{v_1}, ..., \boldsymbol{v_n}\}$ for a lattice L. We start by letting $\boldsymbol{v_1^*} = \boldsymbol{v_1}$, and then for $i \geq 2$ we apply the Gram-Schmidt process and let

$$\boldsymbol{v_i^*} = \boldsymbol{v_i} - \sum_{j=1}^{i-1} \mu_{i,j} \boldsymbol{v_j^*}, \quad \text{where} \quad \mu_{i,j} = \frac{\boldsymbol{v_i} \cdot \boldsymbol{v_j^*}}{\|\boldsymbol{v_j^*}\|^2} \quad \text{for} \quad 1 \leq j \leq i-1 \tag{10}$$

Then $\mathcal{B}^* = \{\boldsymbol{v_1^*}, ..., \boldsymbol{v_n^*}\}$ is an orthogonal basis for the vector space spanned by $\mathcal{B}$. While $\mathcal{B}^*$ is not a basis for L, they do have the same determinant.

Definition: Let $\mathcal{B} = \{\boldsymbol{v_1}, ..., \boldsymbol{v_n}\}$ be a basis for a lattice L and let $\mathcal{B}^* = \{\boldsymbol{v_1^*}, ..., \boldsymbol{v_n^*}\}$ be the associated Gram-Schmidt orthogonal basis as described above. Then the basis $\mathcal{B}$ is said to be LLL *reduced* if it satisfies the following two conditions:

$$\textbf{Size Condition} \qquad |\mu_{i,j}| = \frac{|\boldsymbol{v_i} \cdot \boldsymbol{v_j^*}|}{\|\boldsymbol{v_j^*}\|^2} \leq \frac{1}{2} \quad \text{for all} \quad 1 \leq j < i \leq n. \tag{11}$$

$$\textbf{Lovász Condition} \qquad \|\boldsymbol{v_i^*}\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \|\boldsymbol{v_{i-1}^*}\|^2 \quad \text{for all} \quad 1 < i \leq n. \tag{12}$$

The intuition behind both of these conditions is that while the size condition reduces the length of vectors, the Lovász condition ensures that they are roughly orthogonal and ordered by length. By going back and forth between the two and reducing whenever possible, the LLL algorithm is able to generate a better basis.

Input a basis $\{\boldsymbol{v_1}, ..., \boldsymbol{v_n}\}$ for a lattice L
Set $k = 2$
Set $\boldsymbol{v_1^*} = \boldsymbol{v_1}$
Loop while $k \leq n$
  Loop down from $j = k - 1$ to 1
    Set $\boldsymbol{v_k} = \boldsymbol{v_k} - \lfloor \mu_{k,j} \rceil \boldsymbol{v_j}$ [Size Reduction]
  End j loop
  If $\|\boldsymbol{v_k^*}\|^2 \geq \left(\frac{3}{4} - \mu_{k,k-1}^2\right) \|\boldsymbol{v_{k-1}^*}\|^2$ [Lovász Condition]
    Set $k = k + 1$
  Else
    Swap $\boldsymbol{v_{k-1}}$ and $\boldsymbol{v_k}$ [Swap Step]
    Set $k = \max(k - 1, 2)$
  End If
End k Loop
Return LLL reduced basis $\{\boldsymbol{v_1}, ..., \boldsymbol{v_n}\}$

Figure 4: The LLL lattice reduction algorithm

Example 6.1: Let L be the lattice generated by the rows of the matrix

$$M = \begin{pmatrix} 20 & 51 & 35 & 59 & 73 & 73 \\ 14 & 48 & 33 & 61 & 47 & 83 \\ 95 & 41 & 48 & 84 & 30 & 45 \\ 0 & 42 & 74 & 79 & 20 & 21 \\ 6 & 41 & 49 & 11 & 70 & 67 \\ 23 & 36 & 6 & 1 & 46 & 4 \end{pmatrix}$$

