Project Code: WJM5200

An Exposé on the López-Alt-Tromer-Vaikuntanathan Fully Homomorphic Encryption Scheme

A Major Qualifying Project
submitted to the Faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree of Bachelor of Science in Mathematical Science

by

Julie Anne Wade

Approved:

Dr William Martin, Project Advisor

Dr Berk Sunar, Project Advisor

Abstract

Around every corner, more and more companies and organizations are utilizing the cloud to store and perform expensive computations remotely; all the more noticing the numerous advantages in costs and functionality. However, it is most commonly the case that data stored on the cloud is not safe from snooping and while much of what is stored is riddled with sensitive information this raises grave questions of confidentiality. In order to remove this vulnerability, users must encrypt their data before sending it to the cloud, thus losing the functionality of performing computations. Due to recent advances in fully homomorphic encryption (FHE) it is found to be possible to perform arbitrary computations on this encrypted data, thus enabling the prospect of personal computers as trusted but weak interfaces to the powerful but untrusted cloud on which the bulk of computing can be performed.

Proposed at the Symposium on Theory of Computing (STOC) in 2012, Adriana López-Alt, Eran Tromer and Vinod Vaikuntanathan developed a new notion of secure multiparty computation aided by a computationally-powerful but untrusted cloud server. All users input data and intermediate results are protected from snooping by the cloud, as well by other users. The construction of the López-Alt-Tromer-Vaikuntanathan (LTV) scheme is based on the NTRU encryption scheme originally proposed by Hoffstein, Pipher and Silverman; more precisely, based off the slightly modified version due to Stehlé and Steinfield. NTRU is one of the earliest lattice-based public-key encryption schemes and threatens to replace RSA and elliptic curve cryptosystems in applications where computational efficiency is at a premium.

We present a careful rewrite of the FHE LTV scheme which specializes to the traditional single-key approach. Simplifying notation and filling in gaps, this rewrite will make the information portrayed in the LTV scheme accessible to non-experts. As this construction was one of the authors main contributions, we believe it to be of independent interest. Particularly since the NTRU scheme, when compared to other public key cryptosystems at roughly equivalent levels of security, offers more efficient encryption and decryption as well as much faster key generation. Although some of the efficiency of NTRU was lost during the transformation to a fully homomorphic system, we believe that this system is still a leading candidate for a practical FHE scheme.

Contents

1 Introduction

In a digital era in which we store our data and perform our expensive computations remotely, on powerful servers, commonly referred to as the "cloud," we notice numerous advantages in costs and functionality. However, the cloud raises grave questions of confidentiality since the data stored on its servers is often riddled with sensitive information. As this data could be vulnerable to snooping by the cloud provider or even by other cloud clients, it is necessary for the users to encrypt their data before storing it on the cloud. Due to recent advances in fully homomorphic encryption (FHE) it is possible to perform arbitrary computations on this encrypted data, thus enabling the prospect of personal computers and mobile devices as trusted but weak interfaces to powerful but untrusted cloud on which the bulk of computing is performed.

Traditional FHE schemes are single-key in the sense that they can perform arbitrarily complex computations on inputs encrypted under the same key. However, there are scenarios where users, who have uploaded their data stores to the cloud in encrypted form, decide to compute some joint function of their collective data. This multiparty scenario is significantly more complex, and comes with a set of natural, yet strict, requirements.

Proposed at the Symposium on Theory of Computing (STOC) in 2012, Adriana López-Alt, Eran Tromer and Vinod Vaikuntanathan developed a new notion of secure multiparty computation aided by a computationally-powerful but untrusted cloud server. In the notion that they call on-the-fly multiparty computation (MPC), the cloud can non-interactively perform arbitrary, dynamically chosen computations on data belonging to arbitrary sets of users chosen on-the-fly. All users' input data and intermediate results are protected from snooping by the cloud, as well by other users. This extends the standard notion of fully homomorphic encryption, where users can only enlist the cloud's help in evaluating functions on their own encrypted data.

In the López-Alt-Tromer-Vaikuntanathan (LTV) scheme, we assume the parties do not trust each other and so they will most certainly not want to encrypt their inputs using each others keys. Nevertheless, Gentry proposed a way of using single-key FHE schemes in order to do multiparty computation utilizing a (short) MPC protocol to compute a joint public key and a matching secret key which is secret-shared among the parties. Asharov, Jain, López-Alt, Tromer, Vaikuntanathan and Wichs extended Gentry's scheme in recent work making it efficient in terms of the concrete round, communication and computational complexity. [LATV12]

However, this line of work does not address the dynamic and non-interactive nature of an on-the-fly MPC. In particular, once a subset of parties and a function are chosen, the protocols of Gentry and Asharov et al require the parties to be online and run an interactive MPC protocol to generate a joint public key. Thus, even the feasibility of on-the-fly MPC is not

addressed by previously existing techniques.

In the LTV scheme, each user is involved only when initially uploading his (encrypted) data to the cloud, and in a final output decryption phase when outputs are revealed; the complexity of both is independent to the function being computed and the total number of users in the system. In addition, when users upload their data, they need not decide in advance which function will be computed, nor who they will compute with; they need only retroactively approve the eventually-chosen functions and on whose data the functions were evaluated. [LATV12]

We present a rewrite of the FHE LTV scheme which utilizes the traditional single-key approach. Our construction, like that of the LTV scheme, is based on the NTRU encryption scheme, originally proposed by Hoffstein, Pipher and Silverman; more precisely, the slightly modified version due to Stehlé and Steinfield. NTRU is one of the earliest lattice-based public-key encryption schemes, together with the Atjai-Dwork cryptosystem and the Goldreich-Goldwasser-Halevi cryptosystem. We observe that NTRU can be make single-key fully homomorphic, using relinearization and modulus reduction to create a bootstrappable scheme.

This construction is one of the authors' main contributions and we believe it to be of independent interest. Our construction is particularly interesting since the NTRU scheme was originally proposed as an efficient public-key encryption scheme, meant to replace RSA and elliptic curve cryptosystems in applications where computational efficiency is at a premium. [HPS98] Although the transform to a fully homomorphic system deteriorates the efficiency of NTRU slightly, we believe that this system is a leading candidate for a practical FHE scheme. What's more, as we show, the scheme supports homomorphic operations utilizing modulus reduction to create a bootstrappable scheme.

1.1 Traditional Single-Key Approach to LTV

In our paper, we present a simplification of the LTV scheme to the traditional single-key approach. Similar to the original LTV scheme, we based our scheme off of the NTRU encryption scheme. We observe that NTRU can be made single-key fully homomorphic using the recent techniques of Stehlé and Steinfield. [SS11]

Simplifying notation and filling in gaps, this rewrite will make the information portrayed in the LTV scheme accessible to non-experts. As this construction was one of the authors' main contributions, we believe it to be of independent interest. Particularly since the NTRU scheme, when compared to other public key cryptosystems at roughly equivalent levels of security, offers more efficient encryption and decryption as well as much faster key generation. Although some of the efficiency of NTRU was lost during the transformation to a fully

homomorphic system, we believe that this system is still a leading candidate for a practical FHE scheme. [LATV12]

1.2 Organization

In Section 3 we review a broad range of mathematical topics relevant to the rest of the paper as well as describe the NTRU encryption scheme developed by Hoffstein, Pipher and Silverman [HPS98]. Then in Section 4, we formally present our scheme, showing how to instantiate somewhat homomorphic encryption from the NTRU encryption scheme. Also in Section 4 we show how to achieve full homomorphism, using relinearization and demonstrating that the scheme can be made bootstrappable by modulus reduction. Lastly, we present our conclusions in Section 5.

2 Mathematical Preliminaries

This section reviews a broad range of mathematical topics relevant to the rest of the paper. We begin with a brief discussion of algebra, paying special attention to modular arithmetic and the Euclidean algorithm, followed by a brief review of ring theory. In addition, we will introduce a brief summary of circuits and their importance in cryptography.

2.1 Algebra

In this section we will discuss briefly the necessary algebra background, more specifically modular arithmetic, Euclid's algorithm and ring theory, required for our scheme which we will describe in Section 4.

2.1.1 Modular Arithmetic

Modular arithmetic is an abstraction of a method of counting that we often use. For example, if it is now September, what month will it be 25 months from now? Of course, the answer is October, but the interesting fact is that you didnt arrive at the answer by starting with September and counting off 25 months. Instead, without even thinking about it, you simply observed that $25 = 2 \times 12 + 1$, and you added 1 month to September. Surprisingly, this simple idea has numerous important applications in mathematics and computer science. You will see a few of them in this paper. The following notation is convenient. [Gal94]

When $a = qn + r$, where q is the quotient and r is the remainder upon dividing a by n, we write $a \mod n = r$. Thus,

$$\begin{array}{rcl} 3 \pmod 2 = 1 & \text{since} & 3 = 1 \times 2 + 1, \\ 6 \pmod 2 = 0 & \text{since} & 6 = 3 \times 2 + 0, \\ 11 \pmod 3 = 2 & \text{since} & 11 = 3 \times 3 + 2, \\ 62 \pmod{85} = 62 & \text{since} & 62 = 0 \times 85 + 62, \\ -2 \pmod{15} = 13 & \text{since} & -2 = -1 \times 15 + 13. \end{array}$$

In general, if a and b are integers and n is a positive integer, then $a \pmod n = b \pmod n$ if and only if n divides $a - b$.

In our applications we will use addition and multiplication modulo n. When you wish to compute $ab \mod n$ or $(a + b) \mod n$, and a or b is greater than n, it is easier to mod

first. For example, to compute $(25 \times 36) \mod 11$, we note that $27 \mod 11 = 5$ and $36 \mod 11 = 3$, so $(27 \times 36) \mod 11 = (5 \times 3) \mod 11 = 4$.

2.1.2 Euclidean Algorithm

The Euclidean algorithm is an efficient method for computing the greatest common divisor of two positive integers.

Definition 2.1 (Euclidean algorithm)**.** Let $a, b \in \mathbb{Z}$ with $b > 0$. If $b \mid a$, then $\gcd(a, b) = d$; otherwise there are integers $q_1, r_1, q_2, r_2, ..., q_n, r_n, q_{n+1}$ such that:

$$\begin{aligned}
a &= bq_1 + r_1, \quad 0 < r_1 \leq (b-1), \\
b &= r_1q_2 + r_2 \quad 0 < r_2 \leq (r_1 - 1), \\
r_1 &= r_2q_3 + r_3, \quad 0 < r_3 \leq (r_2 - 1), \\
&\vdots \\
r_{n-3} &= r_{n-2}q_{n-1}, \quad 0 < r_{n-1} \leq (r_{n-2} - 1), \\
r_{n-2} &= r_{n-1}q_n + r_n, \quad 0 < r_n \leq (r_{n-1} - 1), \\
r_{n-1} &= r_nq_{n+1},
\end{aligned}$$

and $\gcd(a, b) = r_n$.

