NTRU-ν-um: Secure Fully Homomorphic Encryption from NTRU with Small Modulus

Kamil Kluczniak
CISPA Helmholtz Center for Information Security
`kamil.kluczniak@cispa.de`

September 20, 2022

Abstract

NTRUEncrypt is one of the first lattice-based encryption schemes. Furthermore, the earliest fully homomorphic encryption (FHE) schemes rely on the NTRU problem. Currently, NTRU is one of the leading candidates in the NIST post-quantum standardization competition. What makes NTRU appealing is the age of the cryptosystem and relatively good performance.

Unfortunately, FHE based on NTRU became impractical due to efficient attacks on NTRU instantiations with "overstretched" modulus. In particular, currently, NTRU-based FHE schemes to support a reasonable circuit depth require instantiating NTRU with a very large modulus. Breaking the NTRU problem for such large moduli turns out to be easy. Due to these attacks, any serious work on practical NTRU-based FHE essentially stopped.

In this paper, we reactivate research on practical FHE that can be based on NTRU. We design an efficient bootstrapping scheme in which the noise growth is small enough to keep the modulus to dimension ratio relatively small, thus avoiding the negative consequences of "overstretching" the modulus. Our bootstrapping algorithm is an accumulator-type bootstrapping scheme analogous to AP/FHEW/TFHE. Finally, we show that we can use the bootstrapping procedure to compute any function over $\mathbb{Z}_t$. Consequently, we obtain one of the fastest FHE bootstrapping schemes able to compute any function over elements of a finite field alongside reducing the error.

1 Introduction

A fully homomorphic encryption scheme gives the possibility to compute any function on encrypted data. Early practical homomorphic encryption schemes were built either from the ring learning with errors problem (e.g. BGV [19, 18] and BFV [17, 34]) or the NTRU problem[1] (LTV [54] and YASHE [14]). Both variants demonstrated similar performance characteristics [30]. It is worth noting that NTRUEncrypt by Hoffstein, Pipher, and Silverman [45] was among the first lattice-based cryptosystems, is currently subject to standardization [1, 9] and considered to be a leading candidate for further standards [4].

The first subfield attack against NTRU was due to Gentry and Szydlo [39] and was directed against the NTRU signature scheme. However, the attack did not get much attention since the original NTRU encryption algorithm does not require a large modulus. Further, Albrecht, Bai, and

[1] The problem is called "Decisional Small Polynomial Ratio Assumption" but here we refer to it briefly as NTRU.

Ducas [5] and independently Cheon, Jeong, and Lee [24] apply the subfield attack to, among other, LTV [54] and YASHE [14].

Roughly speaking, the NTRU lattice contains a sublattice that, when recovered, allows an attacker to recover the secret key almost immediately. Therefore, when the modulus of an NTRU is too large in comparison to the dimension, then NTRU is broken. Kirchner and Fouque [48] later studied the attack and showed that finding the basis vector of the sublattice is faster than recovering the secret key already for moduli as small as $Q = n^{2.783+o(1)}$. The same attack does not apply to ring learning with errors. To support correct computation, all schemes BGV, BFV, LTV, and YASHE need to increase the modulus with the depth of the circuit unless we bootstrap the ciphertext, which in itself is a costly operation. Since for larger moduli, NTRU is broken, to compensate, we would need to increase its dimension, making NTRU-based fully homomorphic encryption schemes uncompetitive to RLWE-based schemes. Very recently, Ducas and van Woerden [33] gave a detailed analysis and estimations, backed by experiments, on the hardness of the NTRU problem when the modulus falls into the overstretched regime.

1.1 Our Contribution.

We leverage the results from Ducas and van Woerden [33] and design a very competitive, fully homomorphic encryption scheme based on NTRU that we call NTRU-ν-um[2]. Importantly, we can keep the modulus of the NTRU instantiation small, and thereby we get reasonable security levels. What is more, we can leverage a larger ring dimension to our advantage. At the core of our scheme is a bootstrapping algorithm, which is based on the homomorphic accumulator technique [8, 32]. To build the bootstrapping algorithm, we construct an NTRU-analog of the GSW encryption scheme by Gentry, Sahai, and Waters [38]. The GSW encryption scheme underlies many previous accumulator-based bootstrapping schemes [8, 32, 26]. In the fastest bootstrapping schemes [32, 26] GSW is instantiated with the ring version of the learning with errors problem. In the case of NTRU, multiplication requires roughly half the work as an instantiation of GSW with ring learning with errors.