Note that the smallest vector in this basis is $\|\boldsymbol{v_6}\| = 63.198$. After performing LLL on this basis, the resulting matrix is

$$M^{LLL} = \begin{pmatrix} -6 & -3 & -2 & 2 & -26 & 10 \\ 11 & 30 & 2 & 5 & -6 & 24 \\ -14 & -10 & 14 & -48 & -3 & -6 \\ -3 & 24 & 43 & 23 & -33 & -38 \\ 64 & -44 & -16 & -46 & -13 & 4 \\ -28 & -25 & 41 & 5 & 30 & 39 \end{pmatrix}$$

Indeed, both matrices have the same determinant as $det(M) = det(M^{LLL}) = \pm 21242880806$. Furthermore, as expected, the LLL reduced matrix has a much better Hadamard Ratio than the original matrix

$$\mathcal{H}(M) = 0.45726 \text{ and } \mathcal{H}(M^{LLL}) = 0.93408$$

so the vectors in the LLL basis are more orthogonal. The smallest vector in the LLL-reduced basis is $\|\boldsymbol{v_1}\| = 28.792$, which is a significant improvement over the original basis. Compare these values to the Gaussian Expected Shortest Length of $\sigma(L) = 40.024$

As explained above, the LLL algorithm returns a basis in which the vectors are *quasi-orthogonal*, i.e., they are reasonably orthogonal to one another. This means that we can combine the LLL algorithm with Babai's algorithm to try to solve the apprCVP and thereby attack a lattice-based cryptosystem such as GGH.
It is important to note that LLL is a polynomial-time algorithm in the dimension of the lattice. Many improvements have been proposed to the original algorithm which attempt to improve the output at the cost of increased running time. However, since the the underlying CVP problem that LLL is trying to solve is $\mathcal{NP}$-hard, these algorithms lie at the border between different time complexity classes. This means that a minor improvement can cause the resulting algorithm to no longer terminate in polynomial time.

Also, as part of this paper, I coded a version of the LLL algorithm that operates on matrices of dimension 6. The source code (written in the C language) is available upon request.

7 Primer for Convolution Polynomial Rings

In this section, I introduce some definitions and notation related to the polynomial quotient rings that are used by the NTRU public key cryptosystem. I assume that the reader is familiar with basic ring theory.

Definition: The *ring of convolution polynomials of rank N* is the quotient ring

$$R = \frac{\mathbb{Z}[x]}{(x^N - 1)} \tag{13}$$

Similarly, the *ring of convolution polynomials of rank N, modulo q* is the quotient ring

$$R_q = \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{(x^N - 1)} \tag{14}$$

Without getting into any technical proofs, the general idea is that for R_q there are two independent moduli: one for the coefficients and another for the exponents. In other words, we must reduce the coefficients of a polynomial modulo q and the exponents modulo N in order for the resulting polynomial to be in R_q. In the case of R, only the exponents are required to be in $\mathbb{Z}/N\mathbb{Z}$ while the coefficients are free to be anywhere in $\mathbb{Z}$.

The idea of multiplication between two polynomials can be generalized as a *convolution product* (denoted by $\star$). We simply multiply the two polynomials as usual and then reduce the exponents modulo N for elements in R or reduce the exponents modulo N and the coefficients modulo q for elements in R_q.

Example 7.1: Let $N = 5$ and let $\boldsymbol{a}(x), \boldsymbol{b}(x) \in R$ be the polynomials

$$\boldsymbol{a}(x) = 2 + 2x - 2x^2 + x^3 - 2x^4 \quad \text{and} \quad \boldsymbol{b}(x) = -1 + 3x - 3x^2 - 3x^3 - 3x^4$$

Then the product

$$\begin{aligned}\boldsymbol{a}(x) \star \boldsymbol{b}(x) &= -2 + 4x + 2x^2 - 19x^3 - x^4 - 9x^5 + 9x^6 + 3x^7 + 6x^8 \\ &= -2 + 4x + 2x^2 - 19x^3 - x^4 - 9 + 9x + 3x^2 + 6x^3 \\ &= -11 + 13x + 5x^2 - 13x^3 - x^4\end{aligned}$$

If we instead work in the ring R_4, then we reduce the coefficients modulo 4 to obtain

$$\boldsymbol{a}(x) \star \boldsymbol{b}(x) = 1 + x + x^2 + 3x^3 + 3x^4 \text{ in } R_4 = (\mathbb{Z}/4\mathbb{Z})[x]/(x^5 - 1)$$