Extended Euclidean Algorithm

By reversing the steps in the Euclidean Algorithm previously discussed, it is possible to find the inverse of a number modulo n. As we carry out each step of the Euclidean algorithm, we will also calcuate an auxiallry number, $p_i = p_{i-2} - p_{i-2}q_{i-2} \pmod n$. Continuing one step beyond the last step of the Euclidean algorithm. If the last non-zero remainder occurs at step $n - 2$ and $r_{n-2} = 1$ then the inverse of p_n is the inverse of $a \pmod n$. However, if a remainder of 1 is never reached, then a does not have an inverse modulo n.

$$\begin{aligned}
n &= aq_1 + r_1, \quad p_0 = 0, \\
a &= r_1q_2 + r_2 \quad p_1 = 1, \\
r_1 &= r_2q_3 + r_3, \quad p_2 = p_0 - p_1q_0 \pmod n, \\
&\vdots \\
r_{n-3} &= r_{n-2}q_{n-1}, \quad p_{n-2} = p_{n-4} - p_{n-3}q_{n-4} \pmod n, \\
r_{n-2} &= r_{n-1}q_n + r_n, \quad p_{n-1} = p_{n-3} - p_{n-2}q_{n-3} \pmod n, \\
r_{n-1} &= r_nq_{n+1}, \quad p_n = p_{n-2} - p_{n-1}q_{n-2} \pmod n,
\end{aligned}$$

Concluding with, the inverse of a as follows

$$p_n = p_{n-1} - p_nq_{n-1} \pmod n.$$

2.1.3 Rings

Many sets are naturally endowed with two binary operations: addition and multiplication. Examples that quickly come to mind are the integers, the integers modulo n, the real numbers, matrices, and polynomials. When considering these sets as groups, we simply used addition and ignored multiplication. In many instances, however, one wishes to take into account both addition and multiplication. One abstract concept that does this is the concept of a ring. [Gal94]

Definition 2.2 (Ring)**.** A ring R is a set with two binary operations, addition (denoted by $a + b$) and multiplication (denoted by ab), such that for all a, b, c in R:

1. $a + b = b + a$
2. $(a + b) + c = a + (b + c)$
3. There is an additive identity 0. That is, there is an element 0 in such that $a + 0 = a$ for all $a$in R.
4. There is an element a in R such that $a + (-a) = 0$.
5. $a(bc) = (ab)c$
6. $a(b + c) = ab + ac$ and $(b + c)a = ba + ca$

So, a ring is an Abelian group under addition, also having an associative multiplication that is left and right distributive over addition. Note that multiplication need not be commutative. When it is, we say that the ring is commutative. Also, a ring need not have an identity under multiplication.

The following terminology and notation are convenient. If a and b belong to a commutative ring R and a is nonzero, we say that a divides b (or that a is a factor of b) and write $a \mid b$, if there exists an element c in R such that $b = ac$. If a does not divide b, we write $a \nmid b$.

Recall that if a is an element from a group under the operation of addition and n is a positive integer, na means $a + a + ... + a$, where there are n summands. When dealing with rings, this notation can cause confusion, since we also use juxtaposition for the ring multiplication. When there is a potential for confusion, we will use $n \cdot a$ to mean $a+a+...+a$ (n summands).[Gal94]

2.2 Circuits

Definition 2.3. A **circuit** is a directed acyclic graph (DAG) of straight-line programs. Where

- *vertices* are associated with program steps and
- *edges* identify dependencies between steps.

In this paper we will focus on **logic circuits**, where all operations are Boolean.

2.2.1 Boolean Operators

Boolean operators are simple words (AND, OR, NOT or XOR) used as conjunctions to combine logical expressions. Utilizing truth tables, we can describe the functions of each of these statements concisely.

For our truth tables we will use the

AND

From the truth table below we can see that the Boolean algebra between p and q is multiplication, represented as $p \cdot q$.

Table 1: AND Truth Table

p	q	p AND q
1	1	1
1	0	0
0	1	0
0	0	0

OR

From the truth table below we can see that the Boolean algebra between p and q is addition, represented as $p + q$.

Table 2: OR Truth Table

p	q	p OR q
1	1	1
1	0	1
0	1	1
0	0	0

NOT

From the truth table below we can see that the Boolean algebra between NOT and p is often written $\bar{p}$ or the converse of p. We can compute $\bar{p}$ as $1 + p \pmod 2$.

Table 3: NOT Truth Table

p	q	NOT p	NOT q
1	1	0	0
1	0	0	1
0	1	1	0
0	0	1	1

XOR

XOR, or as exclusive or, and is thought of as "p or q but not both. From the truth table below we can see that the Boolean algebra between p and q is subtraction, represented by $p - q \pmod 2$.

Table 4: XOR Truth Table

p	q	p XOR q
1	1	0
1	0	1
0	1	1
0	0	0

Using these four Boolean operators we can create any necessary algebraic expression possible, including NAND, NOR and XNOR.

3 A Brief History of Homomorphic Encryption

This section presents a brief history of homomorphic encryption. We begin by describing what homomorphic encryption is and the specialized types of homomorphic encryption which exist. In addition, we will take a look at the RSA encyrtpion scheme developed by Rivest, Shamir and Adleman [RSA83], which is the most commonly used encryption scheme in practice today. WE will then briefly discuss Gentry's new notions creating the first fully homomorphic scheme. Lastly, we will describe the NTRU encryption scheme developed by Hoffstein, Pipher and Silverman [HPS98] of which our scheme is based.

3.1 Definitions

In order to fully understand this paper, we must first understand the concept of homomorphic encryption. Breaking the term down, we can first define the term *homomorphic* from it the algebraic term homomorphism.
Definition 3.1 (Group Homomorophism, See [Gal94])**.** If $(G, *)$ and $(H, \circ)$ are groups, then there exists a mapping $\varphi : G \to H$ that is a **homomorphism** if

$$\varphi(a * b) = \varphi(a) \circ \varphi(b)$$

for all $x, y \in G$ and $\varphi(a), \varphi(b) \in H$.

We can extend this notion to rings as follows.
Definition 3.2 (Ring Homomorphism, See [Gal94])**.** If R and S are two rings, then there exists a mapping $\varphi : R \to S$ that is a **homomorphism** if

1. $\varphi(a + b) = \varphi(a) + \varphi(b)$ and
2. $\varphi(ab) = \varphi(a)\varphi(b)$.

Thus preserving the ring operations of addition and multiplication.

We must now define the latter part of the term, *encryption*, as follows.
Definition 3.3 (Encryption Scheme)**.** If $\mathcal{E}$ is an encryption scheme with a ring of plaintexts, $(\mathcal{P}, +, \cdot)$, and a ring of ciphertexts $(\mathcal{C}, \oplus, \otimes)$ then we have the following two mappings.

1. $\mathsf{Enc} : \mathcal{P} \to \mathcal{C}$
2. $\mathsf{Dec} : \mathcal{C} \to \mathcal{P}$

Applying the above definition to encryption schemes, we have the following definition.

Definition 3.4 (Homomorphic Encryption Scheme)**.** If $\mathcal{E}$ is an encryption scheme with and an operation $\star$, then $\mathcal{E}$ is **homomorphic** with respect to $\star$ on $\mathcal{P}$ if and only if

$$\mathsf{Dec}(\mathsf{Enc}(m_1) * \mathsf{Enc}(m_2)) = \mathsf{Dec}(\mathsf{Enc}(m_1 \star m_2)) = m_1 \star m_2$$

for some operation $*$ on $\mathcal{C}$.

For the context of our paper, and what is generally accepted in the cryptographic community, we will consider a scheme *somewhat homomorphic* if it can properly evaluate on a circuit of limited depth, yet no further due to noise introduced by the operations performed. In addition we will consider a scheme to be *additively homomorphic* if it can perform an unlimited number of operations, regardless of the depth of the circuit, on the ciphertexts that correspond exactly to an unlimited number of additions with the plaintexts. We can easily see that we could manipulate the previous definition by replacing addition with multiplication to describe a scheme which we will consider to be *multiplicatively homomorphic*. Finally, we will consider a scheme to be *fully homomorphic* if it can perform an unlimited number of both additive and multiplicative operations.

3.2 RSA

Developed in 1978, Ron Rivest, Adi Shamir and Leonard Adleman introduced a public key encryption scheme called RSA, which is currently the most commonly used encryption algorithm utilized today. The RSA scheme is based on the difficultly to factor large numbers and works as such. [RSA83]

3.2.1 Key Creation

Bob begins by chooses two large prime numbers p and q and keeps them secret. He will then multiply the two keys creating $N = pq$.
Note. Assumbing Bob choose p and q large enough, while it is relatively easy for Bob to compute N, it is basically impossible for us, or anyone else, to find p and q as we would need to factor N. For example, suppose N is 400 digits long. As most computers use a blunt force method of checking of something that is the order of the size of the square-root of the number to be factored; thus in this case it would be necessary to check a 200 digit number. Now, the lifetime of the universe is approximately 10^{18} second. Let's assume that a computer could test one million factorizations per second, in the lifetime of the universe that would equate to checking 10^{24} possibilities. However, for our 400 digit product recall that there are 10^{200} possibilities to check, an impossibility using today's computing abilities.

Bob must also calculate $t = (p-q)(q-1)$ and choose a prime number e such that e is

relatively prime to t. In other words, $e \nmid t$. Lastly, Bob will need to compute $d \cdot e = 1 \pmod{t}$. Bob will define his public and secret keys

$$pk := (N, e) \text{ and } sk := (N, d),$$

respectively.

3.2.2 Encryption

Alice can now encrypt a plaintext message m for Bob utilizing his public key as follows.

$$c = m^e \pmod{N}$$

She will then send the encrypted message c to Bob.

3.2.3 Decryption

Utilizing his private key, Bob can decrypt the message sent to him by Alice using the following.

$$m = c^d \pmod{N}$$

3.2.4 Why Decryption Works

We can see that decryption works, since

$$\begin{aligned} c^d &= (m^e)^d \mod N \\ &= (m^e)^{e^{-1}} \mod N \\ &= (m^{e \cdot e^{-1}}) \mod N \\ &= m \mod N. \end{aligned}$$

3.2.5 Homomorphic Properties

The RSA encryption scheme is know to be multiplicatively homomorphic, which we can show below. Given two ciphertexts, c_1 and c_2 as defined,

$$c_1 := m_2^e \pmod{N} \text{ and } c_2 := m_1^e \pmod{N}$$

We can compute,

$$\begin{aligned} c_1 \cdot c_2 &= m_1^e \cdot m_2^e \pmod{N} \\ &= (m_1 \cdot m_2)^e \pmod{N} \\ &= \mathsf{Enc}(m_1 \cdot m_2). \end{aligned}$$

Yielding an encryption of the product of our original plaintexts, showing that the RSA scheme is indeed multiplicately homomorphic. Now, let's look to see if the scheme is also additively homomorphic. We can compute,

$$\begin{aligned} c_1 + c_2 &= m_1^e + m_2^e \pmod{N} \\ &\neq (m_1 + m_2)^e \pmod{N} \\ &\neq \mathsf{Enc}(m_1 + m_2). \end{aligned}$$

Therefore, we can easily see the scheme is not additively homomorphic.

