Lattice-Based Encryption Schemes and its Applications to Homomorphic Encryption

Ahmad Faris Durrani Bin Ahmad Shahrir, Leyan Pan, Kevin Hutto, Dr. Vincent Mooney

VIP Secure Hardware Spring 2020

Dec. 07, 2020

Contents

1 Introduction

Homomorphic encryption is a type of encryption that allows performing operation on the ciphertext without having access to the plaintext. While the algorithm are still not efficient enough for practical applications, homomorphic encryption has potential in many areas such as voting, storage of sensitive personal information and analyzing demo-graphical data. In 2009, Gentry proposed the first plausible algorithm for fully homomorphic encryption [4] and various improvements have been built upon this result, significantly increasing the efficiency of homomorphic encryption. In Gentry's original implementation, lattice-based cryptography is used as a basis of the Homomorphic encryption scheme. Lattice-based cryptography still lies at the heart of many fully homomorphic encryption schemes. In this report, we build on previous VIP works of [14] [12] and illustrates various lattice-based encryption schemes and briefly describes how Gentry used lattice-based cryptography to construct the first fully homomorphic encryption scheme. In addition, this sub-team hopes the incoming VIP sub-teams would make use of this report and expand upon our research into homomorphic encryption.

2 Terminology

$\mathbb{R}$ The set of all real numbers: $\{\cdots, -1, 1.12, 0, \pi, 99 \times 10^{99}, \cdots\}$

$\mathbb{Z}$ The set of all integers (whole numbers): $\{\cdots, -2, -1, 0, 1, 2, \cdots\}$

$\forall$ For any

$\exists$ There exists

Vector Space A vector space V is a subset of $\mathbb{R}^n$ with the property that $\alpha_1 \boldsymbol{v}_1 + \alpha_1 \boldsymbol{v}_2 + \cdots + \alpha_m \boldsymbol{v}_n \in V$ for given vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_m \in V$ and scalar factors $\alpha_1, \alpha_2, \cdots, \alpha_m \in \mathbb{R}$, where $m < n$ [11]

Linear Combination $\alpha_1 \boldsymbol{v}_1 + \alpha_1 \boldsymbol{v}_2 + \cdots + \alpha_m \boldsymbol{v}_n \in V$ is a linear combination of $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_m \in V$

Euclidean distance Let $\boldsymbol{v} = (x_1, x_2, \cdots, x_n) \in V$, the Euclidean norm "distance" or vector length is

$$||\boldsymbol{v}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Orthogonality Two Euclidean vectors $\boldsymbol{u}, \boldsymbol{v} \in V$ are orthogonal if they are perpendicular, i.e., they form a right angle: $\boldsymbol{u} \cdot \boldsymbol{v} = 0$ [7]

Ring A set of elements equipped with the $+$ (addition) and $*$ (multiplication) operators; every two elements in the ring must produce a third element in the ring when added or multiplied [11]

Lattice The points in a Euclidean vector space that can be reached with an integer linear combination of a set of basis vectors usually of integer entries [15]

SVP The shortest vector problem in a lattice; to find the shortest non-zero vector in the lattice [15]

CVP The closest vector problem in a lattice; to find the shortest point that is part of a given lattice space to a given target point [15]

Polynomial An expression which is a linear combination of the powers of a variable. The greatest power is called the degree of the polynomial. E.g., $1 + 2x^2 + x^3$ is a polynomial of degree 3

$\mathbb{Z}[n]$ The collection of all polynomials with coefficients taken from $\mathbb{Z}$ forms a ring $\mathbb{Z}[n]$ under the usual operations of polynomial addition and multiplication [7]

$$\mathbb{Z}[n] = \{a_0 + a_1x + a_2x^2 + \cdots + a_nx^n | a_0, a_1, \cdots, a_n \in \mathbb{Z}\}.$$

Encryption The process of encoding a message or information in such a way that only authorized parties can access it

Homomorphic Encryption A kind of encryption scheme which allows a (untrusted) third party (e.g., cloud, service provider) to perform certain computable functions on the encrypted data while preserving the features of the function and format of the encrypted data [12]

Partially Homomorphic Encryption A homomorphic encryption scheme that can only perform one type of homomorphic operation on the plaintext (usually either addition or multiplication) any number of times [12]

Fully Homomorphic Encryption A homomorphic encryption scheme that can perform arbitrary homomorphic operation on ciphertexts (encrypted plaintexts) any number of times [12]

Somewhat Fully Homomorphic Encryption A homomorphic encryption scheme that can perform both addition and multiplication but can only do so for a limited number of time [12]

Leveled Fully Homomorphic Encryption A Somewhat Fully Homomorphic Encryption Scheme that can unlimitedly increase the number of homomorphic operations allowed by increasing its key size and/or bit size [12]

Symmetric Encryption A kind of encryption that involves only one secret key to cipher and decipher information. The main disadvantage of the symmetric key encryption is that all parties involved have to exchange the key used to encrypt the data before they can decrypt it which can be obtained by an adversary

Asymmetric Encryption/Public Key Encryption A kind of encryption that uses two keys – one to encrypt a plaintext (public key) and the other to decrypt a ciphertext (private key)

Cryptosystem/Encryption Scheme A suite of cryptographic algorithms needed to implement a particular security service. Typically, it comprises at least 3 algorithms – one for key generation, one for encryption and one for decryption

Probabilistic Encryption Probabilistic encryption is the use of randomness in an encryption algorithm, so that when encrypting the same message several times it will, in general, yield different ciphertexts

Ciphertext Data that was the encrypted value of some encryption scheme

Plaintext Opposing term to ciphertext which refers to data that was not encrypted

Lattice Mathematical structure that represents linear integer combinations of a vectors of an integer basis of n-dimensional space. More information in later sections

Bootstrapping Homomorphically decrypting a doubly encrypted ciphertext in order to reduce the noise parameter. More information in later sections

KDM Security A encryption scheme is KDM secure if providing the encrypted information of the secret key does not comprehend security

Modular Arithmetic A integer arithmetic system where all numbers are represented as their remainder when divided by a common integer. All numbers with the same remainder under then common positive integer is viewed as equivalent to each other under this modular arithmetic system. The notion "under modulo m" or "mod m" represents a modular arithmetic system with the common positive integer m

Modular Multiplicative Inverse The modular multiplicative inverse of x under modulo m is an positive integer x^{-1} less than m that satisfies $x \times x^{-1} = 1 \mod m$

3 Basic Lattice Concepts and Problems

3.1 Lattice Concepts

Introduction

Given n linearly independent vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n \in \mathbb{R}^m$, the lattice generated by them is defined as

$$\mathcal{L}(\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n) = \{x_1\boldsymbol{b}_1, x_2\boldsymbol{b}_2, \cdots, x_n\boldsymbol{b}_n | x_1, x_2, \cdots, x_n \in \mathbb{Z}\}.$$

We refer to $\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n$ as a basis of the lattice. Equivalently, if we define B as the $m \times n$ matrix whose columns are $\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n$, then the latice generated by B is

$$\mathcal{L}(B) = \mathcal{L}(\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n) = \{B\boldsymbol{x} | \boldsymbol{x} \in \mathbb{Z}^n\}.$$

Note that we can also represent a lattice space in a matrix form where each of its basis vectors constitute its rows.

We say that the rank of the lattice is n and its dimension is m. If $n = m$, the lattice is called a full-rank lattice. In this paper, all lattices are full-rank lattices unless specified otherwise. The

more general case is not substantially different.

An example of a lattice is one generated by basis $\{(1,0),(0,1)\}$. This basis produces the lattice $\mathbb{Z}^2$, the lattice of all integer points which is also a lattice space in $\mathbb{R}^2$. (see below) [15]

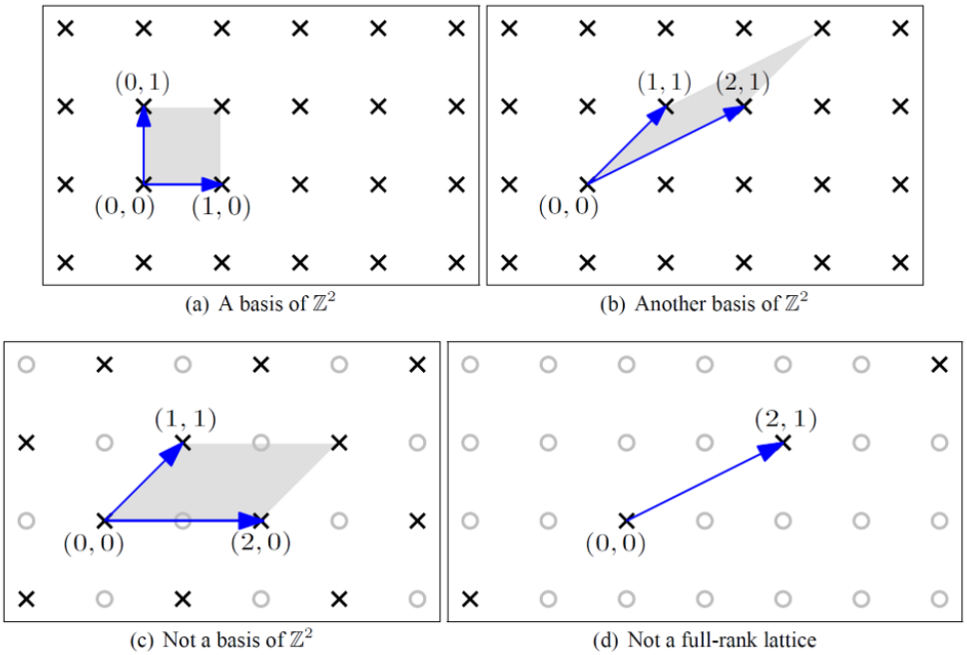

Figure 1: Examples of lattice spaces with different bases. Source: [15]

As seen in Figure 1 (a) and (b), the same lattice space can be produced with more than one distinct basis. In this case, the lattice space $\mathbb{Z}^2$ can be induced with both the bases $\{(0,1),(1,0)\}$ and $\{(1,1),(2,1)\}$. Figure 1 (c) shows the basis $\{(1,1),(2,0)\}$ does not induce the lattice space $\mathbb{Z}^2$ since some points in the $\mathbb{Z}^2$ like $(1,0)$ and $(0,1)$ are not included. Figure 1 (d) shows a basis $(2,1)$ with dimension 2 and rank 1 in the $\mathbb{R}^2$ space induces a non-full rank lattice space.

Additionally, the shaded ares are called the fundamental parallelpiped or fundamental domain of the lattice space. The reader may read more about fundamental domains in [7].