This natural mapping from R to R_q in which we simply reduce the coefficients modulo q, is in fact a ring homomorphism. It turns out to be convenient to have a consistent way of going in the other direction, namely a mapping $R_q \to R$. Thus we define the following:

Definition: Let $\boldsymbol{a}(x) \in R_q$. The *center-lift* of $\boldsymbol{a}(x)$ to R is the unique polynomial $\boldsymbol{a'}(x)$ satisfying $\boldsymbol{a'}(x)$ mod $q = \boldsymbol{a}(x)$ where the chosen coefficients are in the interval

$$-\frac{q}{2} < a'_i \leq \frac{q}{2}$$

It is important to note that this reverse mapping is not a homomorphism.

Example 7.2: Let $\boldsymbol{c}(x) = \boldsymbol{a}(x) \star \boldsymbol{b}(x) = 1 + x + x^2 + 3x^3 + 3x^4 \in R_4$ be the product from Example 7.1. Since $q = 4$, the coefficients of the center-lift of $\boldsymbol{c}(x)$ are chosen from $\{-2, -1, 0, 1, 2\}$. Then

$$\text{Center-lift of } \boldsymbol{a}(x) = \boldsymbol{a'}(x) = 1 + x + x^2 - 2x^3 - 2x^4 \in R$$

Proposition 7.1: Let q be prime. Then $\boldsymbol{a}(x) \in R_q$ has a multiplicative inverse if and only if

$$\gcd(\boldsymbol{a}(x),\ x^N - 1) = 1 \text{ in } (\mathbb{Z}/q\mathbb{Z})[x]$$

Whenever this is the case, the inverse $\boldsymbol{a}(x)^{-1} \in R_q$ can be computed using the Extended Euclidean Algorithm to find the polynomials $\boldsymbol{u}(x), \boldsymbol{v}(x) \in (\mathbb{Z}/q\mathbb{Z})[x]$ satisfying

$$\boldsymbol{a}(x)\boldsymbol{u}(x) + (x^N - 1)\boldsymbol{v}(x) = 1 \text{ in } (\mathbb{Z}/q\mathbb{Z})[x].$$

Then $\boldsymbol{a}(x)^{-1} = \boldsymbol{u}(x)$ in R_q.

Example 7.3: Let $N = 5$, $q = 3$, and $\boldsymbol{a}(x) = -x^3 + x^2 + 1 \in R_3$. We first use the Extended Euclidean Algorithm to compute the greatest common divisor of $-x^3 + x^2 + 1$ and $x^5 - 1$. Note that since we are working modulo 3, $-x^3 + x^2 + 1 = 2x^3 + x^2 + 1$ and $x^5 - 1 = x^5 + 2$. Thus

$$\begin{aligned}x^5 + 2 &= (2x^3 + x^2 + 1)(2x^2 + 2x + 2) + (2x^2 + x) \\ 2x^3 + x^2 + 1 &= (2x^2 + x)(x) + (1) \\ 2x^2 + x &= (1)(2x^2 + x) + (0)\end{aligned}$$

Since the last nonzero remainder is 1, $\gcd(x^5 + 2,\ 2x^3 + x^2 + 1) = 1$ and therefore $-x^3 + x^2 + 1$ has an inverse in $(\mathbb{Z}/3\mathbb{Z})[x]$. Using the substitution method yields

$$(2x)(x^5 - 1) + (2x^3 + 2x^2 + 2x + 1)(-x^3 + x^2 + 1) = 1$$

Proposition 7.1 tells us that $\boldsymbol{a}(x)^{-1} = 2x^3 + 2x^2 + 2x + 1$. Indeed we can check that

$$\begin{aligned}\boldsymbol{a}(x) \star \boldsymbol{a}(x)^{-1} &= (-x^3 + x^2 + 1)(2x^3 + 2x^2 + 2x + 1) \\ &= -2x^6 + 3x^3 + 3x^2 + 2x + 1 \\ &= x + 3x^3 + 3x^2 + 2x + 1 \\ &= 1\end{aligned}$$