3.3 Gentry's New Notion of Bootstrapping

Proposed in 2008, Craig Gentry, Ph.D. candidate at Stanford University proposed a new scheme that was first to be fully homomorphic. Gentry suggested an alternative approach to fully homomorphic encryption, where instead of relaying on the structure of the encryption scheme he suggested to periodically refresh the ciphertext. His new idea allowed for scheme with a limited number of operations to be promoted to one that allows an unlimited number. [Gen09a] We will discuss this notion later on in Section 4.5.1.

However, this result is not so simple, requiring the use of a key that grows substantially in length as the number of operations to be evaluated increases and creating ciphertexts that are in the order of 10^6 times larger than their corresponding plaintext counterparts.

We refer the reader to Rebecca Meissen's, a graduate of Worcester Polytechnic Institute, paper for an in depth discussion of Gentry's work. [Mei12]

3.4 NTRU: A Public Key Cryptosystem

NTRU is a public key cryptosystem developed by Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman. Featuring reasonably short, easily created keys, high speed and low memory requirements, the NTRU scheme was originally proposed as an efficient public-key encryption scheme, meant to replace RSA and elliptic curve cryptosystems in applications where computational efficiency is at a premium. The encryption and decryption use a mixing system suggested by polynomial algebra combined with a clustering principle based on elementary probability theory. [HPS98]

3.4.1 Preliminaries for the NTRU Instantiation

The following are all part of the domain parameters for an implementation of NTRU.

- n The dimension of the polynomial ring used in NTRU. Note: The polynomials will have degree $n-1$.
- p A positive integer specifying a ring $\mathbb{Z}/p\mathbb{Z}$ over which the coefficients of a certain product of polynomials will be reduced during the encryption and encryption processes.
- q A positive integer specifying a ring $\mathbb{Z}/q\mathbb{Z}$ over which the coefficients of a certain product of polynomials will be reduced during the encryption and decryption processes; also used in the construction of the public key.

We will also use the following notation.

- f A polynomial in $\mathbb{Z}[X]/\langle x^n-1\rangle$.
- f_p A polynomial in $\mathbb{Z}[X]/\langle p, X^n-1\rangle$ (this is part of the private key). This polynomial is obtained by reducing the coefficients of f modulo p.
- f_q A polynomial in $\mathbb{Z}[X]/\langle q, X^n-1\rangle$. this polynomial is obtained by reducing the coefficients of f modulo q.
- $\mathcal{L}_f$ The set of polynomials in $\mathbb{Z}[X]/\langle x^n-1\rangle$ who coefficients satisfy d_f.
- g A polynomial in $\mathbb{Z}[X]/\langle q, X^n-1\rangle$ (used with f_q to construct the public key).
- $\mathcal{L}_g$ The set of polynomials in $\mathbb{Z}[X]/\langle x^n-1\rangle$ whose coefficients satisfy d_g.
- $\mathcal{L}_r$ The set of polynomials in $\mathbb{Z}[X]/\langle x^n-1\rangle$ whose coefficients satisfy d_r.
- f_p^{-1} The inverse of f_p in $\mathbb{Z}[X]/\langle p, X^n-1\rangle$.
- f_q^{-1} The inverse of f_q in $\mathbb{Z}[X]/\langle q, X^n-1\rangle$.
- h The public key, a polynomial in $\mathbb{Z}[X]/\langle q, X^n-1\rangle$.
- r A polynomial in $\mathbb{Z}[X]/\langle q, X^n-1\rangle$ (used with h to encode a message).
- m The plaintext message, a polynomial in $\mathbb{Z}[X]/\langle p, X^n-1\rangle$.
- c The encrypted message, a polynomial in $\mathbb{Z}[X]/\langle q, X^n-1\rangle$.

Throughout the NTRU scheme, we work in the ring $R = \mathbb{Z}[X]/\langle x^n - 1\rangle$. An element $f \in R$ will be written as a polynomial of a vector,

$$f = \sum_{i=0}^{n-1} f_i x^i = [f_0, f_1, ..., f_{n-1}].$$

We write $\circledast$ to denote multiplication in R. This *star multiplication* is given explicitly as a cyclic convolution product,

$$f \circledast g = h \quad with \quad h_k = \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{n-1} f_i g_{n+k-1} = \sum_{i+j \equiv k \mod n} f_i g_j.$$

When we do a multiplication modulo (say) q, we mean to reduce the coefficients modulo q, so the result lies in $\mathbb{Z}[X]/\langle x^n - 1\rangle$.

3.4.2 Parameters

More specificaly, the NTRU cryptosystem depends mainly on three integer parameters (N, p, q) satisfying the following conditions,

(i) p and q are relatively prime and

(ii) q will always be considerably larger than p.

In addition, it depends on four sets $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_\phi$, $\mathcal{L}_m$ of polynomials of degree $N-1$ with integer coefficients. This cryptosystem works in the ring $R = Z[X]/\langle X^N - 1\rangle$.

3.4.3 Key Creation

To create an NTRU key, Bob randomly chooses 2 polynomials f,$g \in \mathcal{L}_g$, such that the polynomial f has inverses modulo q and modulo p. We will denotes these inverses as follows,

$$F_q \circledast f = 1 \pmod{q} \tag{1}$$

and

$$F_p \circledast f = 1 \pmod{p}. \tag{2}$$

This will be true for most choices of f when using suitable parameter choices and the actual computation of these inverses is easy using a modification of the Euclidean algorithm. The polynomial f is Bob's private key, however in practice he will also want to store F_p.

Bob can then compute the public key, the polynomial h as defined as

$$h = F_q \circledast g \pmod{q}. \tag{3}$$

3.4.4 Encryption

Suppose that Alice, the encrypter, wants to send a message to Bob, the decrypter. She begins by selecting a one-bit message m from the set of plaintexts $\mathcal{L}_m$. Next she randomly chooses a polynomial $s \in \mathcal{L}_\phi$ and use the previously published public key h to compute

$$c = ps \circledast h + m \pmod{q}. \tag{4}$$

This is the encrypted message which Alice will send.

3.4.5 Decryption

Suppose Bob has now received the transmitted encrypted message e from Alice and wants to decrypt it using his private key f. In order to decrypt c, Bob now computes

$$a = f \circledast c \pmod{q}, \tag{5}$$

where he chooses the coefficients of a such that they lie in the interval $(-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ in order to avoid wrap-around errors. Now treating a as a polynomial with integer coefficients, Bob recovers the message by computing

$$m = F_p \circledast a \pmod{p}. \tag{6}$$

3.4.6 Why Decryption Works

Utilizing equation 5, we recall

$$a = f \circledast e \pmod{q}.$$

Substituting for e using equation (4), we find

$$a = f \circledast p\phi \circledast h + f \circledast m \pmod{q}.$$

Substituting for h using equation (3), we find

$$a = f \circledast p\phi \circledast F_q \circledast g + f \circledast m \pmod{q}.$$

Rearranging, we have

$$a = (f \circledast F_q) \circledast p\phi \circledast g + f \circledast m \pmod{q}.$$

Using equation (1), we can then simply the expression and have

$$a = p\phi \circledast g + f \circledast m \pmod{q}.$$

Since $a \in R$ we then reduce modulo p and find

$$a = f \circledast m \pmod{p}.$$

Multiplying both sides of the equation on the left by F_p, we have

$$F_p \circledast a = F_p \circledast f \circledast m \pmod{p}.$$

Finally, using equation (2) we have

$$F_p \circledast a = m \pmod{p}.$$

Which recovers the message $m \pmod{p}$.

3.4.7 Decryption Criterion

To ease notation, we let

$$m' = \left[m + H(m, [r \circledast h]_p)X^{n-k} + G([r \circledast h]_p)\right]_p$$

be the polynomial used by Alice for encryption. (That is, $e = r \circledast h + m' \mod q$.) In order for the decryption process to work, it is necessary that

$$|f \circledast m'pr \circledast g|_\infty < q.$$

We have found that this will virtually always be true if we choose parameters so that

$$|f \circledast m'|_\infty \leq \frac{q}{4} \text{ and } |pr \circledast g|_\infty \leq \frac{q}{4}.$$

Proposition 3.1. *For any $\epsilon > 0$ there are constants $\gamma_1, \gamma_2 > 0$, depending on ϵ and N, such that for randomly chosen polynomials f, $g \in R$, the probability is greater than $1 - \epsilon$ that they satisfy*

$$\gamma_1 |f|_2 |g|_2 \leq |f \circledast g|_\infty \leq \gamma_2 |f|_2 |g|_2.$$

Of course, this proposition would be useless from the practical viewpoint if the ratio γ_2/γ_1 were very large for small ϵ's. However, it turns out that even for moderately large values of N and very small values of ϵ, the constants γ_1, γ_2 are not at all extreme. We have verified this experimentally for a large number of parameter values.

In view of the above Proposition, this suggests that we take

$$|f|_2|m|_2 \approx \frac{q}{4}\gamma_2 \quad \text{and} \quad |r|_2|g|_2 \approx \frac{q}{4}\gamma_2$$

for a γ_2 corresponding to a small value for ϵ. For example, experimental evidence suggests that for $N = 167$ and $N = 503$, appropriate values for γ_2 are 0.27 and 0.17 respectively.

3.4.8 Homomorphic Properties

The multiple operation homomorphic properties of the scheme are best seen through the lens of the decryption equation. In particular, consider ciphertexts

$$c_1 := ps_1 \circledast h + m_1 \pmod{q} \tag{7}$$

and

$$c_2 := ps_2 \circledast h + m_2 \pmod{q}. \tag{8}$$

that encrypt messages m_1 and m_2 under public key h, with noise terms s_1 and s_2. A little algebraic manipulation shows that $c_{add} := c_1 + c_2$ and $c_{mult} := c_1 c_2$ are ciphertexts that encrypt the sum and product of m_1 and m_2, respectively, albeit with larger error terms.