A lattice has multiple bases

Like all lattice spaces, their bases are not unique. For example, $(1,1)$ and $(2,1)$ generate $\mathbb{Z}^2$. Yet another basis of $\mathbb{Z}^2$ is given by $\{(2005,1),(2006,1)\}$. On the other hand, $\{(1,1)^T,(2,0)^T\}$ is not a basis of $\mathbb{Z}^2$ – instead, it generates the lattice of all integer points whose coordinates sum to an even number. The lattice $\mathbb{Z} = L((1))$ is a one-dimensional full-rank lattice.

Consequently, for any lattice spaces, there can be some 'good' bases and some 'bad' bases. A

'good' basis is one with nearly orthogonal and relatively short vectors by Euclidean norms. A good basis is almost always preferred when solving lattice problems like the SVP and CVP since they require less time and complexity to compute.

As an example, basis A_1 and basis A_2 below both produce the same lattice space but basis A_1 is considerably shorter and therefore better to use in solving problems like the SVP than basis A_2. Recall that each row of the matrix represent a basis vector.

$$A_1 = \begin{pmatrix} 449857 & 1731 & 72769 \\ 224936 & 870 & 36380 \\ 224927 & 861 & 36390 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 6 & 0 & 1 \\ 1 & 3 & 1 \\ 7 & 3 & -5 \end{pmatrix}$$

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

A matrix $U \in \mathbb{Z}^{n \times n}$ is called unimodular if the determinant of U is ± 1.

For example, the matrix

$$\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$

is unimodular as its determinant is 1.

Two bases $B_1, B_2 \in \mathbb{R}^{m \times n}$ are equivalent if and only if $B_2 = B_1 U$ for some unimodular matrix U.

Proof[1]**:** First assume that $L(B_1) = L(B_2)$. Then for each of the n columns b_i of $B_2, b_i \in L(B_1)$. This implies that there exists an integer matrix $U \in \mathbb{Z}^{n \times n}$ for which $B_2 = B_1 U$. Similarly, there exists a $V \in \mathbb{Z}^{n \times n}$ such that $B_1 = B_2 V$. Hence $B_2 = B_1 U = B_2 V U$, and we get

$$B_2^T B_2 = (VU)^T B_2^T B_2 (VU).$$

Taking determinants, we obtain that $\det(B_2^T B_2) = (\det(VU))^2 \det(B_2^T B_2))$ and hance $\det(V)\det(U) = \pm 1$. Since V, U are both integer matrices, this means that $\det(U) = \pm 1$, as required.

For the other direction, assume that $B_2 = B_1 U$ for some unimodular matrix U. Therefore each column of B_2 is contained in $L(B_1)$ and we get $L(B_2) \subseteq L(B_1)$. In addition, $B_1 = B_2 U^{-1}$, and since U^{-1} is unimodular, we similarly get that $L(B_1) \subseteq L(B_2)$. We conclude that $L(B_1) = L(B_2)$ as required.

[1]Source: Oded Regev. "Lecture 1: introduction." Lattices in Computer Science. Tel Aviv University. Fall 2004. NYU [15]

Example

As an example, the two bases of the lattice space $\mathbb{Z}^2$ discussed earlier are related by a unimodular matrix described below:

$$\begin{array}{ccc} B_1 \times U & = & B_2 \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} & = & \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} \end{array}$$

where the unimodular matrix $\begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}$ has determinant -1.

3.2 Computational Problems Related to Lattices

The fundamental computational problems associated to a lattice are those of finding a shortest nonzero vector in the lattice and of finding a vector in the lattice that is closest to a given non-lattice vector. We begin with a description of two fundamental lattice problems.

Shortest Vector Problem (SVP): Find a shortest nonzero vector in a lattice L of dimension n, i.e. find a nonzero vector $\boldsymbol{v}_{shortest} \in L$ that minimizes the Euclidean norm $||\boldsymbol{v}_{shortest}||$.

Example

Given a lattice $\mathbb{Z}^2$ with basis $(0, 1)$ and $(1, 0)$, as shown in Figure 2. Expand a uniform circle from the origin until the circle first touches a lattice point which in this case can include, among others, $(1, 0)$. The distance between the origin and $(0, 1)$ which we denote as λ_1 is equal to $||v_{shortest}||$. In other words, $1 = \sqrt{0^2 + 1^2}$ is the shortest vector distance in the lattice $\mathbb{Z}^2$.

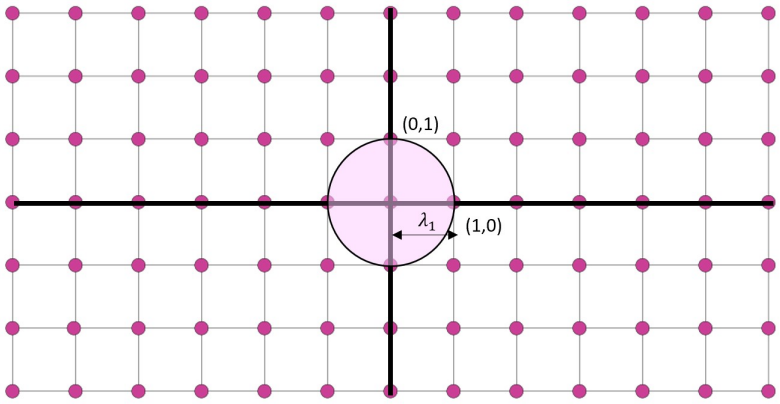

Figure 2: Finding the SVP in lattice $\mathbb{Z}^2$

A similar technique could be used to calculate the SVP in the lattice $\mathbb{Z}^3$ with basis $\{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$ in Figure 3. Expand a uniform ball from the origin until the first instance the surface touches a lattice point, which can include $(0, 0, 1)$ with a distance of 1 from the origin. It is not hard to

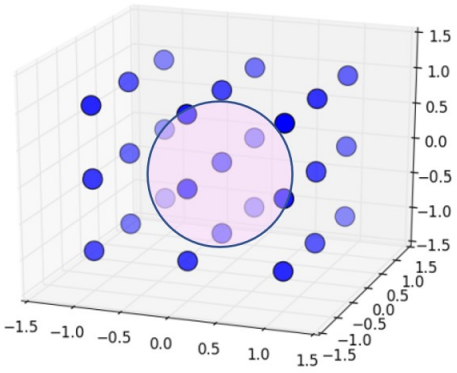

Figure 3: Finding the SVP in lattice $\mathbb{Z}^3$ (not to scale).[2]

see the higher the dimensions of the lattice is, the harder it is to calculate the SVP. In modern lattice-based cryptosystems, the dimension of a lattice can approach hundreds.

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 is closest to $\boldsymbol{t}$, i.e., find a vector $\boldsymbol{v} \in L$ that minimizes the Euclidean norm $||\boldsymbol{t} - \boldsymbol{v}||$.

Example

Using lattice $\mathbb{Z}^2$ in Figure 2 and target point $\boldsymbol{t}$ (1.1, 1.1), it is trivial to see the closest vector in the lattice to $\boldsymbol{t}$ is $(1, 1)$. We can again use the expanding circle method with center at $\boldsymbol{t}$.

Remark. Note that there may be more than one shortest nonzero vector in a lattice. For example, in $\mathbb{Z}^2$, all four of the vectors $(0, \pm 1)$ and $(\pm 1, 0)$ are solutions to SVP. This is why SVP asks for "a" shortest nonzero vector and not "the" shortest vector. A similar remark applies to CVP.

There are many important variants of SVP and CVP that arise both in theory and in practice. We describe a few of them here.

Approximate Shortest Vector Problem (apprSVP)

Let the scalar approximation factor $\gamma(n) \geq 1$ be a function of n. In a lattice L of dimension n, find a nonzero vector that is no more than $\gamma(n)$ times longer than a shortest nonzero vector. In other words, if $\boldsymbol{v}_{shortest}$ is a shortest nonzero vector in L, find a nonzero vector $\boldsymbol{v} \in L$ satisfying

$$||\boldsymbol{v}|| \leq \gamma(n) \cdot ||\boldsymbol{v}_{shortest}||.$$

Each choice of function $\gamma(n)$ gives a different apprSVP. As specific examples, one might ask for an algorithm that finds a nonzero $\boldsymbol{v} \in L$ satisfying

$$||\boldsymbol{v}|| \leq 3\sqrt{n}||\boldsymbol{v}_{shortest}|| \quad \text{or} \quad ||\boldsymbol{v}|| \leq 2^{n/2}||\boldsymbol{v}_{shortest}||.$$

[2]Figure adapted from https://www.physicstomato.com/crystal-lattice/.

Clearly an algorithm that solves the former is much stronger than one that solves the latter, but even the latter may be useful if the dimension is not too large.

Example

Take the lattice L given in Figure 4. And let the approximation factor $\gamma(n)$ be 2. Thus, we are looking a lattice point in L whose distance is $2\lambda_1(L)$ or twice the shortest length of $||v_{shortest}||$. If $\lambda_1(L) = 1$, then $2\lambda_1(L) = 2$. As a side note, $\lambda_2(L)$ represents the second shortest vector in L.

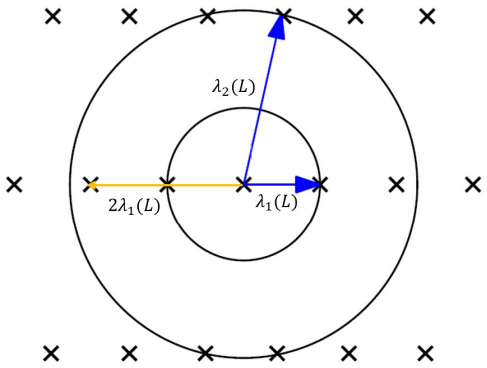

Figure 4: Finding the apprSVP in lattice L. Adapted from [15]

Approximate Closest Vector Problem (apprCVP)

This is the same as apprSVP, but now we are looking for a vector that is an approximate solution to CVP, instead of an approximate solution to SVP.

Example

Again, take lattice $\mathbb{Z}^2$ with $\boldsymbol{t} = (2.6, 1.6)$ as shown in Figure 5. The actual closest vector would be the lattice point $\boldsymbol{v} = (3, 2)$. Hence $(\text{dist})(L, \boldsymbol{t}) = \sqrt{0.4^2 + 0.4^2} = \frac{2\sqrt{2}}{5}$. By constructing a circle of radius $\gamma(n) \cdot \text{dist}(L, \boldsymbol{t})$, we can find a number of lattice points that satisfy our problem restrictions to find an approximate closest vector to $\boldsymbol{t}$ with scalar factor $\gamma(n)$.

Hardness of Problems

Both SVP and CVP are profound problems, and both become computationally difficult as the dimension n of the lattice grows. On the other hand, even approximate solutions to SVP and CVP turn out to have surprisingly many applications in different fields of pure and applied mathematics. In full generality, CVP is known to be NP-hard and SVP is NP-hard under a certain "randomized reduction hypothesis." See [9] for more detials.