Our bootstrapping algorithm, alongside reducing the error, can compute all negacyclic functions $F : \mathbb{Z}_t \mapsto \mathbb{Z}_t$ where $t \in \mathbb{N}$. Using recent techniques from [61, 53] we extend the method to all functions over $\mathbb{Z}_t$. In other words, we can compute arbitrary functions over finite fields alongside bootstrapping the ciphertext and reducing the error. This way, we can leverage a larger ring dimension to perform computation alongside bootstrapping of higher precision. Very recently such full domain functional bootstrapping got more attention [29, 50, 61, 53]. Our bootstrapping algorithm is arguably the fastest among the currently proposed schemes. We give more details on the comparison in the main body of the paper, but in practice, our scheme is roughly two times as fast as the current best schemes.

We show several parameter sets to correctly bootstrap plaintexts from $\mathbb{Z}_t$ where $t \approx 2^8$ or more. One appealing property of our scheme is that we can run in the correct mode and approximate mode. This means, in particular, that we can set the plaintext space even as high as $\log_2 t = 2^{14}$ if the application can tolerate errors. We give an efficient implementation and test a few applications. For example, our bootstrapping allows us to compute univariate polynomials in time independent of the degree of the polynomial. Importantly, our scheme can compute modular inversion of field elements with only a single bootstrapping operation. Consequently, we show applications to solving systems of linear equation over encrypted data by evaluating Gaussian elimination. To the best of our knowledge, this is the first time finite field Gaussian elimination has been efficiently performed over encrypted data. Solving such equations may be useful to build, for example, blind signatures

[2]Read NTRU-nium.

by combining our scheme with multivariate blind signatures like Rainbow [58, 47, 31] (Round 3 candidate in the NIST PQ Competition). In contrast, when applying fully homomorphic encryption for boolean circuits [32, 26, 27], we would need to represent modular reduction and modular inversion as a boolean circuit. For schemes designed to compute arithmetic circuits [17, 19, 18, 34], we still need to represent modular inversion as an arithmetic circuit. While it is theoretically possible to compute such circuits, it is infeasible to apply these schemes, for example, to Gaussian elimination. Finally, we can implement binary decomposition of field elements with only a single bootstrapping, thereby we can efficiently switch between binary and arithmetic homomorphic computation.

1.2 Overview of NTRU-ν-um's Bootstrapping Algorithm.

Let us first start by recalling the structure of NTRU samples and introducing a gadget NTRU version. Denote $\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1)$. An NTRU sample is a polynomial $c \in \mathcal{R}_Q$ of the form $c = e_1/f + e_2 + m$, where $f \in \mathcal{R}_Q$ (usually having coefficients in $\{-1, 0, 1\}$) is the secret key, $e_1, e_2 \in \mathcal{R}_Q$ are the error polynomials and have coefficients from some distribution $\mathcal{X}$, and $m = \frac{Q}{t} \cdot m'$ with $m' \in \mathcal{R}_t$. Note that if we want to add two NTRU ciphertexts c, and $c' = e'_1/f + e'_2 + m'$, we simply compute $c + c' = (e_1 + e'_1)/f + e_2 + e'_2 + m + m'$ which is a valid ciphertext of $m + m'$ but with larger error. We can also multiply a ciphertext by a scalar $a \in \mathcal{R}_Q$ such that $c'' = c \cdot a = e_1 \cdot a/f + e_2 \cdot a + m \cdot a$.