Addition

Namely, decrypting $c_1 + c_2$ with the secret key f in the ring R_q gives us:

$$\begin{aligned}
\mu_{add} &= f(c_1 + c_2) && \text{by (7) and (8)} \\
&= fc_1 + fc_2 \\
&= f(ps_1 \circledast h + m_1) + f(ps_2 \circledast h + m_2) \\
&= ps_1 \circledast h + fm_1 + ps_2 \circledast h + fm_2
\end{aligned}$$

Reducing μ_{add} modulo p, we find

$$\mu_{add} \pmod{p} = fm_1 + fm_2 \pmod{p}$$

We can now compute the last step in the decryption scheme and multiply by f_p as follows.

$$\begin{aligned} m_{add} &= f_p(fm_1 + fm_2) \pmod{p} \\ &= f_p \cdot fm_1 + f_p \cdot fm_2 \pmod{p} \\ &= m_1 + m_2 \end{aligned}$$

This shows that decrypting $c_1 + c_2$ using the secret key f results in the sum of the two messages, assuming the error does not grow to be too large.

Multiplication

Namely, decrypting c_1c_2 with the secret key f in the ring R_q gives us:

$$\begin{aligned} \mu_{mult} &= f^2c_1c_2 && \text{by (7) and (8)} \\ &= fc_1 \cdot fc_2 \\ &= f(ps_1 \circledast h + m_1) \cdot f(ps_2 \circledast h + m_2) \\ &= (ps_1 \circledast h + fm_1) \cdot (ps_2 \circledast h + fm_2) \\ &= (ps_1 \circledast h)(ps_2 \circledast h + fm_2) + (ps_2 \circledast h)(fm_1) + f^2m_1m_2 \end{aligned}$$

Reducing μ_{mult} modulo p, we find

$$\mu_{mult} \pmod{p} = f^2m_1m_2 \pmod{p}$$

We can now compute the last step in the decryption scheme and multiply by f_p^2 as follows.

$$\begin{aligned} m_{add} &= f_p^2 \cdot f^2m_1m_2 \pmod{p} \\ &= m_1m_2 \end{aligned}$$

This shows that decrypting c_1c_2 using the secret key f results in the sum of the two messages, again assuming the error does not grow to be too large.

4 Our Results and Techniques

We show how to construct a single key somewhat homomorphic encryption scheme based on the NTRU encryption system first proposed by Hoffstein, Pipher and Silverman. [HPS98] More precisely, we rely on a variant of the NTRU scheme proposed by Stehlé and Steinfeld. [SS11]

In Section 4.1, we first review definitions and facts form the literature that we use extensively. In Section 4.2, we describe the our single-key encryption scheme. In Section 4.3, we discuss its correctness, and in Section 4.4 show that is single-key somewhat homomorphic. We conclude this chapter showing that our scheme can be made fully homormophic utilizing relinearization and modulus reduction to create a bootstrappable scheme in Section 4.5.

4.1 Slightly Modified Version of NTRU

NTRU is the fastest known lattice-based encryption scheme and it's moderate key-sizes, excellent asymptotic performance and conjectured resistance to quantum computers make it a desirable alternative to factorization and discrete=log based encryption schemes. However, since it's introduction, doubts have regularly arisen on its security and that of its more recent digital signature counterpart, NTRUSign. Stelhé and Steinfeld presented a rewrite of the NTRU scheme in 2011, modifying NTRUEncyrpt and NTRUSign to make them provably secure in the standard model, under the assumed quantum hardness of standard worst-case lattice problems, restricted to a family of lattices related to some cyclotomic fields. Their main contribution was to show that if the secret key polynomials of the encryption scheme are selected by rejection from discrete Gaussians, then the public key, which is their ratio, is statistically indistinguishable from uniform over its domain. [SS11]

Brief comparison between NTRUEncrypt and its provably secure variant Let R_{NTRU} be the ring $\mathbb{Z}[x]/\langle x^n 1\rangle$ with n prime. Let q be a medium-size integer, typically a power of 2 of the same order of magnitude as n. Finally, let $p \in R_{\mathsf{NTRU}}$ with small coefficients, co-prime with q and such that the plaintext space R_{NTRU}/p is large. E.g, if q is chosen as above, one may take $p = 3$ or $p = x + 2$.

The NTRUEncrypt secret key is a pair of polynomials $(f, g) \in R^2_{\mathsf{NTRU}}$ that are sampled randomly with large prescribed proportions of zeros, and with their other coefficients belonging to $\{1, 1\}$. For improved decryption efficiency, one may choose f such that $f = 1 \mod p$. With high probability, the polynomial f is invertible modulo q and modulo p, and if that is the case the public-key is $h = pg/f \mod q$ (otherwise, the key generation process is restarted). To encrypt a message $m \in R_{\mathsf{NTRU}}/p$, one samples a random element $s \in R_q$. The

following procedure allows the owner of the secret key to decrypt:

- Compute fc and reduce the result modulo q. If the ciphertext was properly generated, this gives $pgs+fm \mod q$. Since the five involved ring elements have small coefficients, it can be expected that after reduction modulo q the obtained representative is exactly $pgs + fm$ (in R_{NTRU}).

- Reduce the result of the previous step modulo p. This should provide $fm \mod p$.

- Multiply the result of the previous step by the inverse of f modulo p (this step becomes vacuous if $f = 1 \mod p$).

Note that the encryption process is probabilistic, and that decryption errors can occur for some sets of parameters. However, it is possible to arbitrarily decrease the decryption error probability, and even to prevent them from occurring, by setting the parameters carefully.

In order to achieve CPA-security under the assumption that standard lattice problems are (quantumly) hard to solve in the worst-case for the family of ideal lattices, we make a few modifications to the original NTRU scheme (which preserve its quasi-linear computation and space complexity):

1. We replace R_{NTRU} by $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ with n a power of 2. We will exploit the irreducibility of x^n+1 and the fact that R is the ring of integers of a cyclotomic number field.

2. We choose $q \leq \mathsf{poly}(n)$ as a prime integer such that $f = x^n + 1$ splits into n distinct linear factors modulo q. This allows us to use the search to decision reduction for $\mathsf{R\text{-}LWE}$ with ring $R_q := R/q$. This also allows us to take $p = 2$.

3. We sample f and g from discrete Gaussian distributions over the set of elements of R, rejecting the samples that are not invertible modulo q. We show that $f/g \mod q$ is essentially uniformly distributed over the set of invertible elements of R_q. We may also choose $f = pu + 1$ with u sampled from a discrete Gaussian, to simplify decryption.

4. We add a small error term e in the encryption: $c = hs + pe + m \mod q$, with s and e sampled from the Gaussian error distribution. [SS11]

4.2 The Scheme

We sketch here our variant of the LTV encryption scheme.

The main differences between the original NTRU scheme and our variant are threefold:

- Whereas the original NTRU scheme adds a deterministic noise to the ciphertext, the variant considered here adds noise chosen from a distribution with bounded support (specifically, a discrete Gaussian distribution), a modification recently introduced by Stehlé and Steinfeld [SS11]. It seems that this could only improve security; indeed, the purpose of the Stehlé-Steinfeld work was to prove the security of NTRU based on worst case hardness assumptions on ideal lattices,

- We do all our operations modulo x^n+1 where n is a power of 2 as in [SS11], as opposed to $x^n - 1$ in the original NTRU.

- Our parameters are more aggressive than in [HPS98] and [SS11] to support homomorphisms. As a result, the worst-case to average-case connection shown by [SS11] does not carry over to our setting of parameters.

The scheme is parametrized by the following quantities:

- an integer n

- a prime number q and

- a B-bounded error distribution χ over the ring $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ (i.e., χ is a distribution over polynomials whose coefficients are all at most $B(\chi)$ in absolute value).

The parameters n, q and χ are public. We show how to encrypt bits using the scheme. All operations in the scheme take place in the ring $R_q = R/qR$.

4.2.1 Preliminaries

We work over the rings $R = \mathbb{Z}/\langle\phi(x)\rangle$ and $R_q = R/qR$ for some degree n integer polynomial $\phi(x) \in \mathbb{Z}/\langle\phi(x)\rangle$, i.e., the ring of degree n polynomials modulo $\phi(x)$ with coefficients in $\mathbb{Z}_q$. Addition in these rings is done component-wise in their coefficients (thus, their additive group is isomorphic to $\mathbb{Z}^n$ and $\mathbb{Z}_q^n$ respectively). Multiplication is simply polynomial multiplication modulo $\phi(x)$ (and also q, in the case of the ring R_q). Thus an element in R (or R_q) can be viewed as a degree n polynomial over $\mathbb{Z}$ (or $\mathbb{Z}_q$). We represent such an element using the vector of its n coefficients, each in the range $\{-\lfloor q/2\rfloor, ..., \lfloor q/2\rfloor\}$. For an element $a(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1} \in R$, we let $\|a\|_\infty = \max|a_i|$ denote its ℓ_∞ norm.

In this work, we set $\phi(x) = x^n+1$ to be in the n^{th} cyclotomic polynomial, where n is a power of two. We use distributions over the ring $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$. To show the homomorphic

properties of the scheme, the only property of these distributions we use is the magnitude of the polynomials' coefficients. Hence, we define a B-bounded distribution to be a distribution over R where ℓ_∞-norm of a sample is bounded.

Definition 4.1 (B-Bounded Polynomial)**.** A polynomial $e \in R$ is called B-bounded if $\|e\|_\infty \leq B$.

Definition 4.2 (B-Bounded Distribution)**.** A distribution ensemble $\{\chi_n\}_{n\in N}$, supported over R, is called B-bounded (for $B = B(n)$) if for all e in the support of χ_n, we have $\|e\|_\infty < B$. In other words, a B-bounded distribution over R outputs a B-bounded polynomial.

We present some elementary facts about the Gaussian distribution and multiplication over the ring R. The first fact shows that the Gaussian distribution over $\mathbb{Z}^n$ with standard deviation r, denoted by $D_{\mathbb{Z}^n,r}$, outputs a $(r\sqrt{n})$-bounded and statistically close to the discrete Gaussian. The second says that multiplication in the ring R increases the norm of the constituent elements only by a small amount.

Lemma 4.1 (See [MR07], Theorem 4.4)**.** *Let $n \in \mathbb{N}$. For any real number $r > \omega(\sqrt{\log n})$, we have*

$$\underset{x\leftarrow D_{\mathbb{Z}^n,r}}{Pr}\left[\|x\| > r\sqrt{n}\right] \leq 2^{-n+1}$$

Define the *truncated* discrete Gaussian distribution to be one that samples a polynomial according to the discrete Gaussian $D_{\mathbb{Z}^n,r}$ and repeat the sampling if the polynomial is not $(r\sqrt{n})$-bounded. By Lemma 4.1, this distribution is statistically close to the discrete Gaussian.