Definition: For any positive integer d_1 and d_2, the set of *trinary polynomials* is

$$\mathcal{T}(d_1, d_2) = \left\{ \boldsymbol{a}(x) \in R: \begin{array}{c} \boldsymbol{a}(x) \text{ has } d_1 \text{ coefficients equal to } 1 \\ \boldsymbol{a}(x) \text{ has } d_2 \text{ coefficients equal to } -1 \\ \boldsymbol{a}(x) \text{ has all other coefficients equal to } 0 \end{array} \right\}$$

8 The NTRU Public Key Cryptosystem

Most cryptosystems such as RSA, Diffie-Hellman, Elgamal, and ECC are considered group-based cryptosystems because their underlying hard problem only involves one operation. NTRU, on the other hand, is a ring-based cryptosystem since it's described in terms of convolution polynomial rings. However, its underlying hard problem can be interpreted as either a SVP or CVP in a lattice. In fact, while NTRU doesn't use lattices directly, it derives its security from the difficulty of solving the SVP.

We begin by fixing an integer $N \geq 1$ and two moduli p and q such that $gcd(N, q) = gcd(p, q) = 1$. We then let R, R_p, and R_q be the convolution polynomial rings

$$R = \frac{\mathbb{Z}[x]}{(x^N - 1)} \qquad R_p = \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{(x^N - 1)} \qquad R_q = \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{(x^N - 1)} \tag{15}$$

Notice that a polynomial $\boldsymbol{a}(x) \in R$ can be viewed as an element of R_p or R_q by reducing its coefficients modulo p or q. To go in the other direction, we take the center-lift of an element in R_p or R_q to arrive in R. Next, Alice chooses two random polynomials

$$\boldsymbol{f}(x) \in \mathcal{T}(d+1, d) \qquad \text{and} \qquad \boldsymbol{g}(x) \in \mathcal{T}(d, d) \tag{16}$$

and computes the inverses of $\boldsymbol{f}(x)$ in R_q and R_p using the Extended Euclidean Algorithm.

$$\boldsymbol{F_q}(x) = \boldsymbol{f}(x)^{-1} \text{ in } R_q \qquad \text{and} \qquad \boldsymbol{F_p}(x) = \boldsymbol{f}(x)^{-1} \text{ in } R_p \tag{17}$$

She then computes her public key as

$$\boldsymbol{h}(x) = \boldsymbol{F_q}(x) \star \boldsymbol{g}(x) \text{ in } R_q \tag{18}$$

Her private key is made up of both $\boldsymbol{f}(x)$ and $\boldsymbol{F_p}(x)$. Now, Bob must choose a plaintext $\boldsymbol{m}(x) \in R$ whose coefficients satisfy $-\frac{1}{2}p < m_i \leq \frac{1}{2}p$. This guarantees that $\boldsymbol{m}(x)$ is the center-lift of a polynomial in R_p. He then chooses a random polynomial $\boldsymbol{r}(x) \in \mathcal{T}(d, d)$ (known as a *blinding value*) and computes the following ciphertext in the ring R_q:

$$\boldsymbol{e}(x) \equiv p\boldsymbol{h}(x) \star \boldsymbol{r}(x) + \boldsymbol{m}(x) \quad (mod\ q) \tag{19}$$

Notice that by multiplying $\boldsymbol{h}(x) \star \boldsymbol{r}(x)$ by p, this term will reduce to 0 in R_p once we migrate to that ring. Meanwhile in R_q, it serves to hide the plaintext. The first step in decryption is for Alice to compute the product $\boldsymbol{a}(x) \equiv \boldsymbol{f}(x) \star \boldsymbol{e}(x) \pmod{q}$. We observe that

$$\begin{aligned}
\boldsymbol{a}(x) &\equiv \boldsymbol{f}(x) \star \boldsymbol{e}(x) \pmod{q} \\
&\equiv \boldsymbol{f}(x) \star (p\boldsymbol{h}(x) \star \boldsymbol{r}(x) + \boldsymbol{m}(x)) \pmod{q} \\
&\equiv p\boldsymbol{f}(x) \star \boldsymbol{F_q}(x) \star \boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{q} \\
&\equiv p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{q}
\end{aligned}$$

Proposition 8.1: If the NTRU parameters (N, p, q, d) are chosen such that $q > (6d+1)p$, then the resulting polynomial $p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x)$ is equal to $\boldsymbol{a}(x)$ not just modulo q, but also exactly in R.