Note that the above scalar multiplication is quite expensive as the error terms are multiplied by the scalar a. Hence, to preserve correctness and allow for decryption, we can only multiply with sparse polynomials with small coefficients . To resolve the issue, we introduce a gadget version of NTRU. Gadget NTRU is analogous to the GSW scheme for ring LWE, but we adapt the GSW technique to NTRU. In this paper, we use gadget NTRU to multiply two ciphertexts and use the fact that the resulting error is relatively small. A gadget NTRU sample is a vector $\mathbf{c}_G = [c_i]_{i=1}^{\ell}$ with $\ell = \log_{\mathsf{L}}(Q)$, where each c_i is a NTRU ciphertext of $m_G \cdot \mathsf{L}^{i-1}$. To multiply such ciphertext with a scalar $c \in \mathcal{R}_Q$, we compute the inner product between $\mathbf{c}_G$ and the decomposition of c with respect to the base L. Concretely, let G^{-1} be the decomposition function such that $\mathbf{c}_D \leftarrow \mathsf{G}^{-1}(c, \mathsf{L}) \in \mathcal{R}_{\mathsf{L}}^{\ell}$ where $\sum_{i=1}^{\ell} \mathbf{c}_D[i] \cdot \mathsf{L}^{i-1} = c$. Then to multiply a gadget NTRU ciphertext $\mathbf{c}_G$ with c, we compute

$$c_{\mathsf{out}} = \left\langle \mathbf{c}_G, \mathsf{G}^{-1}(c, \mathsf{L}) \right\rangle = e_{1,G}/f + e_{2,G} + m_G \cdot c$$

Note that when computing the inner product, we make ℓ scalar multiplications and additions, where the scalar multiplications are with polynomials from $\mathcal{R}_{\mathsf{L}}$. Furthermore, if c is itself an NTRU ciphertext as above, then we have

$$\begin{aligned} c_{\mathsf{out}} &= e_{1,G}/f + e_{2,G} + m_G \cdot (e_1/f + e_2 + m) \\ &= (e_{1,G} + m_G \cdot e_1)/f + (e_{2,G} + m_G \cdot e_2) + m_G \cdot m, \end{aligned}$$

which is a valid NTRU ciphertext. Note that the error, in this case, depends on the magnitude of m_G.

Blind Rotating a Homomorphic Accumulator. Following the ideas from [8, 32, 26], we construct a homomorphic accumulator scheme which we can informally summarize as follows. A LWE sample is a vector $\mathbf{c} \in \mathbb{Z}_{2N}^{n+1}$, where $\mathbf{c}[1] = -\mathbf{c}[2{:}n+1]^{\top}\mathbf{s} + e + \frac{2N}{t}m$. To partially decrypt $\mathbf{c}$ it is sufficient to compute the linear function $\mathbf{c}[1] + \mathbf{c}[2{:}n+1]^{\top}\mathbf{s} = e + \frac{2N}{t} \cdot m$. Given that the error $e < \frac{2N}{2 \cdot t}$, we can further decode the message by $\left\lfloor \frac{t}{2N}(e + \frac{N}{t}m) \right\rceil = m$. Note that each message is encoded in an interval of size $\left\lfloor \frac{2N}{t} \right\rfloor$ to ensure correct decryption.

Now let us consider the operation $a_{\mathsf{rot}} \cdot X^{\mathbf{c}[1] + \mathbf{c}[2:n+1]^{\top}\mathbf{s}} = a_{\mathsf{rot}} \cdot X^{e + \frac{2N}{t} \cdot m} \in \mathcal{R}_Q$. Note that when $\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1)$, this operation is a negacyclic rotation of the coefficients of a_{rot} by

$\mathbf{c}[1] + \mathbf{c}[2{:}n+1]^\top \mathbf{s} = e + \frac{2N}{t} \cdot m \mod N$ positions. Hence, the idea is to set the coefficients of the polynomial a_{rot} such that after rotating it, the desired value for $e + \frac{2N}{t} \cdot m$ is encoded in the constant coefficient. Specifically, to compute any negacyclic function $F : \mathbb{Z}_t \mapsto \mathbb{Z}_t$, we set the rotation polynomial such that $a_{\mathsf{rot}}[y+1] = F(\left\lfloor \frac{t}{2N} \cdot y \right\rceil)$ for all $y \in [0, N]$. Remind that if $y = \frac{2N}{t} \cdot m + e$, where $m \in \mathbb{Z}_t$, then $\left\lfloor \frac{t}{N} \cdot y \right\rceil = m$ given that $e \leq \frac{N}{t}$.