Lemma 4.2 (See [MR07], Theorem 4.4)**.** *Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and let $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$. For any s, $t \in R$,*

$$\|s \cdot t \pmod{\phi(x)}\| \leq \sqrt{n} \cdot \|s\| \cdot \|t\|$$
$$\|s \cdot t \pmod{\phi(x)}\|_\infty \leq \sqrt{n} \cdot \|s\|_\infty \cdot \|t\|_\infty$$

Lemma 4.2 yields the following corollary.

Corollary 4.1. *Let $n \in \mathbb{N}$, $\phi(x) = x^n + 1$ and $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$. Let χ be a B-bounded distribution over the ring R and let $s_1, ...s_k \leftarrow \chi$. Then $\prod_{i=1}^{k} s_i$ is $(n^{k-1}B^k)$-bounded.*

Utilizing Corollary 4.1, we can then expand to elaborate on the bounds of our encryption scheme's message, m.

Corollary 4.2. *Let $n \in \mathbb{N}$, $\phi(x) = x^n + 1$, $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$ and every other symbol as defined in the basic scheme. Then fc is $(6nB^n + 2nB + B + 1)$-bounded.*

Proof. Let $n \in \mathbb{N}$, $\phi(x) = x^n + 1$, $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$ and every other symbol as defined in the basic scheme. By setting $\phi(x)$ as such, the parameters used in the basic scheme are bounded as shown in Table 5.

Table 5: Parameter Bounds

Parameter	Bound
g	B-bounded by definition
u	B-bounded by definition
s	B-bounded by definition
e	B-bounded by definition
m	1-bounded by definition

Utilizing equation (12), we recall

$$f := 2u + 1$$

Therefore, we can express f as $2B + 1$-bounded. Similarly, using this result with equation (13), we recall

$$h := 2gu$$

Therefore, we can express h as $4B^2 + 2B$-bounded. Then, from Lemma 4.2, we have

$$\|gs \pmod{\phi(x)}\|_\infty \leq \sqrt{n}\|g\|_\infty \cdot \|s\|_\infty = \sqrt{n}B^2 \tag{9}$$

and

$$\|fe \pmod{\phi(x)}\|_\infty \leq \sqrt{n}\|f\|_\infty \cdot \|e\|_\infty = \sqrt{n}(2B+1)(B) = \sqrt{n}(2B^2 + B). \tag{10}$$

Using equations (9) and (10), we see

$$\begin{aligned}
\|\mu \pmod{\phi(x)}\|_\infty &= \|fc \pmod{\phi(x)}\| \\
&= \|f(hs + 2e + m) \pmod{\phi(x)}\|_\infty \\
&= \|f[(2gf^{-1})s + 2e + m] \pmod{\phi(x)}\|_\infty \\
&= \|2gs + 2fe + (2u + 1)m \pmod{\phi(x)}\|_\infty \\
&= \|2gs + 2fe + 2um + m \pmod{\phi(x)}\|_\infty \\
&= 2\|gs \pmod{\phi(x)}\|_\infty + 2\|fe \pmod{\phi(x)}\|_\infty \\
&\qquad + 2\|um \pmod{\phi(x)}\|_\infty + \|m \pmod{\phi(x)}\|_\infty \\
&= 2\sqrt{n}B^2 + 2\sqrt{n}(2B^2 + B) + B + 1 \\
&= 6\sqrt{n}B^2 + (2\sqrt{n} + 1)B + 1 \\
\|\mu \pmod{\phi(x)}\|_\infty &\leq 6nB^2 + 2nB + B + 1
\end{aligned}$$

□

We need to choose a prime q large enough to guarantee that

$$q/2 > \|\mu\|_\infty$$

Ensuring this inequality holds eliminates *wrap-around*.

We can then describe the bounds needed in order to ensure our scheme is homomorphic as follows.

Corollary 4.3. *Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$, and every other symbol as defined in the basic scheme. Then $f^2(c_1 + c_2)$ is $(37n^2)$-bounded.*

Proof. Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$, and every other symbol as defined in the basic scheme. By setting $\phi(x)$ as such, the parameters used in the basic scheme are bounded as above in the proof of Corollary 4.2.
Then, from Lemma 4.2, we have

$$
\begin{aligned}
\|c \pmod{\phi(x)}\|_\infty &= \|hs + 2e + m \pmod{\phi(x)}\|_\infty \\
&= \|hs \pmod{\phi(x)}\|_\infty + 2\|e \pmod{\phi(x)}\|_\infty \\
&\quad + \|m \pmod{\phi(x)}\|_\infty \\
&= \sqrt{n}\|h\|_\infty \cdot \|s\|_\infty + 2B + 1 \\
&= \sqrt{n}(4B^2 + 2B)(B) + 2B + 1 \\
&= \sqrt{n}(4B^3 + 2B^2) + 2B + 1 \\
&\leq n(4B^3 + 2B^2) + 2B + 1 \\
\|c \pmod{\phi(x)}\|_\infty &= 4nB^3 + 2nB^2 + 2B + 1 \qquad (11)
\end{aligned}
$$

Now for two ciphertexts, c_1 and c_2, as described in equations (19) and (20) we have

$$f(c_1 + c_2) = fc_1 + fc_2$$

Continuing using equation (11) we see

$$\begin{aligned}
\|f(c_1c_2) \pmod{\phi(x)}\|_\infty &= \|fc_1 + fc_2 \pmod{\phi(x)}\|_\infty \\
&= \|fc_1 \pmod{\phi(x)}\|_\infty + \|fc_2 \pmod{\phi(x)}\|_\infty \\
&= \sqrt{n}\|f\|_\infty \cdot \|c_1\|_\infty + \sqrt{n}\|f\|_\infty \cdot \|c_2\|_\infty \\
&= \sqrt{n}(2B+1)\left\{2(4nB^3 + 2nB^2 + 2B + 1)\right\} \\
&= \sqrt{n}\left\{16nB^4 + 16nB^3 + (4n+8)B^2 + 8B + 2\right\} \\
&\leq n\left\{16nB^4 + 16nB^3 + (4n+8)B^2 + 8B + 2\right\} \\
&= 16n^2B^4 + 16n^2B^3 + (4n^2+8n)B^2 + 8nB + 2n \\
&\leq 16n^2 + 16n^2 + 4n^2 + 8n + 8n + 2n \\
&= 36n^2 + 18n \\
\|f(c_1 + c_2) \pmod{\phi(x)}\|_\infty &\leq 37n^2
\end{aligned}$$

Therefore we have shown $f^2(c_1 + c_2)$ is $(37n^2)$-bounded. □

Corollary 4.4. *Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$, and every other symbol as defined in the basic scheme. Then $f(c_1c_2)$ is $(324n^5)$-bounded.*

Proof. Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$, and every other symbol as defined in the basic scheme. By setting $\phi(x)$ as such, the parameters used in the basic scheme are bounded as above in the proff of Corollary 4.2.
Now for two ciphertexts, c_1 and c_2, as described in equations (19) and (20), both bounded as described in equation (11), we have

$$f^2(c_1c_2) = fc_1 \cdot fc_2$$

Continuing using equation (11) we see

$$\begin{aligned}
\|f^2(c_1c_2) \pmod{\phi(x)}\|_\infty &= \|f^2c_1c_2 \pmod{\phi(x)}\|_\infty \\
&= \sqrt{n}\|fc_1 \pmod{\phi(x)}\|_\infty \cdot \|fc_2 \pmod{\phi(x)}\|_\infty \\
&= n^{3/2}{\|f\|_\infty}^2 \cdot \|c_1\|_\infty \cdot \|c_2\|_\infty \\
&= n^{3/2}{\|f\|_\infty}^2 \cdot (\|c_1\|_\infty)^2 \\
&= n^{3/2}(2B+1)^2(4nB^3+2nB^2+2B+1)^2 \\
&= n^{3/2}(4B^2+4B+1)(16n^2B^6+16n^2B^5 \\
&\qquad + 4n^2B^4+16nB^4+16nB^3 \\
&\qquad + 4nB^2+4B^2+4B+1) \\
&= n^{3/2}(64n^2B^8+128n^2B^7+96n^2B^6+64nB^6 \\
&\qquad + 32n^2B^5+128nB^5+4n^2B^4+96nB^4 \\
&\qquad + 16B^4+32nB^3+32B^3+4nB^2+24B^2 \\
&\qquad + 8B+1) \\
&= 64n^{7/2}B^8+128n^{7/2}B^7+96n^{7/2}B^6+64n^{5/2}B^6 \\
&\qquad + 32n^{7/2}B^5+128n^{5/2}B^5+4n^{7/2}B^4+96n^{5/2}B^4 \\
&\qquad + 16B^4+32n^{5/2}B^3+32B^3+4n^{5/2}B^2+24B^2 \\
&\qquad + 8B+1 \\
&\leq 324n^{7/2}+324n^{5/2}+81 \\
&\leq 324n^4+324n^3+81 \|f(c_1c_2) \pmod{\phi(x)}\|_\infty \leq 324n^5.
\end{aligned}$$

Therefore we have shown $f(c_1c_2)$ is $(324n^5)$-bounded. □

4.2.2 Key Generation

Keygen: Sample bounded polynomials u, $g \leftarrow \chi$ and set

$$f := 2u + 1, \tag{12}$$

so that $f = 1 \pmod 2$. Define h to be

$$h := 2gf^{-1} \in R_q, \tag{13}$$

We will now define the public key as the tuple

$$pk := (h, q) \tag{14}$$

and the secret key as the tuple

$$sk := (f, q). \tag{15}$$

If f is not invertible over R_q, resample u.

4.2.3 Encryption

Enc: Sample bounded polynomials s, $e \leftarrow \chi$. Output the ciphertext

$$c := hs + 2e + m \in R_q \tag{16}$$

Here we adopt the notation the element "$\in R_q$" such that all operations are done modulo $\phi(x)$ and reducing $\mathbb{Z}$.

4.2.4 Decryption

Dec: Let

$$\mu := fc \in R_q, \tag{17}$$

We then output $\mu \pmod 2$ as the message.

4.2.5 Example of Our Scheme

Now, let us look at a small example of our scheme.

Our small example will be parametrized by the following quantities:

- $n = 4$
- $q = 89$
- $\phi(x) = x^4 + 1$
- $B = 1$, thus our sampled polynomials from χ will be 1-bounded. In other words, all coefficients for any polynomial $p \leftarrow \chi$ lie in the range $[-1, 1]$.