If the condition of proposition (8.1) is met, then decryption will never fail. Finally, she center-lifts $\boldsymbol{a}(x)$ to an element of R and performs the following computation modulo p.

$$\boldsymbol{b}(x) \equiv \boldsymbol{F_p}(x) \star \boldsymbol{a}(x) \pmod{p} \tag{20}$$

We can see that $\boldsymbol{b}(x)$ is equal to the plaintext $\boldsymbol{m}(x)$ by analysing the computation

$$\begin{aligned}
\boldsymbol{b}(x) &\equiv \boldsymbol{F_p}(x) \star \boldsymbol{a}(x) \pmod{p} \\
&\equiv \boldsymbol{F_p}(x) \star (p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x)) \pmod{p} \\
&\equiv \boldsymbol{F_p}(x) \star \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{p} \\
&\equiv \boldsymbol{m}(x) \pmod{p}
\end{aligned}$$

Notice how $\boldsymbol{f}(x)$ is also canceled out, although this time by its inverse in R_p. The following table summarizes the NTRU cryptosystem.

Public Parameter Creation
A trusted party chooses public parameters (N, p, q, d) with N and p prime, $gcd(N, q) = gcd(p, q) = 1$, and $q > (6d + 1)p$. Typical parameters for standard security are $(N, p, q, d) = (251, 3, 128, 2)$.
Key Creation
Alice chooses a private polynomial $\boldsymbol{f}(x) \in \mathcal{T}(d+1, d)$ that is invertible in R_q and R_p. If $\boldsymbol{f}(x)$ fails to have an inverse in either, she discards this $\boldsymbol{f}(x)$ and chooses a new one. She then uses the Extended Euclidean Algorithm to compute $\boldsymbol{F_q}(x)$ and $\boldsymbol{F_p}(x)$, the inverses of $\boldsymbol{f}(x)$ in R_q and R_p respectively. Next, she chooses a private $\boldsymbol{g}(x) \in \mathcal{T}(d, d)$ and publishes her public key $\boldsymbol{h}(x) = \boldsymbol{F_q}(x) \star \boldsymbol{g}(x)$.
Encryption
Bob chooses a plaintext $\boldsymbol{m}(x) \in R_p$ and a random blinding value $\boldsymbol{r}(x) \in \mathcal{T}(d, d)$. He then uses Alice's public key $\boldsymbol{h}(x)$ to compute the ciphertext $$\boldsymbol{e}(x) \equiv p\boldsymbol{r}(x) \star \boldsymbol{h}(x) + \boldsymbol{m}(x) \pmod{q}$$
Decryption
Alice computes $$\boldsymbol{f}(x) \star \boldsymbol{e}(x) \equiv p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{q}$$ Then she center-lifts this to $\boldsymbol{a}(x) \in R$ and recovers Bob's plaintext by computing $$\boldsymbol{m}(x) \equiv \boldsymbol{F_p}(x) \star \boldsymbol{a}(x) \pmod{p}$$

Table 2: The NTRU Public Key Cryptosystem

Example 8.1: Alice and Bob agree on NTRU parameters $(N, p, q, d) = (7, 2, 29, 2)$. Notice that since $29 = q > (6d + 1)p = 26$, proposition (8.1) ensures that decryption will work. Alice chooses her public key to be

$$\boldsymbol{h}(x) = 23x^6 + 24x^5 + 23x^4 + 24x^3 + 23x^2 + 23x + 23$$

Bob then sends the plaintext message $\boldsymbol{m}(x) = x^5 + 1$ using the random blinding value $\boldsymbol{r}(x) = x^6 + x^3 + x + 1$. He performs the following computation to encrypt his plaintext.