In full generality, both SVP and CVP are considered to be extremely hard problems, but in practice it is difficult to achieve this idealized "full generality". In read world scenarios, cyrptosystems based

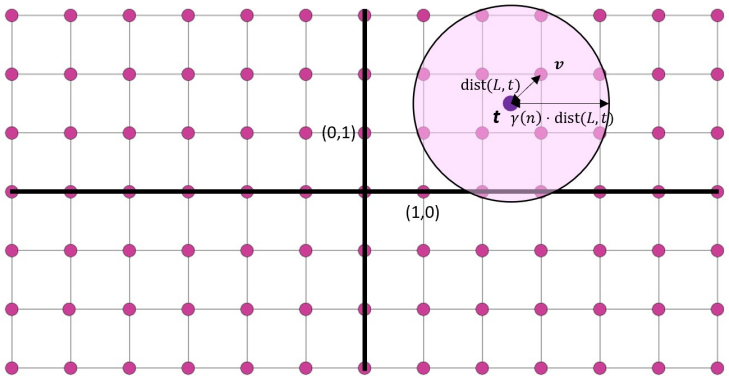

Figure 5: Finding the apprCVP in lattice $\mathbb{Z}^2$.

on NP-hard or NP-complete problems tend to rely on a particular subclass of problems, either to achieve efficiency or to allow the creation of a trapdoor. When this is done, there is always the possibility that some special property of the chosen subclass of problems makes them easier to solve than the general case.

Though there is yet a general algorithm that would solve both problems with precision, there are approximation algorithms. For instance, the Lenstra-Lenstra-Lovász (LLL) Algorithm is an approximation algorithm of the shorest vector problem, which runs in polynomial time and finds an approximation within an exponential factor of the correct answer. It is practical for solving integer linear programming, factorizing polynomials over integers, and breaking cryptosystems with enough accuracy.

Here, it is important to note that a lattice can have "good" bases and "bad" bases, which can affect the level of difficulty in solving these computational problems. The following figure shows two different bases for the same lattice. The first basis is "good" in the sense that the vectors are fairly orthogonal and consists of relatively short Euclidean length vectors; the second basis is "bad" because the angle between the basis vectors is small and it consists of relatively long Euclidean length vectors.

For instance, using Babai's Algorithm [7][pp. 403] to solve CVP on a lattice will return outputs with different distance to the target vector depending on the lattice's basis. When the method is executed using a good basis, it works effectively for returning a lattice vector that is located close to the target vector. On the contrary, Babai's may return a lattice vector that is located far from the target vector on a bad basis.

4 The GGH Encryption Scheme

The first encryption scheme that will be introduced is the the GGH Encryption Scheme. The GGH (Goldreich–Goldwasser–Halevi) encryption scheme is a relatively simple lattice-based encryption scheme whose security is based upon the CVP (Cloest Vector Problem). The subsection below illustrates the difference between a good and bad basis in solving the CVP problem.

4.1 Solving The CVP Problem

The CVP Problem is defined as follows:

Definition 1 *Given an integer lattice, represented by a basis* $\mathbf{B}$*, and a target vector* $\mathbf{c}$*, find a lattice point* $\mathbf{Bx}$ *that minimizes the distance* $||\mathbf{Bx} - \mathbf{c}||$*, where* $||\mathbf{x}||$ *denotes the Euclidean length of vector* $\mathbf{x}$*.*

Note that if the target vector $\mathbf{t}$ is in the lattice, then the problem becomes trivial as the solution is $\mathbf{t}$ itself. The difficulty of solving the CVP is strongly dependent on the "quality" of the given lattice. As an example, consider the following "good" basis for lattice $\mathbf{B}$:

$$B = \begin{pmatrix} 7 & 0 & 0 \\ 0 & 23 & 0 \\ 0 & 0 & 99 \end{pmatrix} \tag{1}$$

The lattice contains all vectors of the form $\mathbf{x} = (7a, 23b, 99c)a, b, c \in \mathbb{Z}$. The basis $\mathbf{B}$ is "good" because the basis vectors $(7, 0, 0), (0, 23, 0), (0, 0, 29)$ are othogonal to each other. In this case, the closest lattice vector to the vector $\mathbf{c} = (1, -1, 1)$, for example, can be obtained by the formula $[\mathbf{cB}^{-1}]\mathbf{B} = (0, 0, 0)$, where $[\mathbf{x}]$ denotes rounding of each entry of $\mathbf{x}$ to the nearest integer. To dissect this formula, cB^{-1} computes the real (not necessarily integers as for a lattice vector) coefficients of the basis vectors to form the vector c. $[cB^{-1}]$ rounds the coefficients to the nearest integer to form the coefficients that would result in a lattice vector. $[cB^{-1}]B$ computes the nearest of the lattice points that would forms the region that contains c. If the vectors of B are "relatively orthogonal", this vector would be the same as the closest vector to c.

Now, instead of the good basis B, consider the following basis B':

$$B' = \begin{pmatrix} 7 & 69 & -990 \\ 56 & 575 & -8514 \\ -77 & -644 & 8019 \end{pmatrix} \tag{2}$$

which still contains represents the same lattice as B, i.e. $\mathbf{x} = (7a, 23b, 99c)a, b, c \in \mathbb{Z}$. If we try to use the same method to solve the CVP, we would get a wrong result: $[\mathbf{cB'}^{-1}]\mathbf{B'} = (-56, -529, 7227)$. This is because, with a bad basis, even a small change with the given point (c) will create a large change in the coefficients of the basis vectors. The following image gives a illustration of attempting to solve the CVP problem using a bad basis:

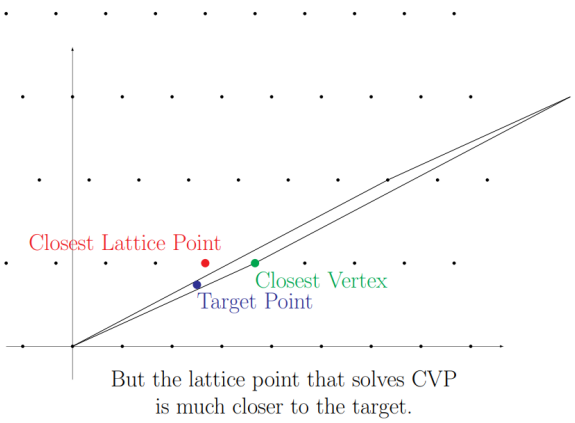

Figure 6: A good basis and a bad basis[4]

[4] The algorithm would attempt to find the closest vertex that encloses the lattice cell containing the vertex as described by the given basis. However, as shown in the image, the closest vertex is not necessarily the same as the closest lattice point. In fact, the more relatively orthogonal ("good") the basis are, and the smaller the error vector, the more likely that the closest vertex and the closest lattice point will be the same.

4.2 Key Generation

In order to define a GGH instance, we need to first define a lattice L of dimension n by giving a $n \times n$ ("good") basis matrix B with relatively orthogonal row vectors, where n is a parameter that is decided based on the required security level. The higher the value n is, the more difficult it is to break the cryptosystem and also the more time it takes to perform the encryption and decryption operations. For a secure GGH encryption scheme instance, it is generally recommended that $n > 400$ for attacks that uses the LLL [10] algorithm unfeasible.

Now that we have the "good" basis, we will create a bad basis B' by generating a random large unimodular matrix U and computing $B' = UB$ [7]. Note that a upper or lower triangular matrix with diagonal entries 1 is always unimodular, and the product of two unimodular matrix is also unimodular. One way to generate a large unimodular matrix is by generating a upper triangular and a lower triangular with random entries in the non-zero region and multiplying them together (will be illustrated in the exmaple).

After generating the matrices as stated above, the public key will be B' and the private key will be the tuple (B, U).

4.3 Encryption

The plaintext space of a GGH instance is a n dimensional integer vector in the fundamental region of the "good" basis. However, if it is desired to encrypt other types of data like a string, they can be converted into a vector that carries the same amount of information (A vector that contains n character's ASCII code in the string, for example). Suppose the target plaintext vector is m, the encryption algorithm is as follows:

First, generate a random small vector e. Usually, each entry in e is selected to have absolute value smaller than or equal to 3. The ciphertext is computed as $c = mB' + e$. Note that this vector is not in the lattice and thus we can apply the CVP algorithm using B but not B' to find the closest lattice point. e needs to be small because mB' must be the closest lattice point to $mB' + e$ for the decryption to work as expected. If we ignore the error vector e, mB' is a lattice point with the entries of m as coefficients of the ("bad") basis vectors in B'.

[4] From slide by Silverman, http://archive.dimacs.rutgers.edu/Workshops/Post-Quantum/Slides/Silverman.pdf

4.4 Decryption

The decryption is a partial application of the CVP algorithm described previously. First, the "good" basis B is used to compute the coefficients of the closest vector to c by computing $[cB^{-1}]$, where $[x]$ as usual denotes the rounding operation of each entry in a vector. However, note that this is the coefficient of the closest lattice point, or mB' during encryption, stated in terms of the basis vectors in B. However, in order to obtain m, we need to state the coefficients in terms of B', the "bad" basis, and therefore a translation between the two coefficient is needed. Luckily, the translation can be simply achieved by multiplying the inverse uni-modular matrix, U^{-1} (recall that $B' = UB$, therefore $mB' = mUB$, and if $xB = mUB$ then $m = xU^{-1}$). Therefore, the final decryption formula would be $m = [cB^{-1}]U^{-1}$.

4.5 Example of a GGH encryption scheme

To put the GGH encryption scheme together, the scheme can be described as follows:

- **Secret Key**: $B \in \mathbb{Z}^{n \times n}$, uni-modular $U \in \mathbb{Z}^{n \times n}$
- **Public Key**: $B' = UB$
- **Plaintext**: $m \in F_B$, where F_B is the fundamental region of basis B. $F_B \subset \mathbb{Z}^n$
- **Encryption**: $c = mB' + e$, where $e \in \mathbb{Z}^n$ is small random vector.
- **Decryption**: $m = [cB^{-1}]U^{-1}$,where $[x]$ denotes rounding of each entry.