We will do all operations modulo 89 and $x^4 + 1$ and define

$$R := \mathbb{Z}[x]/\left\langle x^4 + 1 \right\rangle \quad \text{and} \quad R_q := R/89R$$

Keygen: We sampled $u, g \leftarrow \chi$ and have

$$u = x^2 \quad \text{and} \quad g = x + x^2.$$

Computing f as described in Section 4.2, we find

$$\begin{aligned} f &= 2u + 1 \\ &= 2(x^2) + 1 \\ &= 1 + 2x^2. \end{aligned}$$

Now we apply the Extended Euclidean Algorithm, as previously discussed in Section 2.1.2, to ensure f is invertible in the R_q. In other words, that is there exists some f^{-1} such that $ff^{-1} = 1 \in R_q$.

$$(x^4 + 1) = (2x^2 + 1)(16x^2 + 8) + 9$$

$$(16x^2 + 8) = (9)(19x^2 + 25) + 0$$

Therefore, we have

$$9 = (x^4 + 1) - (2x^2 - 1)(16x^2 + 8)$$

$$1 = (7)(x^4 + 1) - (2x^2 - 1)(112x^2 + 56)$$

Thus, we have found the inverse of f as

$$\begin{aligned} f^{-1} &= -112x^2 - 56 \\ &= 12x^2 + 6 \pmod{89}. \end{aligned}$$

Using f^{-1} and the definition of h, as described in Section 4.2, we find

$$\begin{aligned} h &= 2(x + x^2)(12x^2 + 6) \\ &= 12x + 12x^2 + 24x^3 + 24x^3 \\ &= -24 + 12x + 12x^2 + 24x^3 \pmod{x^4 + 1} \\ &= 7 + 12x + 12^2 - 7x^3 \pmod{89}. \end{aligned}$$

Thus, we define our public and secret keys as

$$pk := (7 + 12x + 12^2 - 7x^3, 89) \text{ and } sk := (-1 + 2x^2, 89),$$

respectively.

Enc: Assume we would like to encrypt to message $m = 1$. We begun by sampling $s, e \leftarrow \chi$ and have

$$s = 1 + x + x^3 \text{ and } e = x^3.$$

We can now compute of ciphertext of m as follows.

$$\begin{aligned} c &= (7 + 12x + 12^2 - 7x^3)(1 + x + x^3) + 2(x^3) + 1 \\ &= 8 + 19x + 24x^2 + 14x^3 + 5x^4 + 12x^5 - 7x^6 \\ &= 3 + 7x + 31x^2 + 14x^3 \pmod{x^4 + 1} \\ &= 3 + 7x + 14x^3 \pmod{89} \end{aligned}$$

Dec: Suppose we would now like to decrypt our message. We proceed by comping μ as defined in Section 4.2.

$$\begin{aligned}\mu &= fc \\ &= (-1 + 2x^2)(3 + 7x + 14x^3) \\ &= -3 - 7x + 6x^2 + 28x^5 \\ &= -3 - 35x + 6x^2 \pmod{x^4 + 1} \\ &= -3 - 4x + 6x^2 \pmod{89}\end{aligned}$$

Reducing μ modulo 2 we see,

$$\begin{aligned}\mu \pmod 2 &= 1 \\ &= m.\end{aligned}$$

Thus, we have successfully encrypted and decrypted our message without any disruption. Later on in Section 4.2.6, we will show an example where wrap around error can occur.

4.2.6 Example of Wrap Around Error

Now, let us look at a small example of our scheme which causes wrap around error.
Example 4.1. This small example will be parametrized by the following quantities:

- $n = 4$
- $q = 7$
- $\phi(x) = x^4 + 1$
- $B = 1$, thus our sampled polynomials from χ will be 1-bounded. In other words, all coefficients for any polynomial $p \leftarrow \chi$ lie in the range $[-1, 1]$.

Note. Notice that the only value we have changed in this example from the last is reducing q from 31 to 7. Thus reducing to a prime integer much smaller than the bound on q as described in Section 4.2.1.

We will do all operations modulo 7 and $x^4 + 1$ and define

$$R := \mathbb{Z}[x]/\left\langle x^4 + 1 \right\rangle \quad \text{and} \quad R_q := R/7R$$

Keygen: We will use the same parameters for generating our keys as described in the previous example.

$$u = -1 + x^2, g = x + x^2 \quad \text{and} \quad f = -1 + 2x^2$$

Since we have changed the ring in which we are doing computations, we must recompute f^{-1}.

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

$$(-1 + 2x^2) = (3)(2 + 3x^2) + 0$$

Therefore, we have

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

$$1 = (5)(x^4 + 1) - (2x^2 - 1)(20x^2 + 10)$$

Thus, we have found the inverse of f as

$$\begin{aligned} f^{-1} &= -20x^2 - 10 \\ &= -3 + x^2 \pmod 7. \end{aligned}$$

Using f^{-1} and the definition of h, as described in Section 4.2, we find

$$\begin{aligned} h &= 2(x + x^2)(x^2 - 3) \\ &= -6x - 6x^2 + 2x^3 + 2x^4 \\ &= -2 - 6x - 6x^2 + 2x^3 \pmod{x^4 + 1} \\ &= -2 + x + x^2 + 2x^3 \pmod 7. \end{aligned}$$

Thus, we define our new public and secret keys as

$$pk := (-2 + x + x^2 + 2x^3, 7) \quad \text{and} \quad sk := (-1 + 2x^2, 7),$$

respectively.

Enc: We would still like to encrypt to message $m = 1$. Utilizing the same s and e as the previous example, we have

$$s = 1 + x + x^3 \quad \text{and} \quad e = x^3.$$

We can now compute of ciphertext of m as follows.

$$\begin{aligned} c &= (-2 + x + x^2 + 2x^3)(1 + x + x^3) + 2(x^3) + 1 \\ &= -1 - x + 2x^2 + 3x^3 + 3x^4 + x^5 + 2x^6 \\ &= -4 - 2x + 3x^3 \pmod{x^4 + 1} \\ &= 3 - 2x + 3x^3 \pmod{7} \end{aligned}$$

Dec: Suppose we would now like to decrypt our message. We proceed by comping μ as defined in Section 4.2.

$$\begin{aligned} \mu &= fc \\ &= (-1 + 2x^2)(3 - 2x + 3x^3) \\ &= -3 + 2x + 6x^2 - 7x^3 + 6x^5 \\ &= -3 - 4x + 6x^2 - 7x^3 \pmod{x^4 + 1} \\ &= -3 + 3x - 1x^3 \pmod{7} \end{aligned}$$

Reducing μ modulo 2 we see,

$$\mu \pmod{2} = 1 + x^2 + x^3$$

$$m.$$

Notice we cannot recover m from μ because a wrap-around error had occurred in both the x and x^2 terms' coefficients.

4.3 Why Decryption Works

We can show decryption works, assuming there is no overflow or wrap around error, by the following. All operations are done in the ring R_q.

$$\begin{aligned}
\mu &= fc && \text{by (17)} \\
&= f(hs + 2e + m) && \text{by (16)} \\
&= fhs + 2fe + fm \\
&= f(2gf^{-1}s + 2fe + fm && \text{by (13)} \\
&= 2g(ff^{-1})s + 2fe + fm \\
&= 2gs + 2fe + fm && \text{since } ff^{-1} = 1 \in R_q \\
&= 2(gs + ef) + fm \\
&= 2(gs + ef) + (2u + 1)m && \text{by (12)} \\
&= 2(gs + ef + um) + m \\
\mu &= m \pmod 2 && (18)
\end{aligned}$$

4.4 Homomorphic Properties

The multiple operation homomorphic properties of the scheme are best seen through the lens of the decryption equation. In particular, consider ciphertexts

$$c_1 := hs_1 + 2e_1 + m_1 \in R_q \tag{19}$$

and

$$c_2 := hs_2 + 2e_2 + m_2 \in R_q. \tag{20}$$

that encrypt messages m_1 and m_2 under public key $pk := (h, q)$, with noise terms e_1 and e_2. A little algebraic manipulation shows that $c_{add} := c_1 + c_2$ and $c_{mult} := c_1 c_2$ are ciphertexts that encrypt the sum and product of m_1 and m_2, respectively, albeit with larger error terms.

Later on we will go on to show that modulus reduction may not be needed in the case of addition, however it will almost always be necessary when multiple multiplications are to be used in the circuit.

Extending this to circuits, we observe that the effective secret key required to decrypt the ciphertext c resulting from evaluating a multivariate polynomial function on the inputs is f^d where d is the degree of the polynomial function. This makes the secret key required to decrypt c *dependent* on the circuit evaluated, which is unacceptable even for somewhat homomorphic encryption. We use the relinearization technique from [11] to transform the ciphertext into one that can be decrypted using the secret key f (namely, reduce all the exponents from d to 1), after every operation. In effect, this ensures that the secret key is no longer dependent on the function involved in the computation. With the use of relinearization, one can show that the scheme is *somewhat homomorphic*, i.e., capable of evaluating circuits of depth $\epsilon \log n$ for some small constant $\epsilon < 1$. [LATV12]

To turn this into a fully homomorphic encryption scheme we use the technique of modulus reduction from the work of Brakerski and Vaikuntanathan [BV11a], later refined in [BV11b]. Modulus reduction shows how to reduce the magnitude of the error (while simultaneously reducing the size of the modulus). This technique works transparently in the single key setting, giving us a leveled FHE scheme. Finally, to turn this into a full-fledged FHE scheme (whose complexity is independent of the complexity of the function being computed), we show that we can reach a bootstrappable scheme utilizing modulus reduction.

4.4.1 Addition

Namely, decrypting $c_1 + c_2$ with the secret key $sk := (f, q)$ in the ring R_q gives us:

$$
\begin{aligned}
f(c_1 + c_2) &= fc_1 + fc_2 && \text{by (19) and (20)} \\
&= f(hs_1 + 2e_1 + m_1) + f(hs_2 + 2e_2 + m_2) \\
&= fhs_1 + 2fe_1 + fm_1 + fhs_2 + 2fe_2 + fm_2 \\
&= f(2gf^{-1})s_1 + 2fe_1 + fm_1 + f(2gf^{-1})s_2 + 2fe_2 + fm_2 && \text{by (13)} \\
&= 2g(ff^{-1})s_1 + 2fe_1 + fm_1 + 2g(ff^{-1})s_2 + 2fe_2 + fm_2 \\
&= 2gs_1 + 2fe_1 + fm_1 + 2gs_2 + 2fe_2 + fm_2 && \text{since } ff^{-1} = 1 \in R_q \\
f(c_1 + c_2) &= 2(gs_1 + fe_1 + gs_2 + fe_2) + fm_1 + fm_2
\end{aligned}
$$

From this point on we will consider the error due to addition

$$E_{add} = gs_1 + fe_1 + gs_2 + fe_2. \tag{21}$$

Continuing with our previous results we have:

$$f(c_1 + c_2) = 2E_{add} + f(m_1 + m_2) \in R_q. \tag{22}$$

Reducing this result modulo 2, we find:

$$\begin{aligned} f(c_1 + c_2) &= 2E_{add} + f(m_1 + m_2) \pmod 2 \\ &= 2E_{add} + (2u + 1)(m_1 + m_2) \pmod 2 && \text{by (12)} \\ &= 2[E_{add} + u(m_1 + m_2)] + m_1 + m_2 \pmod 2 \\ f(c_1 + c_2) &= m_1 + m_2 \pmod 2 && (23) \end{aligned}$$

This shows that decrypting $c_1 + c_2$ using f results in the sum of the two messages, assuming the error does not grow to be too large.