The Bootstrapping Procedure. Now we are ready to describe the bootstrapping procedure. Let's say that we want to bootstrap an LWE ciphertext with a secret key $\mathbf{s} \in \{0,1\}^n$. We publish n gadget NTRU ciphertexts that encrypt the bits of the LWE secret key. Denote the vector of those gadget NTRU ciphertexts by $\mathbf{c}_{\mathsf{Bk}}$. Furthermore, we have an NTRU ciphertext c_{acc} that encodes a_{rot}. We call c_{acc} the accumulator. To bootstrap an LWE ciphertext $\mathbf{c}$ we compute

$$\begin{aligned} c_{\mathsf{out}} &= c_{\mathsf{acc}} \cdot X^{\mathbf{c}[1]} \cdot \prod_{i=1}^{n} X^{\mathbf{c}[i+1]} \cdot \mathbf{c}_{\mathsf{Bk}}[i] \\ &= c'_{\mathsf{acc}} \cdot X^{\mathbf{c}[2:n+1]^\top \mathbf{s}}. \end{aligned}$$

where c'_{acc} encrypts a_{rot} just as c_{acc} but with a higher error. Finally, we have that the message in c_{out} contains the desired result in its constant coefficient.

The problem now is that if we want to continue computing and bootstrapping on the resulting ciphertext, we need to transform the NTRU ciphertext into an LWE ciphertext which encrypts the message in the constant coefficient of the NTRU ciphertext. Hence we design a special key switching procedure that homomorphically extracts the dth coefficient, by computing the linear function

$$(c_{\mathsf{acc}} \cdot f)[d] = \sum_{\substack{i=1, j=1, \\ i+j-2 \ mod \ N = d}}^{N} c_{\mathsf{acc}}[i] \cdot f[i]$$

from the coefficient of the NTRU ciphertext and its secret key. Furthermore, we use the NTRU to LWE key switching procedure to pack N messages into a single NTRU ciphertext which we can then extract for bootstrapping. This way, we transmit only a single integer per message.

Note, however, that there is a problem with this solution. Namely, when computing $c_{\mathsf{acc}} \cdot f$ we obtain and encryption of $m \cdot f$ instead of m. In other words, we have the message masked by the secret key f. So how can we possibly continue to bootstrap such ciphertext? We solve this issue, by including $f^{-1} \in \mathcal{R}_Q$ in the accumulator c_{acc}. That is the accumulator will encrypt $f^{-1} \cdot a_{\mathsf{rot}}$. When multiplying f we immediately recover a_{rot} (or the negacyclic rotation of a_{rot}). Unfortunately, the trick requires us to assume NTRU is key-dependent message (KDM) secure with respect to f^{-1}. While we do not have a formal reduction, we believe that this version preserves security as we can write such NTRU samples as $c = e_1/f + e_2 + m/f = (e_1 + m)/f + e_2$. In this case, the constant coefficient of the e_1 error is shifted by m. If coefficients of e_1 are random variables with expectations equal to zero, then the KDM version shifts the expectation by the coefficients of m.

Another problem appears when using such a scheme in practice. Namely, since we require the message in the accumulator to be key-dependent, an evaluator cannot freely choose rotation polynomials, and we need to publish all potential accumulators together with the bootstrapping key. To resolve this issue, instead of publishing an accumulator with the rotation polynomials, we can publish a gadget NTRU encryption of $\frac{Q}{t} \cdot f^{-1}$. The evaluator can then choose its own a_{rot} and multiply it with the accumulator. Note that if plaintexts are $\mathbb{Z}_t$, then $a_{\mathsf{rot}} \in \mathcal{R}_t$ and $t << Q$. Hence we actually need to publish a smaller gadget that supports the composition of numbers up to t instead of Q.

1.3 Related Work

Gentry's introduction of the bootstrapping technique [37] opened a floodgate of research on fully homomorphic encryption [19, 17, 34, 18, 7, 38, 41, 22, 42].

The NTRU problem and the corresponding cryptosystem dates back to the work by Hoffstein, Pipher, and Silverman [45]. One of the earliest schemes by López-Alt, Tromer, and Vaikuntanathan [54], and its scale-invariant version YASHE [14] are based on the Stehlé, and Steinfeld's [60] variant of the NTRU problem.