As an example, we will use the following "good" lattice basis B with dimension $n = 3$ (Note that GGH with such small n is very inefficient and we're using this small)the same as the basis used in the CVP to define the lattice used in the GGH encryption instance:

$$B = \begin{pmatrix} 7 & 0 & 0 \\ 0 & 23 & 0 \\ 0 & 0 & 99 \end{pmatrix} \tag{3}$$

In order to generate a uni-modular matrix U, create a upper triangular matrix and a lower triangular matrix and set U to be their matrix multiplication product:

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 8 & 1 & 0 \\ -11 & 5 & 1 \end{pmatrix} \begin{pmatrix} 1 & 3 & -10 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 3 & -10 \\ 8 & 25 & -86 \\ -11 & -28 & 81 \end{pmatrix} \tag{4}$$

Now, compute $B' = UB$

$$B' = UB = \begin{pmatrix} 1 & 3 & -10 \\ 8 & 25 & -86 \\ -11 & -28 & 81 \end{pmatrix} \begin{pmatrix} 7 & 0 & 0 \\ 0 & 23 & 0 \\ 0 & 0 & 99 \end{pmatrix} = \begin{pmatrix} 7 & 69 & -990 \\ 56 & 575 & -8514 \\ -77 & -644 & 8019 \end{pmatrix} \tag{5}$$

The public key is B' and the secret/private key is U and B. Suppose that we are encrypting the message $m = (2, -1, 3)$. First, we must generate a small random vector e. Let $e = (-1, 1, 1)$ for this example. The encryption would be

$$c = mB' = \begin{pmatrix} 2 & -1 & 3 \end{pmatrix} \begin{pmatrix} 7 & 69 & -990 \\ 56 & 575 & -8514 \\ -77 & -644 & 8019 \end{pmatrix} + \begin{pmatrix} -1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -274 & -2368 & 30592 \end{pmatrix} \quad (6)$$

In order to decrypt the encrypted message we just generated, we need to use the public secret keys U and B to partially solve the CVP problem:

$$\begin{aligned} c &= [cB^{-1}]U^{-1} \\ &= \text{round}\left(\begin{pmatrix} -274 & -2368 & 30592 \end{pmatrix} \begin{pmatrix} 7 & 0 & 0 \\ 0 & 23 & 0 \\ 0 & 0 & 99 \end{pmatrix}^{-1} \right) \begin{pmatrix} 1 & 3 & -10 \\ 8 & 25 & -86 \\ -11 & -28 & 81 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} -39 & -103 & 309 \end{pmatrix} \begin{pmatrix} 1 & 3 & -10 \\ 8 & 25 & -86 \\ -11 & -28 & 81 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} 2 & -1 & 3 \end{pmatrix} \end{aligned}$$

Note that if we simply try to decrypt c using B' by calculating $[cB'^{-1}]$, we would get the vector $(70, -8, 4)$, which is fairly far away from the correct answer. In fact, the large n becomes, the less information the "bad" basis gives.

5 The NTRU Public Key Cryptosystem

In this section, we will talk about NTRUEncrpt, the NTRU (pronounced *en-trū*) public key cryptosystem presented in 1996 by Hoffstein, Pipher, and Silverman. [6] [7] A post-quantum cryptosystem, NTRUEncrypt is most naturally described using convolution polynomial rings, but the underlying hard mathematical problem can also be interpreted as a SVP in a lattice.

5.1 NTRUEncrypt

NTRU stands for *n-th degree truncated polynomial*. Truncated here means that the highest degree of the polynomial is restricted to a highest power. We first fix a prime N and define a few terms.

Convolution Polynomial Rings

Let R be the convolution polynomial ring:

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

where R is the polynomial $\beta(x) \in \mathbb{Z}[x]$ reduced modulo $x^N - 1$.

As a result, $\boldsymbol{a}(x) \in R$ is a polynomial with integer coefficients and degree strictly less than N. We may view a polynomial $\beta(x)$ of degree equal to or greater than N as an element of R by simply replacing each instance of x^N with 1. [13] The following example attempts to explain why is this the case when $\beta(x)$ is divided by $x^N - 1$.

Example of a Convolution Polynomial Ring

Say we have $\boldsymbol{a}_0(x) = x^4 + 3x^3 + 2x^2 + 2x + 1 \in \mathbb{Z}[x]$. And suppose we are in a ring

$$R_0 = \frac{\mathbb{Z}[x]}{x^3 - 1}.$$

Doing long division between $\boldsymbol{a}_0(x)$ over $x^3 - 1$, we get:

$$\begin{aligned} \frac{\boldsymbol{a}_0(x)}{x^3-1} &= \frac{x^4 + 3x^3 + 2x^2 + 2x + 1}{x^3 - 1} \\ &= \frac{(x+3)(x^3-1) + 2x^2 + 3x + 4}{x^3 - 1} \\ &= x + 3 + \frac{2x^2 + 3x + 4}{x^3 - 1} \end{aligned}$$

where R_0 is the remainder of the long division:

$$R_0 = 2x^2 + 3x + 4$$

which is exactly what we get when replacing all instances of x^3 with 1 in $\boldsymbol{a}_0(x)$:

$$\begin{aligned} \boldsymbol{a}_0(x) &= x^4 + 3x^3 + 2x^2 + 2x + 1 \\ &= x^3(x) + x^3(3) + 2x^2 + 2x + 1 \\ R_0 &= 1(x) + 1(3) + 2x^2 + 2x + 1 \\ &= 2x^2 + 3x + 4. \end{aligned}$$

The coefficients in the ring may be further reduced modulo various integers p and q.

Note that each polynomial in R can be treated as a vector by extracting the coefficients. For example:

$$\begin{aligned} F(x) &= x^2 + 2x + 3 = (3, 2, 1). \\ G(x) &= x + 5 = (5, 1, 0). \\ F(x) \times G(x) &= (3, 2, 1) \times (5, 1, 0) = x^3 + 7x^2 + 13x + 15 \\ &= 7x^2 + 13x + 16 \mod (x^3 - 1) \\ &= (16, 13, 7). \end{aligned}$$

Convolution multiplication

Let

$$\begin{aligned} a(X) &= a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-1} X^{N-1} \in R \\ b(X) &= b_0 + b_1 X + b_2 X^2 + \cdots + b_{N-1} X^{N-1} \in R. \end{aligned}$$

It should be trivial to see that when the two polynomials $a(X), b(X)$ multiply each other, the highest degree of the coefficient of the product $a(X) \times b(X) = c(X)$ may be bigger than N which means we have to reduce modulo the product by $x^N - 1$ to stay within in the ring R. A shorter method which does not require us to reduce modulo $c(X)$ involves setting the coefficients of $c(X) = c_0 + c_1 X + c_2 X^2 + \cdots + c_{N-1} X^{N-1}$ be:

$$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_{N-1} b_{k+1}$$

where $b_k = b_{N+k}$ if $k < 0$. In other words, $b_{k+1} = b_{k-N+1}$.

Note that $c(X)$ is equivalent to $a(X) \times b(X) \mod (X^N - 1)$ using normal polynomial multiplication.

Inverse of a polynomial in $R \mod q$

The inverse of a polynomial $a(X) \in R \mod q$ is a polynomial $a(X)^{-1} \in R$ satisfying

$$a(X) \times a(X)^{-1} = 1 \mod q.$$

As an example, given $\boldsymbol{a}(x) = 3x^4 + 10x^3 + 7x^2 - 1 \in \frac{\mathbb{Z}[x]}{x^5 - 1} \mod 16$, we find:

$$(3x^4 + 10x^3 + 7x^2 - 1)^{-1} = 5x^4 + 3x^3 + 13x^2 + 8x + 14 \mod 16$$

Ternary Polynomials

For any positive integers d_1, d_2, we let:

$$\mathcal{T}(d_1, d_2) = \boldsymbol{a}(x) \in R$$

such that $\boldsymbol{a}(x)$ has d_1 coefficients equal to 1, d_2 coefficients equal to -1, and all other coefficients equal to 0.

Polynomials in $\mathcal{T}(d_1, d_2)$ are called ternary or trinary polynomials. They are analagous to binary polynomials which have 0 and 1 coefficients only.

Center-lifiting

Let $\boldsymbol{a}(x) \in R \mod q$. The center-lift of $\boldsymbol{a}(x)$ to R is the unique polynomial $\boldsymbol{a'}(x) \in R$ satisfying:

$$\boldsymbol{a'}(x) \mod q = \boldsymbol{a}(x)$$

whose coefficients are chosen in the interval

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

For instance, if $q = 2$, the resulting center-lift of $\boldsymbol{a}(x)$ will be a binary polynomial.

As an example [7][pp. 415], let $N = 5$ and $q = 7$, and consider the polynomial

$$\boldsymbol{a}(x) = 5 + 3x - 6x^2 + 2x^3 + 4x^4 \in R \mod 7.$$

The coefficients of the resulting $\boldsymbol{a}'(x)$ are chosen from $\{-3, -2, \cdots, 2, 3\}$, so

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

Similarly, the lift of $\boldsymbol{b}(x) = 3 + 5x^2 - 6x^3 + 3x^4$ is $3 - 2x^2 + x^3 + 3x^4$. Notice that

$$(\text{Lift of } \boldsymbol{a}(x)) \times (\text{Lift of } \boldsymbol{b}(x)) = 20x + 10x^2 - 11x^3 - 14x^4$$

and

$$(\text{Lift of } \boldsymbol{a} \times \boldsymbol{b}) = -x + 3x^2 + 3x^3$$

are not equal to one another, although they are both congruent modulo 7.

NTRUEncrypt: Public Parameters

We are now ready to describe NTRUEncrypt. Alice the sender and Bob the receiver takes some public parameters (N, p, q, d) pre-determined by a trusted authority satisfying these guidelines:

1. Integer $N \geq 1$ and N is prime.
2. $\gcd(N, q) = \gcd(p, q) = 1$.
3. $q > (6d + 1)p$.

Most commonly, $N > 100$, $p = 3$, and q is a multiple of 2. Additionally, for best security as explained in [7] [pp. 424], $d \approx N/3$.

NTRUEncrypt: Key Generations

Alice's private key consists of two randomly chosen polynomials

$$\boldsymbol{f}(x) = \mathcal{T}(d + 1, d) \quad \text{and} \quad \boldsymbol{g}(x) = \mathcal{T}(d, d).$$

Alice computes the inverses

$$\boldsymbol{F}_q(x) = \boldsymbol{f}(x)^{-1} \text{ in } R \mod q \quad \text{and} \quad \boldsymbol{F}_p(x) = \boldsymbol{f}(x)^{-1} \text{ in } R \mod p.$$

If there are no inverses of $\boldsymbol{f}(x)$ within either rings, then Alice shall discard this $\boldsymbol{f}(x)$ and pick another random polynomial in $\mathcal{T}(d + 1, d)$.

Next, Alice computes

$$\boldsymbol{h}(x) = \boldsymbol{F}_q(x) \times \boldsymbol{g}(x) \quad \text{in } R \mod q.$$

The polynomial $\boldsymbol{h}$(x) shall be Alice's public key that she shares publicly. Her private key is the pair $(\boldsymbol{f}(x), \boldsymbol{F}_p(x))$. Or more succinctly, only $\boldsymbol{f}(x)$ since Alice can calculate $\boldsymbol{F}_p(x)$ with the public parameter p.