$$\begin{aligned}
\boldsymbol{e}(x) &\equiv p\boldsymbol{r}(x) \star \boldsymbol{h}(x) + \boldsymbol{m}(x) \pmod{q} \\
&\equiv 2(7x^6 + 6x^5 + 7x^4 + 6x^3 + 6x^2 + 6x + 5) + (x^5 + 1) \pmod{29} \\
&\equiv 14x^6 + 13x^5 + 14x^4 + 12x^3 + 12x^2 + 12x + 11 \pmod{29}
\end{aligned}$$

This is the ciphertext that Bob sends to Alice. Now suppose that Alice's private key is $\boldsymbol{f}(x) = x^5 + x^4 + x^2 + x + 1$ and $\boldsymbol{F_2}(x) = x^6 + x^5 + 1$. She decrypts Bob's ciphertext $\boldsymbol{e}(x)$ as follows:

$$\begin{aligned}
\boldsymbol{f}(x) \star \boldsymbol{e}(x) &\equiv p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{q} \\
&\equiv 7x^6 + 4x^5 + 5x^4 + 5x^3 + 4x^2 + 5x + 4 \pmod{29}
\end{aligned}$$

Center-lifting this polynomial to $\boldsymbol{a}(x) \in R$ gives the same polynomial, that is $\boldsymbol{a}(x) = 7x^6 + 4x^5 + 5x^4 + 5x^3 + 4x^2 + 5x + 4$. She performs a final computation, this time modulo 3.

$$\begin{aligned}
\boldsymbol{m}(x) &\equiv \boldsymbol{F_p}(x) \star \boldsymbol{a}(x) \pmod{p} \\
&\equiv (x^6 + x^5 + 1)(7x^6 + 4x^5 + 5x^4 + 5x^3 + 4x^2 + 5x + 4) \pmod{2} \\
&\equiv 16x^6 + 15x^5 + 16x^4 + 14x^3 + 14x^2 + 14x + 13 \\
&\equiv x^5 + 1 \pmod{2}
\end{aligned}$$

Indeed she is able to retrieve Bob's plaintext.

One of the advantages NTRU has over GGH is that it's much faster to implement. The most time-consuming part of encryption and decryption is the convolution product. Since these convolution products are being

performed on ternary polynomials, they can be computed without any multiplications; they require $\frac{2}{3}N^2$ additions and subtractions. Thus NTRU encryption and decryption takes $\mathcal{O}(N^2)$ operations, where each step is extremely fast.

Nevertheless, it still remains to be seen what exactly is the difficult problem the eavesdropper Eve is stuck trying to solve. Multiplying both sides of equation (18) by $\boldsymbol{f}(x)$ reveals that

$$\boldsymbol{f}(x) \star \boldsymbol{h}(x) \equiv \boldsymbol{g}(x) \pmod{q} \tag{21}$$

where $\boldsymbol{f}(x)$ and $\boldsymbol{g}(x)$ have very small coefficients. Therefore breaking NTRU comes down to solving the following problem.

Definition: Given $\boldsymbol{h}(x)$, find ternary polynomials $\boldsymbol{f}(x)$ and $\boldsymbol{g}(x)$ satisfying $\boldsymbol{f}(x) \star \boldsymbol{h}(x) \equiv \boldsymbol{g}(x) \pmod{q}$. This is known as the NTRU Key Recovery Problem.

As we will see in the next section, this problem can be reformulated as a Shortest Vector Problem in a special kind of lattice.

9 NTRU as a Lattice Cryptosystem

So far, it's no obvious how NTRU is a lattice-based cryptosystem. We can draw some parallels between GGH and NTRU. For example, both are probabilistic cryptosystems that mix the plaintext with a random element to produce a ciphertext. In the case of GGH, this was a random perturbation vector which was later rounded off in Babai's algorithm. In the case of NTRU, this was the blinding value which later reduced to zero in the ring R_p.