The first accumulator-based bootstrapping scheme is due to Alperin-Sheriff, and Peikert [8]. The techniques require representing the decryption circuit as an arithmetic circuit, and we do not rely on Barrington's theorem. Furthermore, the method exploits error characteristics of the GSW cryptosystem by Gentry, Sahai, and Waters [38]. Hiromasa, Abe, and Okamoto [44] improved upon [8] and build a version of GSW that natively encrypts matrices. Genise et al. [36] showed an encryption scheme that further improves the efficiency of matrix operations, albeit using a novel NTRU-like assumption. Ducas and Miccancio [32], building on [8], design a practical bootstrapping algorithm called FHEW. FHEW uses the ring version of the GSW cryptosystem. Chillotti et al. [26, 28] showed numerous optimizations to FHEW bootstrapping algorithm. On the other hand, their scheme called TFHE relies on LWE with binary keys, while FHEW was originally designed to support keys with much larger coefficients. We refer to the work by Micciancio and Polyakov [56] for an excellent comparison of both methods. The FHEW and TFHE bootstrapping algorithms, by far, are the fastest bootstrapping algorithms to date. Further improvements mostly relied on incorporating packing techniques [27, 57], and improved lookup tables evaluation [27, 20]. Initially, FHEW/TFHE were designed to bootstrap ciphertexts with binary plaintexts, but a series of works [12, 20, 40] showed that extending the computation to larger plaintexts may be beneficial in practice.

Concurrently, Chillotti et al. [29] and Kluczniak and Schild [50], who was very quickly followed by Yang et al. [61] and Liu et al. [53], showed how to resolve the limitation of the FHEW/TFHE functional/programmable bootstrap. In particular, while previous schemes could bootstrap larger plaintexts, due to the negacyclicity of the function that the bootstrapping could compute, it wasn't easy to compute arithmetic circuits over $\mathbb{Z}_p$. The works [29, 50, 61, 53] resolve the issue by using FHEW/TFHE as a subroutine. Still, the resulting bootstrapping algorithms are inherently slower than the original TFHE algorithm, and so far, only [50, 61, 53] implemented their schemes.

Concurrent and Independent Work. We note that Bonte et al. [13] independently published a fully homomorphic encryption scheme similar to ours. In particular, they also define a gadget NTRU cryptosystem and build an accumulator bootstrapping algorithm. We note that there are several differences in our designs. The most crucial difference seems to be that Bonte et al. build their scheme with binary ciphertexts in mind while we compute arbitrary functions on plaintexts from $\mathbb{Z}_t$. There are also some very technical differences, like the way both works extract LWE ciphertexts. We describe a generalized algorithm that we can later use to extract LWE samples from the packed NTRU ciphertexts. Bonte et al. [13] show a faster blind rotation algorithm for ternary keys. We note that the optimization is general and can be used with our algorithm as well. Finally, we note that Bonte et al. [13] need to reduce the security of their scheme to a less understood version of the decisional small polynomial ration (also called NTRU) assumption. In contrast, we can reduce the security of our scheme to standard NTRU and RLWE. We describe more details on this in Section 5.

2 Preliminaries

Notation. We denote as $\mathcal{R}$ the ring of polynomials $\mathbb{Z}_Q[X]/(X^N + 1)$ where N is prime. We denote vectors with bold lowercase letters, e.g., $\mathbf{v}$. We denote a n dimensional column vector as

$[f(.,i)]_{i=1}^{n}$, where $f(.,i)$ defines the i-th coordinate. For brevity, we will also denote as $[n]$ the vector $[i]_{i=1}^{n}$, and more generally $[n,m]_{i=n}^{m}$ the vector $[n,\ldots,m]^\top$. We address the ith entry of a vector $\mathbf{v}$ by $\mathbf{v}[i]$, and denote a slice of the vector by $\mathbf{v}[i{:}j]$. In particular, if $\mathbf{v} = [v_1, v_2, \ldots, v_m]$, then $\mathbf{v}[i{:}j] = [v_i, v_{i+1}, \ldots, v_j]$. For a random variable $a \in \mathbb{Z}$ we denote as $\mathsf{Var}(a)$ the variance of a and as $\mathsf{E}(x)$ its expectation. For $a \in \mathcal{R}_Q$, we define $\mathsf{Var}(a)$ and $\mathsf{E}(a)$ to be the variance and expectation respectively of the coefficients of the polynomial a. By $\mathsf{Ha}(\mathbf{a})$ we denote the hamming weight of the vector $\mathbf{a}$, i.e., the number of of non-zero coordinates of $\mathbf{a}$. We represent numbers in $\mathbb{Z}_Q$ as integers in $[-Q/2, Q/2)$.