Notice that although the coefficients of $\boldsymbol{f}(x)$ are small, the coefficients of its inverse $\boldsymbol{F}_q(x)$ will look random which is a characteristic that defines its security.

NTRUEncrypt: Encryption

Bob's plaintext is a polynomial $\boldsymbol{m}(x) \in R$ whose coefficients satisfy

$$-\frac{1}{2}p < m_i < \frac{1}{2}p$$

i.e. the plaintext $\boldsymbol{m}(x)$ is a polynomial in R that is the center-lift of a polynomial in $R \mod p$. Bob chooses a random polynomial $\boldsymbol{r}(x) \in \mathcal{T}(d, d)$ and computes the ciphertext

$$\boldsymbol{e}(x) \equiv p\boldsymbol{h}(x) \times \boldsymbol{r}(x) + \boldsymbol{m}(x) \pmod{q}.$$

As a result, Bob's ciphertext $\boldsymbol{e}(x)$ is in the ring $R \mod q$.

NTRUEncrypt: Decryption

On receiving Bob's ciphertext, Alice starts the decryption process. She computes

$$\boldsymbol{a}(x) \equiv \boldsymbol{f}(x) \times \boldsymbol{e}(x) \pmod{q}.$$

She then center-lifts $\boldsymbol{a}(x)$ to an element of R and does a mod p computation:

$$\boldsymbol{b}(x) \equiv \boldsymbol{F}_p(x) \times \boldsymbol{a}(x) \pmod{p}.$$

Assuming that the parameters have been chosen properly, we now verify the polynomial $\boldsymbol{b}(x)$ is equal to the plaintext $\boldsymbol{m}(x)$.

NTRUEncrypt: Proof $\boldsymbol{b}(x) = \boldsymbol{m}(x)$

If the NTRUEncrypt parameters are chosen to satisfy $q > (6d + 1)p$, then the polynomial $\boldsymbol{b}(x)$ computed by Alice is equal to Bob's plaintext $\boldsymbol{m}(x)$.

Going through the calculation of $\boldsymbol{a}(x)$:

$$\begin{aligned}
\boldsymbol{a}(x) &\equiv \boldsymbol{f}(x) \times \boldsymbol{e}(x) \pmod{p} && (7)\\
&\equiv \boldsymbol{f}(x) \times (p\boldsymbol{h}(x) \times \boldsymbol{r}(x) + \boldsymbol{m}(x)) \pmod{p} && (8)\\
&\equiv p\boldsymbol{f}(x) \times \boldsymbol{F}_q(x) \times \boldsymbol{g}(x) \times \boldsymbol{r}(x) + \boldsymbol{f}(x) \times \boldsymbol{m}(x) \pmod{q} && (9)\\
&\equiv p\boldsymbol{g}(x) \times \boldsymbol{r}(x) + \boldsymbol{f}(x) \times \boldsymbol{m}(x) \pmod{q}. && (10)
\end{aligned}$$

Consider the resulting polynomial

$$p\boldsymbol{g}(x) \times \boldsymbol{r}(x) + \boldsymbol{f}(x) \times \boldsymbol{m}(x), \tag{11}$$

computed exactly in R, rather than modulo q. We need to bound the largest possible coefficient. The polynomials $\boldsymbol{g}(x)$ and $\boldsymbol{r}(x)$ are in $\mathcal{T}(d,d)$, so if in the convolution product $\boldsymbol{g}(x) \times \boldsymbol{r}(x)$, all of their 1's match up and all of their -1's match up, the largest possible coefficient of $\boldsymbol{g}(x) \times \boldsymbol{r}(x)$ is $2d$. Similarly, $\boldsymbol{f}(x) \in T(d+1,d)$ and the coefficients of $\boldsymbol{m}(x)$ are between $-\frac{1}{2}$ and $-\frac{1}{2}$, so the largest possible coefficient of $\boldsymbol{f}(x) \times \boldsymbol{m}(x)$ is $(2d+1) \cdot \frac{1}{2}p$.

So even if the largest coefficient of $\boldsymbol{g}(x) \times \boldsymbol{r}(x)$ happens to coincide with the largest coefficient of $\boldsymbol{r}(x) \times \boldsymbol{m}(x)$,the largest coefficient of $p\boldsymbol{g}(x) \times \boldsymbol{r}(x) + \boldsymbol{f}(x) \times \boldsymbol{m}(x)$ has magnitude at most

$$p \cdot 2d + (2d+1) \cdot \frac{1}{2}p = \left(3d + \frac{1}{2}\right) p.$$

Thus our assumption that $q > (6d+1)p$ ensures that every coefficinet of the polynomial has magnitude strictly smaller than $\frac{1}{2}q$. Hence when Alice computes $\boldsymbol{a}(x)$ modulo q (i.e. in R_q) and then lifts it to R, she recovers the exact value of the polynomial. In other words,

$$\boldsymbol{a}(x) = p\boldsymbol{g}(x) \times \boldsymbol{r}(x) + \boldsymbol{f}(x) \times \boldsymbol{m}(x)$$

is exactly in R, not merely modulo q.

Alice then simply multiplies $\boldsymbol{a}(x)$ by the inverse of $\boldsymbol{f}(x)$ mod p i.e. $\boldsymbol{F}_p(x)$, and reduces the result modulo p to obtain

$$\begin{aligned}
\boldsymbol{b}(x) &\equiv \boldsymbol{F}_p(x) \times \boldsymbol{a}(x) \pmod{p} \\
&\equiv \boldsymbol{F}_p(x) \times (p\boldsymbol{g}(x) \times \boldsymbol{r}(x) + \boldsymbol{f}(x) \times \boldsymbol{m}(x)) \pmod{p} \\
&\equiv \boldsymbol{F}_p(x) \times \boldsymbol{f}(x) \times \boldsymbol{m}(x) \pmod{p} \qquad \text{reducing mod } p \\
&\equiv \boldsymbol{m}(x) \pmod{p}.
\end{aligned}$$

Hence, $\boldsymbol{b}(x)$ and $\boldsymbol{m}(x)$ are the same modulo p.

Example (from [7] [pp. 421])

In the table below, R_q represents $R \mod q$ and R_p represents $R \mod p$.

Public Paramaters	
A trusted entity chooses values $(N, p, q, d) = (7, 3, 41, 2)$. Notice that N and p are prime, $\gcd(p, q) = \gcd(N, q) = 1$, and $q > (6d + 1)p$.	
Alice	**Bob**
• Choose $\boldsymbol{f}(x) = x^6 - x^4 + x^3 + x^2 - 1 \in \mathcal{T}(3, 2)$. • Choose $\boldsymbol{g}(x) = x^6 + x^4 - x^2 - x \in \mathcal{T}(2, 2)$. • Compute $\boldsymbol{F}_q = 8x^6 + 26x^5 + 31x^4 + 21x^3 + 40x^2 + 2x + 37 \in R_q$. • Compute $\boldsymbol{F}_p(x) = x^6 + 2x^5 + x^3 + x^2 + x + 1 \in R_p$. • Alice stores $(\boldsymbol{f}(x), \boldsymbol{F}_p(x))$ as her private key. • Compute $\boldsymbol{h}(x) = \boldsymbol{F}_q(x) \times \boldsymbol{g}(x) = 20x^6 + 40x^5 + 2x^4 + 38x^3 + 8x^2 + 26x + 30 \in R_q$. • Alice publishes $\boldsymbol{h}(x)$ as her public key.	
Encryption	
	• Choose message to send $\boldsymbol{m}(x) = -x^5 + x^3 + x^2 - x + 1$. • Choose random element $\boldsymbol{r}(x) = x^6 - x^5 + x - 1$. • Compute $\boldsymbol{e}(x) \equiv p\boldsymbol{r}(x) \times \boldsymbol{h}(x) + \boldsymbol{m}(x) \equiv 31x^6 + 19x^5 + 4x^4 + 2x^3 + 40x^2 + 3x + 25 \pmod q$. • Send $\boldsymbol{e}(x)$ to Alice.
Decryption	
• Compute $\boldsymbol{f}(x) \times \boldsymbol{e}(x) \equiv x^6 + 10x^5 + 33x^4 + 40x^3 + 40x^2 + x + 40 \pmod q$. • Center-lift $\boldsymbol{f}(x) \times \boldsymbol{e}(x)$ modulo q to obtain $\boldsymbol{a}(x) = x^6 + 10x^5 - 8x^4 - x^3 - x^2 + x - 1 \in R$. • Reduce $\boldsymbol{a}(x)$ modulo p and compute $\boldsymbol{F}_p(x) \times \boldsymbol{a}(x) \equiv 2x^5 + x^3 + x^2 + 2x + 1 \pmod p$. • Center-lifting the previous result modulo p to retrieve Bob's plaintext $\boldsymbol{m}(x) = -x^5 + x^3 + x^2 - x + 1$.	

5.2 NTRUEncrypt Security

The coefficients of the public key $\boldsymbol{h}$(x) appear to be random integers modulo q, but there is a hidden relationship

$$\boldsymbol{f}(x) \times \boldsymbol{h}(x) \equiv \boldsymbol{g}(x) (\text{mod } q), \tag{12}$$

where the coefficients of $\boldsymbol{f}(x)$ and $\boldsymbol{g}(x)$ are very small as they are ternary polynomials. Thus breaking NTRUEncrypt by finding the private key comes down to solving the following problem:

The NTRU Key Recovery Problem

Given $\boldsymbol{h}(x)$, find ternary polynomials $\boldsymbol{f}(x)$ and $\boldsymbol{g}(x)$ satisfying

$$\boldsymbol{f}(x) \times \boldsymbol{h}(x) \equiv \boldsymbol{g}(x) \pmod{q}.$$

Take note that the solution to the NTRU key recovery problem is not unique, as for any integer $0 \leq k < N$, the rotations of $\boldsymbol{f}(x)$ and $\boldsymbol{g}(x)$ i.e. $(x^k \times \boldsymbol{f}(x), x^k \times \boldsymbol{g}(x))$ are also valid solutions. More details can be found in [7] [pp. 422].

One way to view how to find a solution is by reqording the problem as a lattice problem. To find the private keys $\boldsymbol{f}(x), \boldsymbol{g}(x)$, let

$$\boldsymbol{h}(x) = h = h_0 + h_1 x + h_2 x^2 \cdots + h_{N-1} x^{N-1}$$

be the public key. Thus, the NTRU lattice $L_{\boldsymbol{h}}$ associated to $\boldsymbol{h}$ is the $2N$-dimensional lattice space spanned by the rows of the matrix:

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

where the top-left quadrant is the Identity matrix, the top-right quadrant is the cyclical permutations of the coefficients of $\boldsymbol{h}$(x), the bottom-left quadrant is the zero matrix, and the bottom-right is the Identity matrix multiplied by q.