4.4.2 Multiplication

Likewise, decrypting c_1c_2 with the secret key $sk := (f, q)$ in the ring R_q we have:

$$\begin{aligned} f^2(c_1c_2) &= fc_1 \cdot fc_2 \\ &= f(hs_1 + 2e_1 + m_1) \cdot f(hs_2 + 2e_2 + m_2) && \text{by (19) and (20)} \\ &= (fhs_1 + 2fe_1 + fm_1) \cdot (fhs_2 + 2fe_2 + fm_2) \\ &= (f(2gf^{-1})s_1 + 2fe_1 + fm_1) \cdot (f(2gf^{-1})s_2 + 2fe_2 + fm_2) && \text{by (13)} \\ &= (2g(ff^{-1})s_1 + 2fe_1 + fm_1) \cdot (2g(ff^{-1})s_2 + 2fe_2 + fm_2) \\ &= (2gs_1 + 2fe_1 + fm_1) \cdot (2gs_2 + 2fe_2 + fm_2) && \text{since } ff^{-1} = 1 \in R_q \\ &= 4g^2s_1s_2 + 4gs_1fe_2 + 2gs_1fm_2 + 4gs_2fe + 4f^2e_1e_2 + 2fe_1fm_2 + f^2m_1m_2 \\ &= 2(2g^2s_1s_2 + 2gs_1fe_2 + gs_1fm_2 + 2gs_2fe + 2f^2e_1e_2 + 1fe_1fm_2) + f^2m_1m_2 \end{aligned}$$

From this point on we will consider the error due to multiplication

$$E_{mult} = 2g^2 s_1 s_2 + 2g s_1 f e_2 + g s_1 f m_2 + 2g s_2 f e + 2f^2 e_1 e_2 + 1 f e_1 f m_2 \tag{24}$$

Continuing with our previous results, we have:

$$f(c_1 c_2) = 2E_{mult} + f(m_1 m_2) \in R_q \tag{25}$$

Reducing this result modulo 2 we find:

$$\begin{aligned} f^2(c_1 c_2) &= 2E_{mult} + f(m_1 m_2) \pmod 2 \\ &= 2E_{mult} + (2u+1)(m_1 m_2) \pmod 2 && \text{by (12)} \\ &= 2[E_{mult} + u(m_1 m_2)] + m_1 m_2 \pmod 2 \\ f^2(c_1 c_2) &= m_1 m_2 \pmod 2 \end{aligned} \tag{26}$$

This shows that decrypting $c_1 c_2$ using f^2 results in the product of the two messages, assuming that the error does not grow to be too large. We will show later on in Section 4.5.2 that we can indeed remove the need to decrypt the multiplication of d ciphertexts using f^d and instead utilize a single multiplication of f.

4.5 From Somewhat to Fully Homomorphic Encryption

We use Gentry's bootstrapping theorem [Gen09a, Gen09b] to convert our single-key somewhat homomorphic scheme into a single-key fully homomorophic one. Unfortunately, as is described in Section 4.5.1, we can not simply apply the bootstrapping theorem directly to the somewhat homomorphic encryption scheme from Section 4.2. We therefore turn to modulus reduction, a technique introduced by [BV11a, BV11b], which can later be applied to convert our somewhat homomorphic scheme into a bootstrappable scheme. We will first describe the bootstrapping theorem, relinearization and then present the modulus reduction technique and how we can apply it in our case.

4.5.1 Bootstrapping

We remind the reader of the definition of a bootstrappable encryption scheme and present Gentry's bootstrapping theorem [Gen09a, Gen09b] that states a bootstrappable scheme can be converted into a fully homomorphic one. We modify the theorem and the corresponding definitions from their original presentation. Taking $N = 1$ yields the theorem and the definitions from [Gen09a, Gen09b].

Definition 4.3 (Bootstrappable Scheme)**.** Let $\mathcal{E} = \left\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\right\}_{N>0}$ be a family of single-key $\mathcal{C}$-homomorphic encryption schemes, and let f_{add} and f_{mult} be the augmented decryption functions of the scheme defined as

$$f_{add}(c_1, c_2) = \mathsf{Dec}(c_1) \ \ \mathsf{XOR} \ \ \mathsf{Dec}(c_2) \ \ \text{and} \ \ f_{mult}(c_1, c_2) = \mathsf{Dec}(c_1) \ \ \mathsf{AND} \ \ \mathsf{Dec}(c_2)$$

Then $\mathcal{E}$ is **bootstrappable** if[1]

$$\{f_{add}(c_1, c_2), f_{mult}(c_1, c_2)\}_{c_1, c_2} \subseteq \mathcal{C}$$

Namely, the scheme can homomorphically evaluate f_{add} and f_{mult}.

In other words, we consider a scheme bootstrappable if it can homomorphically evaluate its own decryption circuit. Thus, the scheme must be homomorphic with respect to any gates that appear in the decryption circuit and must be able to evaluate the entire decryption circuit without creating an excess of noise.

Unfortunately, the somewhat homomorphic scheme that we described in Section 4.2 is not bootstrappable. We therefore turn to modulus reduction, which will allow us to convert our somewhat homomorphic encryption scheme into a bootstrappable scheme.

4.5.2 Relinearization

Recall from Section 4.4 that in the somewhat homomorphic LTV scheme, we find it necessary that in order to decrypt a product of d ciphertexts, we will need to multiply c_{mult} by f^d. We can easily see that is undesirable as with added multiplications our error terms grow exponentially.

We utilize relinearization to grant us the power to decrypt any combination of ciphertexts using only one multiplication of f. In order to do this, we introduce an evaluation phase into our encryption scheme which is now described as follows.

[1]Recall, in Section 3.1 that we define $\mathcal{C}$ to be the ring of ciphertexts created by the encryption scheme $\mathcal{E}$.

Key Creation

Again, let $\phi(x) = x^n + 1$ for some $n \in \mathbb{N}$, let $R = \mathbb{Z}[x]/\langle \phi(x) \rangle$ and let χ be a probability distribution that is B-bounded, where $B \in \mathbb{Z}$. We will also define the ring $R_q = R/qR$.

First we will sample g, $u \leftarrow \chi$ and set $f := 2u+1$ so that $f \equiv 1(mod 2)$. If f is not invertible in the ring R_q, resample u. Let $h := 2gf^{-1} \in R_q$, and set

$$pk := (h, q) \quad \text{and} \quad sk := (f, q)$$

as our public and secret keys, respectively.

For $\tau \in \{0, ..., \lfloor \log q_{i-1} \rfloor\}$, sample s_τ, $e_\tau \leftarrow \chi$ and compute

$$\gamma_\tau := hs_\tau + 2e_\tau + 2^\tau f \in R_q.$$

Define the evaluation key ek as

$$ek := \left\{\gamma_0, \gamma_{\lfloor \log q \rfloor}\right\} \in R_q^{\lfloor \log q \rfloor}.$$

Encryption

Sample s, $e \leftarrow \chi$. Output the ciphertext

$$c := hs + 2e + m \in R_q$$

to encrypt our single bit message $m \in \{0, 1\}$.

Decryption

Given c and a secret key $sk = (f, q)$, we can decrypt the ciphertext by computing

$$\mu := fc \in R_q.$$

We must then compute

$$m' := \mu \pmod 2.$$

We will output m' as the decrypted message.

Evaluation

We show how to evaluate a t-input circuit C. We assume w.l.o.g. that the circuit C is leveled in that it is composed of alternating XOR and AND levels.

We show how to homomorphically add and multiply two elements in $\{0, 1\}$. Given two ciphertexts c_1 and c_2, defined previously, we assume we also encrypted using a single distinct public key and secret key tuple.

Addition We can continue to do homomorphic addition as we have previous defined, as we can always decrypt any sum of d ciphertexts by utilizng a single f multiplication.

Multiplication Given two ciphertexts $c_1, c_2 \in R_q$, we first let

$$c_0 := c_1 \cdot c_2 \in R_q.$$

Then we can find the binary representation of c_0 by generating $\tilde{c}_\tau$ such that

$$\tilde{c}_0 = \sum_{\tau=0}^{\lfloor \log q \rfloor} \tilde{c}_\tau 2^\tau.$$

Next, we define

$$\tilde{c}_{mult} := \sum_{\tau=0}^{\lfloor \log q \rfloor} \tilde{c}_\tau \gamma_\tau,$$

Which we will output as our ciphertext.

4.5.3 Modulus Reduction

Modulus reduction [BV11a, BV11b] is a noise-management technique that provides an *exponential* gain on the depth of the circuits that can be evaluated, while keeping the depth of the decryption circuit unchanged. Informally, if d_{dec} is the depth of the decryption circuit of the scheme described in Section 3.4, then we modify the scheme so that that its decryption circuit is unchanged but the scheme can now evaluate its own decryption circuit, making it bootstrappable. Details follow.

We let $[\cdot]_q$ denote the corresponding element in R_q (ie. Reducing modulo $\phi(x)$ and q), as in [BV11a]. Then, if (h, f) is a key pair and c is a ciphertext under public key h, $[fe]_q$ corresponds to the noise in c. Recall that for decryption to be successful, we need $[fc]_q$ to be equal to fc over the integers. However, each homomorphic operation increases this noise. Modulus reduction allows us to keep the noise magnitude small by simply scaling the ciphertext after each operation. The key idea is to exploit the difference in how the noise affects security and homomorphisms:

- The growth of the noise upon homomorphic multiplication is governed by the *magnitude* of the noise.
- Security is governed by the *ratio* between the magnitude of the noise and the modulus q.