At Table 1 we list commonly used parameters. Throughout the paper, we denote as $Q, P \in \mathbb{N}$ to be moduli. The parameter $n \in \mathbb{N}$ always denotes the dimension of an LWE sample. For rings, we always use N to denote the degree of $(X^N + 1)$. We define $\ell = \lceil \log_{\mathsf{L}} Q \rceil$ for some decomposition base $\mathsf{L} \in \mathbb{N}$. We define the decomposition algorithm $\mathbf{a} = \mathsf{G}^{-1}(c, \mathsf{L})$ to take a ring element c, a decomposition base L and output a vector $\mathbf{a} \in \mathcal{R}_{\mathsf{L}}$ with coefficients in $[-\mathsf{L}/2, \mathsf{L}/2)$ such that $c = \sum_{k=1}^{\ell} \mathbf{a}[i] \cdot \mathsf{L}^{k-1}$. Finally, we refer to random variables from the discrete Gaussian distribution with parameter (standard deviation) σ. Remind that the variance of discrete Gaussian random variables is σ^2.

Assumptions. Below we recall the learning with errors assumption by Regev [59] and recall the error analysis for its linear homomorphism.

Definition 1 (Learning With Errors) *Let $\mathbf{s} \in \mathcal{X}_{\mathsf{sk}}$ be a secret key, for a secret key distribution $\mathcal{X}_{\mathsf{sk}}$ over and and $e \in \mathbb{N}$ be form the discrete Gaussian distribution of parameter σ. We define a LWE sample of a message $m \in \mathbb{Z}_Q$ as $\mathbf{c} = \mathsf{LWE}_{\sigma}(\mathbf{s}, m) \in \mathbb{Z}_Q$ where $\mathbf{c}[1] = -\mathbf{c}[2{:}n+1]^\top \cdot \mathbf{s} + e + m \in \mathbb{Z}_Q$, and $\mathbf{c}[2{:}n+1]$ is a vector that is chosen from the uniform distribution over $\mathbb{Z}_Q$. We define the phase of $\mathbf{c}$ as $\mathsf{Phase}(\mathbf{c}) = \mathbf{c}[1] + \mathbf{c}[2{:}n+1]^\top \cdot \mathbf{s}$. We define the learning with error distribution $\mathsf{LWE}_{n,\mathcal{X}_{\mathsf{sk}},\sigma}$, to consist of LWE samples of zero, as defined above.*

The learning with error assumption states that it is hard to distinguish elements sampled from $\mathsf{LWE}_{n,\mathcal{X}_{\mathsf{sk}},\sigma}$ and elements sampled uniformly at random over $\mathbb{Z}_Q^{n+1}$.

It is well know that the following holds.

Lemma 1 (Linear Homomorphism of LWE samples) *Let $\mathbf{c} = \mathsf{LWE}_{\sigma_{\mathbf{c}}}(\mathbf{s}, m_{\mathbf{c}})$ and $\mathbf{d} = \mathsf{LWE}_{\sigma_{\mathbf{d}}}(\mathbf{s}, m_{\mathbf{d}})$. If $\mathbf{c}_{\mathsf{out}} \leftarrow \mathbf{c} + \mathbf{d}$, then $\mathbf{c}_{\mathsf{out}} \in \mathsf{LWE}_{\sigma_{\mathsf{out}}}(\mathbf{s}, m_{\mathbf{c}} + m_{\mathbf{d}})$, where $\sigma_{\mathsf{out}}^2 = \sigma_{\mathbf{c}} + \sigma_{\mathbf{d}}$. Furthermore, let $d \in (-\mathsf{L}/2, \mathsf{L}/2)$. If $\mathbf{c}_{\mathsf{out}} \leftarrow \mathbf{c} \cdot d$, then $\mathbf{c}_{\mathsf{out}} \in \mathsf{LWE}_{\sigma_{\mathsf{out}}}(\mathbf{s}, m_{\mathbf{c}} \cdot d)$, where $\sigma_{\mathsf{out}}^2 \leq \frac{\mathsf{L}^2}{4} \cdot \sigma_{\mathbf{c}}^2$.*