A shorter way to present $M_{\boldsymbol{h}}$ is:

$$M_{\boldsymbol{h}} = \begin{pmatrix} I & \boldsymbol{h} \\ 0 & qI \end{pmatrix}.$$

We are going to identify each pair of polynomials

$$\boldsymbol{a}(x) = a_0 + a_1 x + \cdots + a_{N-1} x^{N-1}$$

and

$$\boldsymbol{b}(x) = b_0 + b_1 x + \cdots + b_{N-1} x^{N-1}$$

in R as a $2N-$dimensional vector

$$(\boldsymbol{a}, \boldsymbol{b}) = (a_0, a_1, \cdots, a_{N-1}, b_0, b_1, \cdots, b_{N-1}) \in \mathbb{Z}^{2N}.$$

Now, assuming the proposition $\boldsymbol{f}(x) \times \boldsymbol{g}(x) \equiv \boldsymbol{g}(x) \pmod{q}$ is true, we can have a polynomial $\boldsymbol{u}(x) \in R$ such that:

$$\boldsymbol{f}(x) \times \boldsymbol{g}(x) = \boldsymbol{g}(x) \times \boldsymbol{u}(x).$$

This is true since $\boldsymbol{u}(x)$ is the quotient of $\boldsymbol{f}(x) \times \boldsymbol{g}(x)$ modulo q.
Then it must be true that

$$(\boldsymbol{f}, -\boldsymbol{u})M_{\boldsymbol{h}} = (\boldsymbol{f}, \boldsymbol{g}).$$

Proof

It is clear that the first N coordinates of $(\boldsymbol{f}, \boldsymbol{h})M$ are $\boldsymbol{f}$. To understand why, take the first column of the product first. The value of the first column of the product is equal to the value of the dot product between the first row of $(\boldsymbol{f}, \boldsymbol{g})$ and the first column of $M_{\boldsymbol{h}}$ which will be:

$$f_0 \cdot 1 + f_1 \cdot 0 + f_2 \cdot 0 + \cdots + g_N \cdot 0 = f_0.$$

Next, consider what happens when we multiply the next N columns of $M_{\boldsymbol{h}}$ which have h_k coefficient values at the top with the vector $(\boldsymbol{f}, -\boldsymbol{u})$. We get the quantity:

$$h_k f_0 + h_{k-1} f_1 + \cdots + h_{k+1} f_{N-1} - q u_k,$$

which is the k^{th} entry of the vector $\boldsymbol{f}(x) \times \boldsymbol{g}(x) - q\boldsymbol{u}(x)$, which is equal to $\boldsymbol{g}(x)$. Hence, the second N coordinates of the product will be the coefficients of $\boldsymbol{g}(x)$. We can say that the vector $(\boldsymbol{f}, \boldsymbol{g})$ is derived from a linear combination of the rows of $M_{\boldsymbol{h}}$. Hence, $(\boldsymbol{f}, \boldsymbol{g}) \in L_{\boldsymbol{h}}$, shown by the succint computation:

$$(\boldsymbol{f}, -\boldsymbol{u}) \begin{pmatrix} 1 & \boldsymbol{h} \\ 0 & qI \end{pmatrix} = (\boldsymbol{f}, \boldsymbol{f} \times \boldsymbol{h} - q\boldsymbol{u}) = (\boldsymbol{f}, \boldsymbol{g}).$$

In other words, to get the vector $(\boldsymbol{f}, \boldsymbol{g})$, we only need to multiply the first row of $M_{\boldsymbol{h}}$ with f_0, added to the product of the second row with f_1, added to the product of the third row with f_2, and so on.

Recall that the matrix $M_{\boldsymbol{h}}$ is public and the hidden vector $(\boldsymbol{f}, \boldsymbol{g})$ is our private key. The basis vectors which make up $M_{\boldsymbol{h}}$ may be long and complex but if an adversary is able to obtain the short and good basis of $(\boldsymbol{f}, \boldsymbol{g})$, they will have theoretically broken NTRUEcrypt. $(\boldsymbol{f}, \boldsymbol{g})$ is a short vector comprising of only the coefficients from the trinary $\{0, -1, 1\}$, hence $(\boldsymbol{f}, \boldsymbol{g})$ is also a short basis in $L_{\boldsymbol{h}}$. In other words, solving the approximate SVP for lattice $L_{\boldsymbol{h}}$ is one way an adversary can break NTRUEncrypt.

Briefly Quantifying NTRU's Security

The proposition above says that an adversary can determine Alice's private key if they can find an approximate shortest vector in $L_{\boldsymbol{h}}$. Thus, the security of NTRUEncrypt depends at least on the difficulty of solving $L_{\boldsymbol{h}}$ in $L_{\boldsymbol{h}}$. More generally, if the adversary can solve apprSVP to within a factor of approximately N^{ϵ} for sone $\epsilon < \frac{1}{2}$, then the short vector that the adversary finds will probably serve as a decryption key. For more details, please refer to [7][pp. 427].

Though there is no known algorithm that can solve SVP in polynomial time, there are certain lattice reduction algorithms that can solve apprSVP to within a certain factor. One of the most

well-known, the Lenstra–Lenstra–Lovász lattice basis reduction algorithm (LLL) runs in polynomial time and solves apprSVP to within a factor of 2^N, but if N is large, LLL does not find very small vectors in $L_{\boldsymbol{h}}$. A generalization of the LLL algorithm, the BKZ-LLL algorithm is able to solve apprSVP to within a factor of $\beta^{2N/\beta}$, where β is a blocksize parameter that the algorithm takes in. However, its running time is exponential in β.

Unfortunately, the characteristics of such lattice basis reduction algorithms are still not as well understood, which makes it difficult to predict theoretically the performance of such algorithms on any class of lattices and hence, how fast can one apprSVP be solved. Thus, the security of a lattice-based cryptosystem such as NTRUEncrypt is currently estimated experimentally by taking in a sequence of parameters (N, q, d) and modifying values of the paramters while holding certain ratios approximately constant.

For each set of parameters, one runs experiments using the BKZ-LLL algorithm using different block sizes β until the algorithm finds a short vector in $L_{\boldsymbol{h}}$. Then one plots the logarithm of the average running time against N and computes the best-fitting line

$$\log(\text{Running Time}) = AN + B.$$

According to Hoffstein et. al. (2014) [7][pp. 428], such experiments suggest the values of N in the range from 250 to 1000 yield security levels comparable to RSA, Elgamal, and ECC. Experiments done by Hoffstein in 1996 [17] estimate the time needed to find a target vector for NTRU with $N = 167$ and $N = 263$ gives the values in the table below:

N	T (seconds)	T (MIPS-years)
167	$1.638 \cdot 10^{11}$	$2.077 \cdot 10^{6}$
263	$3.634 \cdot 10^{19}$	$4.607 \cdot 10^{14}$

Table 1: Estimated Breaking Times for NTRU 167 and NTRU 263

Here, a MIPS-year is a "standard" measure of computational effort in cryptography: It refers to the amount of work performed, in one year, by a computer operating at the rate of one million operations per second (1 MIPS). [8]

This shows at least some versions of NTRUEcnrypt has a security level comparable to other currently used cryptosystems such as the ones mentioned above.

6 Gentry's Somewhat Fully Homomorphic Encryption Scheme based on Lattices

In 2009, Gentry developed in his Phd thesis the first plausible fully Homomorphic Encryption Scheme based on lattice-based encryption. [4] Initially, a somewhat fully homomorphic encryption scheme is developed based on polynomial rings similar to those used in NTRU. Several tweaks are then introduced to the scheme and bootstrapping is applied to allow the scheme to become fully homomorphic. This section will focus on the initial somewhat fully homomorphic encryption scheme that Gentry proposed in chapter 5 and 7 in his thesis.

6.1 Polynomial Rings as Euclidean Space

Consider the polynomial ring $\frac{\mathbb{Z}[x]}{x^n-1}$ as described in the previous section. Note that this contains all the polynomials with degree less than or equal to $n-1$. In other words, the ring contains all polynomials of the form $\{a_0 + a_1x + a_2x^2 + \cdots + a_{n-1} | a_0 \dots a_{n-1} \in \mathbb{Z}\}$. Importantly, observe that each element in the ring can not only be viewed as a polynomial. Each element can also be viewed as vector in the Euclidean Space by viewing the coefficient within polynomial as entries in the vector. The polynomial $a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}$ within the space of $\frac{\mathbb{Z}[x]}{x^n-1}$ can also be viewed as the vector $(a_0, a_1, \dots a_{n-1})$ within $\mathbb{Z}^n$.

This gives us several interesting properties. By defining the polynomial ring and viewing the ring as an Euclidean space, the "+" operation can be viewed as simple vector addition. But the benefit is the convolution multiplication, which multiplies two polynomials within $\frac{\mathbb{Z}[x]}{x^n-1}$ and produces a product of these polynomials that is also in $\frac{\mathbb{Z}[x]}{x^n-1}$. Translated to the Euclidean Space $\mathbb{Z}^n$, the multiplication takes two n dimensional integer vectors and the product would also be a n dimensional integer vector. (This type of multiplication is not generally well-defined in a normal Euclidean space. The dot product, for example, produces a number instead of vector and the cross product is only applicable to $\mathbb{Z}^3$) In fact, the Euclidean space $\mathbb{Z}^n$, which is not intuitively an ring under the usual definition, now has the properties of a ring, permitting several additional operations with n-dimensional vectors, including "ideal lattices" as described as follows.

6.2 Ideal Lattices

Gentry's scheme made use of an important concept of "Ideal Lattices". An ideal is a mathematical structure within a ring informally defined as follows[3]:

Definition 2 *An ideal I within an ring R is a subset of R that satisfies the following properties:*

- $\forall a, b \in I, a + b \in I, ab \in I$
- $\forall a \in R, i \in I, ai \in I$

A simple example of an ideal within the ring of $\mathbb{Z}$ is the set of all numbers that are multiples of 3 (or n, where n is any positive integer). Note that the sum and product of two numbers that are multiples of three is also a multiple of 3. More importantly, the product of any integer (within the ring R, which in this case is the set of all integers) with a multiple of three is still a multiple of three regardless of whether the first element is with the ideal or not. This satisfies the first and second condition of the definition of an ideal respectively and therefore the set of all integers that are multiples of 3 is an ideal.

A lattice, under previous definitions, does not provide a multiplication between two vectors in the lattice. In order to allow the lattice to become an ideal within the ring of the $\mathbb{Z}^n$, there must be an appropriate multiplication between two vectors in a lattice that would produce a third element in the lattice. Fortunately, the polynomial multiplication defined in the Ring $\frac{\mathbb{Z}[x]}{x^N-1}$ converted to the corresponding vectors would provide a multiplication that satisfy the property of the multiplication that would allow a lattice to become an ideal. An lattice that also has the structure of an ideal is called an ideal lattice.