As it turns out, we can create a lattice based on the coefficients of the public key.
Let $\boldsymbol{h}(x) = h_0 + ... + h_{N-1}x^{N-1}$ be an NTRU public key. Then the *NTRU Lattice* $L_{\boldsymbol{h}}^{NTRU}$ associated with $\boldsymbol{h}(x)$ is the $2N$-dimensional lattice spanned by the rows of the matrix

$$L_{\boldsymbol{h}}^{NTRU} = \left(\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{N-1} \\ 0 & 1 & \dots & 0 & h_{N-1} & h_0 & \dots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & h_1 & h_2 & \dots & h_0 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right)$$

Notice that $L_{\boldsymbol{h}}^{NTRU}$ is composed of four N-by-N blocks:

Upper left block = Identity matrix,
Lower left block = Zero matrix,
Lower right block = q times the identity matrix,
Upper right block = Cyclical permutations of the coefficients of $\boldsymbol{h}(x)$.

It is often convenient to abbreviate the NTRU matrix as

$$L_{\boldsymbol{h}}^{NTRU} = \begin{pmatrix} I & \boldsymbol{h} \\ 0 & qI \end{pmatrix} \tag{22}$$

We are now going to identify each pair of polynomials in R

$$\boldsymbol{a}(x) = a_0 + ... + a_{N-1}x^{N-1} \quad \text{and} \quad \boldsymbol{b}(x) = b_0 + ... + b_{N-1}x^{N-1}$$

with a $2N$-dimensional vector

$$(\boldsymbol{a}, \boldsymbol{b}) = (a, ..., a_{N-1}x^{N-1}, b_0, ..., b_{N-1}x^{N-1}) \in \mathbb{Z}^{2N}.$$

Proposition 9.1: Assuming that equation (21) holds, the vector $(\boldsymbol{f}, \boldsymbol{g})$ is in the NTRU lattice $L_{\boldsymbol{h}}^{NTRU}$ and is the shortest nonzero vector in this lattice (with very high probability).

In other words, the private key is hidden in this lattice in the shape of the shortest vector. This means that Eve can determine Alice's private NTRU key if she can solve the SVP in $L_{\boldsymbol{h}}^{NTRU}$. Thus the security of NTRU depends on the difficulty of solving the SVP. As mentioned in the section on LLL, if N is large, the LLL algorithm does not find very small vectors. Experimental results on some of the best variations of LLL suggest that values of N in the range of 250 to 1000 yield security levels comparable to current secure implementations of RSA, Elgamal, and ECC.

10 Conclusion

One final topic that is worth mentioning is that of lattice-based digital signature schemes. For GGH, such a scheme is straight-forward. Alice chooses a good basis for a lattice and publishes a bad basis as her public key. She then signs a document $\boldsymbol{d} \in \mathbb{Z}^n$ using Babai's algorithm with the good basis to compute a vector that is close to $\boldsymbol{d}$. She then rewrites this in terms of the bad public basis and publishes it as her signature. In order for Bob to verify this signature, he would simply verify that it's sufficiently close to $\boldsymbol{d}$. Only someone with a good basis could have solved this apprCVP and so the scheme works.

In the case of NTRU, such a scheme is more complicated to construct and is outside of the scope of this paper. For further research, the name of the signature scheme is NTRUMLS (NTRU modular lattice signature scheme) and is based on the idea of rejection sampling.

We have seen how the GGH and NTRU lattice-based cryptosystems offer promising alternatives for modern cryptography, especially in the context of quantum resistance. These cryptosystems tend to be easy to implement as the numbers involved are much smaller than those used by typical cryptosystems. We also saw how lattice reduction algorithms such as LLL are a potential avenue of attack for these cryptosystems. However, the operating characteristics of these algorithms is not very well understood and still remains an active area of research.

11 References

1. An Introduction to Mathematical Cryptography, by Jeffrey Hoffstein et al., Springer, 2014, pp. 373-470.

2. Advances in the GGH Lattice-based Cryptosystem, by Thomas Rond, California State University, 2016.

3. Hoffstein J., Lieman D., Pipher J., Silverman J. "NTRU: A Public Key Cryptosystem", NTRU Cryptosystems, Inc. (www.ntru.com).

4. Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures, by Phong Q. Nguyen and Oded Regev, International Association for Cryptologic Research, 2006.