Proof 1 *(Sketch) The proof follows from the elementary calculus of random variables. The noise variance of the sum of the LWE ciphertexts follows form the sum of two discrete Gaussian random variables. The bound on the noise variance for scalar multiplication follows from the fact that we bound d by $\mathsf{L}/2$ and that the noise of the LWE sample is centered around zero. Remind that $\mathsf{Var}(B \cdot e) = B^2 \cdot \mathsf{Var}(e)$, where e has Gaussian parameter σ.*

Now we recall the decisional small polynomial ratio assumption [45] (or NTRU assumption).

Definition 2 (Decisional Small Polynomial Ratio Assumption) *Let $f \in \mathcal{R}_Q$ be a secret key with coefficients samples uniformly form the ternary distribution, i.e., from $\{-1, 0, 1\}$ conditioned to have an inverse in $\mathcal{R}_Q$. We define the distribution $\mathsf{DSPRA}_{\mathcal{R}_Q}$ to be the distribution of elements $[g_i/f_i]_{i=1}^{m}$, where f_i is from the same distribution as f, and $g_i \in \mathcal{R}_Q$ has coefficients chosen form the uniform ternary distribution but must not necessarily be invertible in $\mathcal{R}_Q$. The decisional small polynomial ratio assumption says it is hard to distinguish elements sampled from $\mathsf{DSPRA}_{\mathcal{R}_Q}$ and elements sampled uniformly from $\mathcal{R}_Q$.*

Q, P	Prime moduli such that $Q < P$.
$\mathcal{R}_Q, \mathcal{R}_P$	$\mathbb{Z}_Q[X]/(X^N+1)$ and $\mathbb{Z}_P[X]/(X^N+1)$
n	LWE dimension
$\mathsf{Bk}, \mathsf{Ksk}$	Blind rotation and key-switching keys
L_{Bk}	Decomposition base for the blind rotation key where $\ell_{\mathsf{Bk}} = \lceil \log_{\mathsf{L}_{\mathsf{Bk}}}(P) \rceil$
$\mathsf{L}_{\mathsf{Ksk}}$	Decomposition base for the key switching key where $\ell_{\mathsf{Ksk}} = \lceil \log_{\mathsf{L}_{\mathsf{Ksk}}}(Q) \rceil$
f	NTRU secret key in $\mathcal{R}_P$
s	LWE secret key in $\mathcal{R}_Q$
$\sigma_{\mathsf{Bk}}, \sigma_{\mathsf{Ksk}}, \sigma_{\mathsf{acc}}$	Standard deviations of the noise terms in the bootstrapping key, key-switching key, and the ciphertext c_{acc} respectively.
a_{rot}	$a_{\mathsf{rot}} \in \mathcal{R}_P$ s.t. $(a_{\mathsf{rot}} \cdot X^y)[1] = F(\lfloor \frac{t_1}{N} \cdot y \rceil)$, where $y \in \mathbb{Z}_N$ and $F : \mathbb{Z}_{t_1} \mapsto \mathbb{Z}_{t_2}$.

Table 1: Summary of Variables.

Stehlé and Steinfeld [60] showed that the assumption holds unconditionally for a certain choice of parameters. Furthermore, they showed a reduction to worst-case lattice problems when for the ring $\mathbb{Z}_Q[X]/(X^N+1)$, where N is a power of two. Homomorphic encryption schemes like LTV [54] and YASHE [14] follow Stehlé and Steinfeld's version of the assumption.

3 Homomorphic Encryption Techniques from NTRU

This section describes the algorithms that we use to build the bootstrapping algorithm. Below we recall the basic cryptosystem. First of all, we describe the algorithm as a symmetric key cryptosystem. Specifically, we do not define a public key version, as it is unnecessary in this paper and simplifies the exposition.