[3] https://mathworld.wolfram.com/Ideal.html

6.3 Key Generation

Similar to the NTRU [6] encryption scheme, Gentry's initial scheme operates on the polynomial ring $\frac{\mathbb{Z}[x]}{f(x)}$, where $f(x)$ is a polynomial of degree N. For simplicify, $X^N - 1$ will be used for $f(x)$ as in NTRU in this survey although other more complex function has been used to improve security.

In order to generate an instance of the encryption scheme, we need to generate two lattices I and J within and $\frac{\mathbb{Z}[x]}{f(x)}$ produce a "good" basis for I, B_I, and a good and bad basis pair for J, named B_J^{sk} and B_J^{pk} respectively. The lattice I and J must be relatively prime such that, for any vector v in the Euclidean space $\mathbb{Z}^N$, there must be a vector $i \in I$ and $j \in J$ such that $i + j = v$. The term "relatively prime" is similar to the definition of relatively prime on integers, as only relatively prime integers can linearly combine to form all integers. If both integers are multiples of 2, for example, then the corresponding linear combination cannot form any integer that is not a multiple of 2. The public key for the encryption scheme would be B_J^{pk} and B_I and the secret key would be B_J^{sk} and B_I.

6.4 Encryption

The plaintext for Gentry's S/F Homomorphic encryption scheme is a integer vector $m \in \mathbb{Z}^n$ within the fundamental region of B_I. Recall that the fundamental region consist of all the vectors within the parallelogram formed by the basis vectors. A vector in the fundamental region is usually not a vector in the lattice and therefore can be considered as a error vector. The first step of encryption is to pick a random vector in the Lattice I and add m to the lattice vector. This can be achieved by picking a random vector v in $\mathbb{Z}^n$ and computing $B_I v + m$. Note that $B_I v$ uses the entries in v as coefficients of basis vectors in B. Let $\psi' = B_I v + m$. There is an additional requirement that ψ' must be within the fundamental region of B_J^{sk}. While the encryption must not have access the exact value of B_J^{sk} because it is a secret key, this restriction can be achieved by having a restriction on the minimum size of the fundamental region of B_J^{sk} during key generation such that all vectors within a threshold would be within the fundamental region of B_J^{sk} regardless of the exact value of B_J^{sk}.

The ciphertext value for gentry's S/F scheme is computed as $\psi = \psi' \mod B_J^{pk} = (B_I v + m) \mod B_J^{pk}$.

The result of a mod operation of a vector with respect to a basis, $v \mod B$, is a vector v' in the fundamental region of B such that $v - v'$ is in the lattice produced by B. This can be interpreted as an extension of the mod operation within integers as $a \mod b$ can be interpreted as finding a integer $i < b$ such that $a - i$ is a multiple of b. In our case here, the mod operation is finding a vector v' within the fundamental region of B_j^{pk} such that $B_I v + m - v$ is within the lattice J. Note that a different basis always produce a different fundamental region even though they may produce the same lattice. Therefore, performing the mod operation on different basis of the same lattice with the same vector would produce different results.

6.5 Decryption

Given a encrypted ciphertext ψ and the secret key B_J^{sk} and B_I, the decryption algorithm works as follows: First, compute the value of $\psi' = \psi \mod B_J^{sk}$. Note that this value is always equal to ψ' during the encryption process. This is because $\psi = \psi' \mod B_J^{pk} = \psi' + v$ where $v \in J$ according to the definition of the mod operation, and also ψ' is within the fundamental region of B_J^{sk}. Then, compute $\psi' \mod B_I$, which would give back the value of m because $\psi' = B_I v + m$ and m is in the fundamental region of B_I.

Therefore, the decryption algorithm can be simply described as $m = (\psi \mod B_J^{sk}) \mod B_I$.

6.6 Homomorphic Operations

The defining feature of a S/F Homomorphic Encryption scheme is its ability to perform homomorphic operations (operations on the ciphertext that correspond to a specific operations on the plaintext without needing to decrypt). In this Gentry's S/F Homomorphic Encryption, the plaintext operations of interest are vector addition and multiplication within the fundamental region B_I (i.e. $\mod B_I$). Note that vectors in the scheme are also polynomials in the polynomial ring $\frac{\mathbb{Z}[x]}{f(x)}$, thus vector multiplication is equivalent to (convolution) polynomial multiplication as described in 5.1. The homomorphic operations equivalent to addition and multiplication of plaintext vectors are simply the corresponding operations on the ciphertext. Therefore, if two ciphertext vectors ψ_1 and ψ_2 correspond to plaintext vectors m_1 and m_2, then $\psi_1 + \psi_2$ (using simple vector addition) would decrypt to $m_1 + m_2$, and $\psi_1 \times \psi_2$ (using polynomial multiplication in the polynomial ring, i.e. convolution) would decrypt to $m_1 \times m_2$.

The following formula is a justification for the homomorphic evaluation of the addition operation

$$
\begin{aligned}
\mathit{Decrypt}(\psi_1 + \psi_2) &= \mathit{Decrypt}((B_I v_1 + m_1) \mod B_J^{pk} + (B_I v_2 + m_2) \mod B_J^{pk}) && (13)\\
&= (((B_I v_1 + m_1) \mod B_J^{pk} + (B_I v_2 + m_2) \mod B_J^{pk}) \mod B_J^{sk}) \mod B_I && (14)\\
&= (B_I v_1 + B_I v_2 + m_1 + m_2) \mod B_I && (15)\\
&= m_1 + m_2 \mod B_I && (16)
\end{aligned}
$$

Note that the equivalence between step (13) and (14) is only true when $\psi_1' + \psi_2' = B_I(v_1 + v_2) + (m_1 + m_2)$ is still within the fundamental region of B_J^{sk}. Therefore, the scheme is not yet fully homomorphic as there is not guarantee that $B_I(v_1 + v_2) + (m_1 + m_2)$ is still within the fundamental region of B_J^{sk} after multiple such homomorphic evaluations. A similar justification exist for homomorphic multiplication, but it is required that $(((B_I v_1 + m_1) \mod B_J^{pk} \times (B_I v_2 + m_2) \mod B_J^{pk}))$ is in B_J^{sk}. Note that even if we narrow down the possible range of $B_I v + m$ during so that multiple homomorphic evaluations can be performed while guaranteeing decryption correctness, the value of ψ' will always exceed the range of the fundamental region of B_J^{sk} after numerous homomorphic evaluations and therefore cause the decryption to fail. In some literature, the value of ψ', or whatever variable that would cause decryption to fail if exceeded a certain range, is referred to as *noise* of the S/F Homomorphic Encryption Scheme(e.g. [4]).

6.7 Example of Gentry's S/F Homomorphic encryption scheme

To put Gentry's S/F Homomorphic encryption scheme together, the scheme can be described as follows:

- **Secret Key**: $B_I \in \mathbb{Z}^{n\times n}$, good basis $B_J^{sk} \in \mathbb{Z}^{n\times n}$
- **Public Key**: $B_I \in \mathbb{Z}^{n\times n}$, bad basis $B_J^{pk} \in \mathbb{Z}^{n\times n}$
- **Plaintext**: $m \in F_{B_I}$, where F_{B_I} is the fundamental region of basis B_I. $F_{B_I} \subset \mathbb{Z}^n$
- **Encryption**: $\psi = \psi' \mod B_J^{pk} = (B_I v + m) \mod B_J^{pk}$, where $v \in \mathbb{Z}^n$ and $\psi' \in F_{B_J^{sk}}$
- **Decryption**: $m = (\psi \mod B_J^{sk}) \mod B_I$

For our example, we would use $n = 3$ as the lattice dimension. The basis matrices are as follows:

$$B_I = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix} B_J^{sk} = \begin{pmatrix} 21 & 0 & 0 \\ 0 & 23 & 0 \\ 0 & 0 & 99 \end{pmatrix} B_J^{pk} = \begin{pmatrix} 21 & 69 & -990 \\ 168 & 575 & -8514 \\ -231 & -644 & 8019 \end{pmatrix}$$

The plaintext we'll use for now is $m = (1, 0, 2)$. Note that m is within the fundamental region B_I, which includes all vectors with entries less than 4. For Encryption, we would need to generate a random vector $v = (1, 1, 0)$ such that $B_I v + m$ is within the fundamental region of B_J^{sk}. Then, $\psi' = B_I v + m = (5, 4, 2)$ and $\psi = \psi' \mod B_J^{pk} = (89, 326, -5047)$.

Now, suppose we have $\psi = (89, 326, -5047)$ and would like to decrypt the message. First compute $\psi' = (89, 326, -5047) \mod B_J^{sk} = (5, 4, 2)$. Note that this is exactly the same as ψ' during encryption. Then compute $m = \psi' \mod B_I = (5, 4, 2) \mod B_I = (1, 0, 2)$, which is the message that we originally encrypted.

In order to demonstrate homomorphic operations, we'll need another ciphertext to perform the add and muliply operations, suppose that the previously demonstrated plaintext and ciphertext are m_1 and ψ_1 respectively, and let $m_2 = (1, 1, 1)$. The encrypted value of m_2 is $(-16, 5, -692)$. Addition operation: $\psi_{sum} = \psi_1 + \psi_2 = (73, 331, -5739)$. Then we apply the decryption algorithm: $m_{sum} = (\psi_{sum} \mod B_J^{sk}) \mod B_I = (10, 9, 3) \mod B_I = (2, 1, 3)$, which is correctly equal to $m_1 + m_2$. For multiplication, a larger lattice dimension and larger basis entry is needed and is not suitable as an example in this report.

7 Further Step Proposed by Gentry to make the scheme Fully Homomorphic

The S/F Homomorphic Encryption scheme introduced in the previous section is an intermediate step in Gentry's Proposal of the first plausible Fully Homomorphic Encryption Scheme. To create a Fully Homomorphic Encryption Scheme from a S/F Homomorphic Encryption scheme, Gentry proposed the method of bootstrapping, which is still used in almost every fully homomorphic encryption scheme to date. The idea of bootstrapping is briefly introduced in [], but as we have described the initial S/F Homomorphic Encryption scheme by Gentry Bootstrapping may make more sense with this new knowledge.

7.1 The Recrypt Algorithm

In order for a ciphertext of a homomorphic encryption scheme to decryption correctly, the "noise" of the ciphertext ($\psi' = B_I v + m$ in the previous section) must be within a particular range. Repeated homomorphic evaluations will eventually exceed the range of allowed noise values. However, note that a plaintext value does not have any "noise", and thus the process of decryption removes all noise from the ciphertext as long as it is decrypted correctly. However, it is impossible to actually decrypt the ciphertext during homomorphic evaluations because homomorphic evaluators must not have access to the private key.