This suggest a method of reducing the magnitude of the noise and the modulus by roughly the same factor, thus preserving security while at the same time making homomorphic multiplications easier.
In particular, modulus reduction gives us a way to transform a ciphertext $c \in R_{q_1}$ into a different ciphertext $c' \in R_{q_0}$ (where q_0 is smaller than q_1) while preserving correctness: for the secret key f. The transformation from c to c' involves simply scaling by $\frac{q_0}{q_1}$ and rounding appropriately.
Lemma 4.3 (Modulus Reduction, See [BV11a, BV11b])**.** *Let q_0 and q_1 be two odd moduli and let $c \in \mathbb{Z}^n$. Let $c' \in \mathbb{Z}^n$ be the integer vector closest to $\frac{q_1}{q_0} \cdot c$. Then for any $f \in \mathbb{R}^n$, with*

$$|fc \mod q_0| < \frac{q_0}{2} - \frac{q_1}{q_0}\|f\|_1 \tag{27}$$

we have,

$$|fc' \mod q_1| \equiv |fc \mod q_0| \pmod 2 \tag{28}$$

$$|fc' \mod q_1| < \frac{q_1}{q_0}|fc \mod q_0| + \|f\|_1 \tag{29}$$

The beauty of Lemma 4.3 is that if we know the depth d of the circuit we want to evaluate (in our case, $d = d_{dec}$, the depth of the augmented decryption circuit), then we can construct a ladder of decreasing moduli $q_0, ..., q_d$ and preform the modulus reduction after each operation so that at level i all ciphertexts reside in R_{q_i} and the magnitude of the noise at each level is small. In particular, this is true at level d so that once the evaluation is complete, it is possible to decrypt the resulting ciphertext without decryption errors. We will prove this lemma in Section 4.5.3.

Bootstrappable Scheme

We change the scheme so that it uses modulus reduction during ab evaluation phase. Keygen will now sample a ladder of decreasing moduli $q_0, ..., q_{d_{dec}}$, where d_{dec} is the depth of the circuit we are looking to evaluate. We will choose the distribution χ in order to guarantee that any sample is B-bounded, where $B \ll q_{d_{dec}}$. The modified, bootstrappable, scheme is as below.

Key Creation

Again, let $\phi(x) = x^n + 1$ for some $n \in \mathbb{N}$, let $R = \mathbb{Z}[x]/\langle \phi(x) \rangle$ and let χ be a probability distribution that is B-bounded, where $B \ll q_{d_{dec}}$. We will now sample a ladder of decreasing moduli $q_0, ..., q_{d_{dec}}$.

For every $i \in \{0, ..., d_{dec}\}$, sample $g^{(i)}$, $u^{(i)} \leftarrow \chi$ and set $f^{(i)} := 2u^{(i)}+1$ so that $f^{(i)} \equiv 1 (mod 2)$. If $f^{(i)}$ is not invertible in the ring R_{q_i}, resample $u^{(i)}$. Let $h^{(i)} := 2g^{(i)}(f^{(i)})^{-1} \in R_{q_{i-1}}$, and set

$$pk := (h^{(0)}, q_0) \quad \text{and} \quad sk := (f^{(d_{dec})}, q_{d_{dec}})$$

as our public and secret keys, respectively.

For all $i \in [d_{dec}] := \{1, 2, ..., d_{dec}\}$ and $\tau \in \{0, ..., \lfloor \log q_{i-1} \rfloor\}$, sample $s_\tau^{(i)}$, $e_\tau^{(i)} \leftarrow \chi$ and compute

$$\gamma_\tau^{(i)} := h^{(i)} s_\tau^{(i)} + 2e_\tau^{(i)} + 2^\tau f^{(i-1)} \in R_{q_{i-1}} \text{and} \zeta_\tau^{(i)} := h^{(i)} s_\tau^{(i)} + 2e_\tau^{(i)} + 2^\tau (f^{(i-1)})^2 \in R_{q_{i-1}}$$

Define the evaluation key ek as

$$ek := \left\{ \gamma_\tau^{(i)}, \zeta_\tau^{(i)} \right\}_{i \in [d_{dec}], \tau \in \{0, ..., \lfloor \log q_i \rfloor\}}.$$

Encryption

Sample s, $e \leftarrow \chi$. Output the ciphertext

$$c := hs + 2e + m \in R_{q_0}$$

to encrypt our single bit message $m \in \{0, 1\}$.

Decryption

Assume w.l.o.g that $c \in R_{q_{d_{dec}}}$. Given c and a secret key $sk = (f^{(d_{dec})}, q_{d_{dec}})$, we can decrypt the ciphertext by computing

$$\mu := f^{d_{dec}} c \in R_q.$$

We must then compute

$$m' := \mu \pmod 2.$$

We will output m' as the decrypted message.

Evaluation

We show how to evaluate a t-input circuit C. We assume w.l.o.g. that the circuit C is leveled in that it is composed of alternating XOR and AND levels.

We show how to homomorphically add and multiply two elements in $\{0, 1\}$. Given two ciphertexts c_1 and c_2, defined previously, we assume we also encrypted using a single distinct public key and secret key tuple.

Addition Given two ciphertexts $c_1, c_2 \in R_{q_i}$, we first compute $\tilde{c}_0 = c_1 + c_2$. For $\tau \in \{0, ..., \lfloor \log q_i \rfloor\}$, define $\tilde{c}_{0,\tau}$ such that

$$\tilde{c}_0 = \sum_{\tau=0}^{\lfloor \log q_i \rfloor} 2^\tau \tilde{c}_{0,\tau}.$$

Next, we define

$$\tilde{c}_1 := \sum_{\tau=0}^{\lfloor \log q_i \rfloor} \tilde{c}_{0,\tau} \gamma_\tau^{(i)} \in R_{q_i}$$

Finally, reduce the modulus: let c_{add} be the integer vector closest to $(q_{i+1}/q_i) \cdot \tilde{c}_1$ such that $c_{add} \equiv \tilde{c}_1 \pmod 2$. Output $c_{add} \in R_{q_{i+1}}$ as an encryption of the *sum* of the underlying messages.

Multiplication Given two ciphertexts $c_1, c_2 \in R_{q_i}$, we first compute $\tilde{c}_0 = c_1 \cdot c_2$. For $\tau \in \{0, ..., \lfloor \log q_i \rfloor\}$, define $\tilde{c}_{0,\tau}$ such that

$$\tilde{c}_0 = \sum_{\tau=0}^{\lfloor \log q_i \rfloor} 2^\tau \tilde{c}_{0,\tau}.$$

Next, we define

$$\tilde{c}_1 := \sum_{\tau=0}^{\lfloor \log q_i \rfloor} \tilde{c}_{0,\tau} \zeta_\tau^{(i+1)} \in R_{q_i}$$

Finally, reduce the modulus: let c_{mult} be the integer vector closest to $(q_{i+1}/q_i) \cdot \tilde{c}_1$ such that $c_{add} \equiv \tilde{c}_1 \pmod 2$. Output $c_{add} \in R_{q_{i+1}}$ as an encryption of the *sum* of the underlying messages.

Proof of Modulus Reduction

We will now present the proof that modulus reduction does indeed work. Recall the definition of modulus reduction from Lemma 4.3 in Section 4.5.3.

Proof. We can write

$$fc \mod q_0 = fc - kq_0 \tag{30}$$

for some $k \in \mathbb{Z}$. We will define

$$n_1 = fc' - kq_1 \tag{31}$$

From (30) and (31)

$$c \equiv c' \pmod 2 \text{and} q_0 \equiv q_1 \pmod 2.$$

So

$$|fc \mod q_0| \equiv n_1 \pmod 2,$$

moreover

$$|fc \mod q_0| \equiv |n_1| \pmod 2.$$

We need to prove that $n_1 = fc' \mod q$. Since $fc = (fc \mod q_0) + kq_0$, we have

$$\frac{q_1}{q_0} fc = \frac{q_1}{q_0}(fc \mod q) + fc' - \frac{q_1}{q_0}(fc).$$

So, we may write

$$\begin{aligned} n_1 &= fc' - kq_1 \\ &= \frac{q_1}{q_0}(fc \mod q_0) + fc' - \frac{q_1}{q_0}(fc) \\ &= \frac{q_1}{q_0}(fc \mod q_0) + fd \end{aligned}$$

where $d = c' - \frac{q_1}{q_0}c$.

Since for any element $e \in d$ we know $e \in [-1, 1]$,

$$|fd| < \|f\|_1$$

So by the triangle inequality, we have

$$|n_1| < \frac{q_1}{q_0}|fc \mod q_0| + \|f\|_1.$$

Therefore by hypothesis, we have

$$\begin{aligned} |fc \mod q_0| &< q_0/2 - \frac{q_0}{q_1}\|f\|_1 \\ \frac{q_1}{q_0}|fc \mod q_0| &< q_1/2 - |f|_1 \\ |n_1| &< \frac{q_1}{2}. \end{aligned}$$

Thus, n_1 must be closer to zero than any other integer i satisfying $fc' - iq_1$. So,

$$n_1 = fc' \mod q_1 \text{ and } |fc' \mod q_1| < \frac{q_1}{q_0}|fc \mod q_0| + \|f\|_1.$$

□

Thus showing we can recover the same message in two distinct rings defined as

$$R_{q_0} := R/q_0R \text{ and } R_{q_1} := R/q_1R.$$

after modulus reduction.

5 Conclusions

When published there were many oversights presented in the context of the LTV scheme and we successfully presented a rewrite of the FHE LTV scheme which utilizes the traditional single-key approach. Simplifying notation and filling in gaps, this rewrite will make the information portrayed in the LTV scheme accessible to non-experts as well as serve as proof that many of the claims in the original LTV paper were justified.

Based off of the NTRU cryptosystem, developed by Hoffstein, Pipher and Silverman, our scheme utilizes ring theory and sampling from elementary probability theory to create a somewhat homomorphic system. Whats more, as we have shown, the scheme supports homomorphic operations utilizing Gentrys bootstapping theorem and modulus reduction and relinearization.

The authors' construction is particularly interesting since the NTRU scheme was originally proposed as an efficient public-key encryption scheme, meant to replace RSA and elliptic curve cryptosystems in applications where computational efficiency is at a premium. Although the transform to a fully homomorphic system deteriorates the efficiency of NTRU slightly, we believe that this system is a leading candidate for a practical FHE scheme.

References

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *Electronic Colloquium on Computational Complexity (ECCC)*, pages 109–109, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Proceedings of the 31st annual conference on Advances in cryptology*, CRYPTO'11, pages 505–524, Berlin, Heidelberg, 2011. Springer-Verlag.

[Gal94] J.A. Gallian. *Contemporary Abstract Algebra: Answer Key*. D. C. Heath Canada, Limited, 1994.

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, Stanford, CA, USA, 2009. AAI3382729.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In *Lecture Notes in Computer Science*, pages 267–288. Springer-Verlag, 1998.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 1219–1234, New York, NY, USA, 2012. ACM.

[Mei12] Rebbeca Meissen. A mathematical approach to fully homomorphic encryption. MQP Final Report, April 2012.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

[RSA83] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26(1):96–99, January 1983.

[SS11] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In *Proceedings of the 30th Annual international conference*

on Theory and applications of cryptographic techniques: advances in cryptology, EUROCRYPT'11, pages 27–47, Berlin, Heidelberg, 2011. Springer-Verlag.