Therefore, Gentry introduces the *Recrypt* algorithm. Instead of having direct access to the secret key, the evaluator will have a copy of the encrypted version of the secret key. Note that it is not guaranteed that security will not be compromised if other entities have access of the encrypted version of the secret key, and encryption schemes that allow such access is called "KDM Secure" encryption schemes. Therefore, in order for Bootstrapping to work, the S/F Homomorphic Encryption Scheme must be KDM secure.

The *Recrypt* algorithm works as follows: Take a ciphertext with high noise, encrypt the ciphertext again using the encryption algorithm to obtain a doubly encrypted ciphertext. Then, use the encrypted secret key to execute the decryption circuit homomorphically. This will produce a ciphertext with the same plaintext value as the input ciphertext.

A graphical illustration of the *Recrypt* algorithm is as follows:

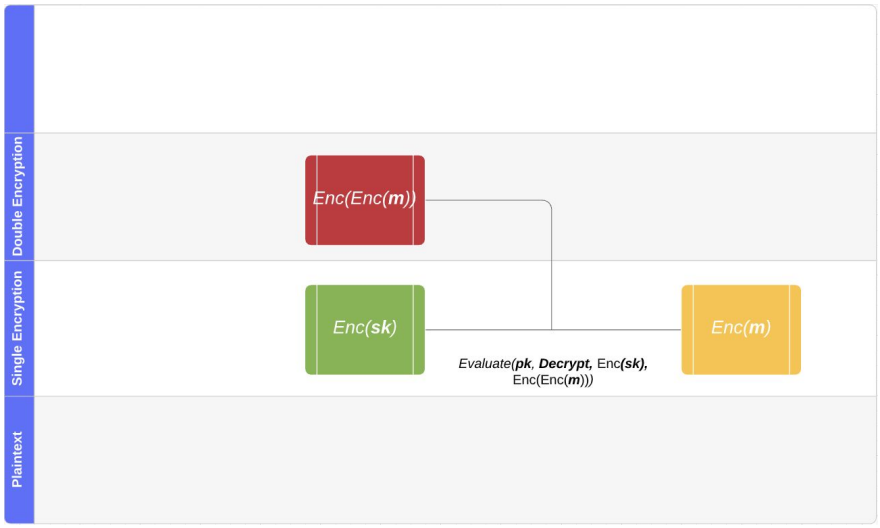

Figure 7: Recrypt algorithm illustration [14]

The *Recrypt* algorithm is basically a decryption of a doubly encrypted ciphertext, which removes all the noise (Here, noise can be viewed as the possibility of the noise exceeding the allowed range) within the original ciphertext. Therefore, the noise remaining in the recrypted ciphertext are those produced in the homomorphic evaluation process of the decryption circuit. Therefore, if the decryption circuit can be executed without the noise exceeding the allowed range and also produce

less noise than the original ciphertext, then the *Recrypt* algorithm effectively reduces the noise of the original ciphertext. Now, we introduce the definition of **bootstrappable**:

Definition 3 *A somewhat fully homomorphic encryption scheme is bootstrappable if it can homomorphically evaluate its own decryption circuit (and one extra homomorphic operation) without exceeding the noise parameter limit and is KDM secure.*

If a S/F Homomorphic Encryption Scheme is bootstrappable, then the following algorithm can be used to evaluate a circuit of any depth and thus transform the scheme into a fully homomorphic encryption scheme:

1. Start with original ciphertexts and encryption of the secretkey, $Enc(sk)$
2. Perform one (maybe more, depending on how many evaluations can be performed before homomorphically evaluating the decryption circuit) homomorphic operation required in the circuit.
3. Perform the *Recrypt* algorithm to reduce the noise of the ciphertexts.
4. Repeat steps 2 and 3 until the circuit is fully evaluated.

In order to construct a bootstrappable scheme, the scheme's decryption circuit depth must be reduced while maximizing the evaluation depth. However, the current S/F Homomorphic Encryption scheme as proposed by Gentry is not yet bootstrappable due to its large decryption circuit size. Therefore, Gentry would further introduce several tweaks of the scheme to decrease its decryption circuit depth and finally allow the scheme to become fully homomorphic. However, these tweaks would not be covered in this report.

8 Observations

This section consist of our current observations on the topics of homomorphic encryption and lattice-based cryptography. They have not been seriously verified and only serves as tentative information for future VIP students researching on the topic.

8.1 Low Efficiency

Currently, fully homomorphic encryption schemes are very inefficient in both time and space and thus lack known practical applications. In fact, the original fully homomorphic encryption scheme proposed by Gentry [4] requires 30 minutes per basic bit operation. Many generations of fully homomorphic encryption have been proposed, and one of the most popular implementation of Fully homomorphic encryption, the TFHE (Fast Fully Homomorphic Encryption Over the Torus) [3] requires 13 milliseconds single-core time to evaluate the bit operation and require 2KB for a single encrypted bit [14]. Despite several magnitude of improvement from the first Fully homomorphic encryption scheme, the homomorphic evaluation of TFHE is still too much slower than plaintext operations and require too much space to be suitable for practical use.

Several reasons may be attributed to this lack of efficiency. Firstly, Lattice-based encryption schemes, used by many homomorphic encryption schemes, generally require a good and bad basis

pair of the same lattice to be effective. However, algorithms such as the LLL lattice basis reduction algorithm [10] are effective at obtaining a good basis from a bad one, and a lattice dimension of 400 is generally required to prevent feasible attack by simply obtaining the good basis using LLL. Each vector is usually used to only encrypt a single bit to guarantee semantic security [4] (security even when a very small range of plaintext is possible). This results in very large ciphertext size and inefficient computation of homomorphic bit operations. Moreover, a large lattice dimention is also requires to increase the evaluation depth so that the decryption circuit can fit in the range of allowed circuits.

Another important reason for inefficiency lies in the nature of bootstrapping. Note that the *Recrypt* algorithm is performed after only very few homomorphic operations of the actual circuit being evaluated. Despite the fact that the depth of decryption circuit is minimized, homomorphic decryption still takes much more time than the allowed homomorphic evaluations of the actually evaluated circuit. Therefore, a large portion of the computation during homomorphic evaluations is spent on the *Recrypt* algorithm, causing fully homomorphic evaluations to be much slower than their S/F Fully homomorphic counterparts.

8.2 Homomorphism in NTRU

Recent advances in research about homomorphic encryption have produced many different fully homomorphic encryption schemes based upon different types of S/F Homomorphic schemes (bootstrapping is used to convert them to fully homomorphic ones). While we generally introduced NTRU as a example lattice-based encryption scheme, recent research have [2] already turned a enhanced version of NTRU into a fully homomorphic scheme. In fact, another research [16] claims that "the NTRU utilizes low memory use and generates key at a rapid rate. One main advantage of the NTRU over the DGHV is slower growth of noise than DGHV", which would theoretically cause more efficient fully homomorphic evaluations due to less frequent and more efficient bootstrapping.

9 Future Work

Several topics are available for future VIP researchers to investigate further into the topic. Firstly, we have not yet able to create, either theoretical or implementations level, a homomorphic algorithm that uses S/FHE or PHE to solve the Obtention Scenario [14] (In short, the scenario ask to homomorphically evaluate whether $|a - b| > d$). In particular, the comparison operations is on main focus/concern as most homomorphic encryption schemes only considers addition and multiplication. In the previous VIP semester, comparison and also operations including addition and multiplication is implemented using homomorphic bit-wise operations, which is inefficient compared to the simplicity of the original formula. Therefore, an efficient homomorphic comparison may be a key to solving the Obtension Scenario.

Another direction is to investigate other lattice-based cryptographic schemes. Lattice-based cryptography, on its own, is a very robust candidate for post-quantum cryptography. No know algorithms, including quantum ones, have been discovered to efficiently solve lattice problems. Also, as we mentioned in section 8, more lattice-based encryption schemes [5] have been discovered to have

[4] For better definition see https://doi.org/10.1007/978-1-4419-5906-5_23

homomorphic properties and are potential candidates for efficient fully homomorphic encryption. Therefore, further understanding lattice-based encryption schemes is a great direction for better understanding of homomorphic encrytion.

Recent advances in homomorphic encryption have also created many different schemes without the use of lattices (TFHE [3] for example). Many of these scheme have significant improvements in efficiency in both space and time. We believe that the following papers are great potential candidates for investigation into this direction for their simplicity in concept and easy of understanding:

1. Fully Homomorphic Encryption Over the Integers [18]
2. Leveled Fully Homomorphic Encryption Without Bootstrapping [1]

These suggestions of directions are only tentative and future VIP students may investigate other Homomorphic Encryption Schemes as well.

References

[1] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory*, 6(3), jul 2014.

[2] X Che, T Zhou, N Li, H Zhou, Z Chen, and X Yang. Modified multi-key fully homomorphic encryption based on NTRU cryptosystem without key-switching. *Tsinghua Science and Technology*, 25(5):564–578, oct 2020.

[3] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast Fully Homomorphic Encryption Library, aug 2016.

[4] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[5] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. pages 469–477, 06 2015.

[6] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. "NTRU: A ring-based public key cryptosystem". In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[7] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. *An Introduction to Mathematical Cryptography*. Springer Science+Business Media, 2014.

[8] Burt Kaliski. *MIPS-Year*, page 383. Springer US, Boston, MA, 2005.

[9] Subhash Khot. Hardness of Approximating the Shortest Vector Problem in Lattices. *J. ACM*, 52(5):789–808, sep 2005.

[10] A K Lenstra, H.W.jun. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[11] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[12] Peyan Pan and Dr. Vincent Mooney. Lattice-Based Cryptography in Homomorphic Encryption. Technical report, Georgia Institute of Technology, Atlanta, 2019.

[13] Jill Pipher. Mathematical Ideas in Lattice Based Cryptography, may 2018. https://youtu.be/msPrqDwLhi8.

[14] Vignesh Raman and Dr. Vincent Mooney. Security for Public Infrastructure Security for Public Infrastructure using Homomorphic Encryption. *Georgia Institute of Technology*, 2019.

[15] Oded Regev. Lecture 1: Introduction. Lattices in Computer Science, 2004. New York University.

[16] B Santhiya and K Anitha Kumari. Analysis on DGHV and NTRU Fully Homomorphic Encryption Schemes. In L Ashok Kumar, L S Jayashree, and R Manimegalai, editors, *Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications*, pages 669–678, Cham, 2020. Springer International Publishing.

[17] Joseph H Silverman. NTRU Cyrptosystems Technical Report. Technical report, mar 1999. NTRU.

[18] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.