Definition 3 (An NTRU Homomorphic Encryption Scheme) *Let the message modulus as be $t < Q$. Let the secret key f have coefficients from the uniform ternary distribution and have an inverse in $\mathcal{R}_Q$. We define an NTRU encryption as $c = \mathsf{NTRU}_\sigma(f, m) = e_1 \cdot g/f + e_2 + r + \frac{Q}{t} \cdot m$, where e_1, e_2 are random variables over $\mathcal{R}_Q$ with coefficients from the discrete Gaussian distribution of parameter σ, g has coefficients form the uniform ternary distribution, r is in $[-\mathsf{U}(c), \mathsf{U}(c)]$ and $m \in \mathcal{R}_t$. Furthermore, we assume f and g have the same hamming weight. To decrypt we compute $f \cdot m = \lfloor \frac{t}{Q} \cdot c \cdot f \rceil \in \mathcal{R}_t$.*

Note that in this definition we recover $m \cdot f$ instead of m. In this paper, we use a key-dependent version of the scheme, where we encrypt $\frac{Q}{t} \cdot m \cdot f^{-1}$. Therefore, the decryption process cancels the secret key out of the message. Furthermore, we remark that t usually does not divide Q, and in the implementation we round the fraction $\left\lfloor \frac{Q}{t} \cdot m \right\rfloor$ which adds a small rounding error to the r noise. Note that we introduce the notation $\mathsf{U}(c)$ to keep track of the error term that is not from the discrete Gaussian distribution. Usually this error stems from rounding operations. In practice the keys f, g are often generated by choosing an element from $\{-1, 1\}$ uniformly at random and setting random 1/3 coefficients to zero. We follow the same method and set the hamming weights to $2/3 \cdot N$.

Lemma 2 (Correctness of the NTRU Decryption) *Let $c = \mathsf{NTRU}_\sigma\left(f, \frac{Q}{t} \cdot m\right)$. We have that $\frac{Q}{t} \cdot m \cdot f + e + f \cdot r = f \cdot c$, where $\mathsf{Var}(e) \leq 4/3 \cdot N \cdot \sigma^2$ and $E(|f \cdot r|) \leq 2/3 \cdot N \cdot \mathsf{U}(c)$. Furthermore if $|e| < \frac{Q}{2 \cdot t} - |f \cdot r|$, then $\left\lfloor \frac{t}{Q} \cdot c \cdot f \right\rceil = m \cdot f \in \mathbb{Z}_t$.*

Proof 2 *From definition 3 we have that $c = e_1 \cdot g/f + e_2 + r + \frac{Q}{t} \cdot m \in \mathcal{R}_Q$ where $m \in \mathcal{R}_t$. Then $f \cdot c = e_1 \cdot g + f \cdot e_2 + f \cdot r + \frac{Q}{t} \cdot f \cdot m = e + f \cdot r + \frac{Q}{t} \cdot f \cdot m$. Note that $e = e_1 \cdot g + f \cdot e_2$. Hence*

$$\mathsf{Var}(e) = \mathsf{Var}(e_1 \cdot g) + \mathsf{Var}(f \cdot e_2) \leq 2/3 \cdot N \cdot \big(\mathsf{Var}(e_1) + \mathsf{Var}(e_2)\big)$$

Remind that e_1 and e_2 are random variables with Gaussian parameter σ hence their variance is σ^2. Note that we cannot include $f \cdot r$ into the variance as both f and r are from the uniform distributions and not the discrete Gaussian. Hence we upper-bound the expectation of the final noise term. The above follows from e_1, e_2 being independent and centered around zero. Note that in the ring $\mathcal{R}_Q$ multiplication of ring elements corresponds to computing their negacyclic convolutions. Hence the dth coordinate of $f \cdot e_2$ is given by

$$\mathsf{Var}\big((f \cdot e_2)[d]\big) = \sum_{\substack{i=1, j=1, \\ i+j-2 \ mod \ q=d}}^{N} f[i] \cdot e_2[j]),$$

where $\psi(i,j)$ returns 1 if $i+j \geq N$ and -1 otherwise. Therefore if f has $\mathsf{Ha}(f)$ non-zero coefficients and its non-zero are bounded by 1, then we have $\mathsf{Var}\big((f \cdot e_2)[d]\big) \leq 2/3 \cdot N \cdot \mathsf{Var}(e_2)$. The same argument can be made for $\mathsf{Var}(e_1 \cdot g)$. Then we bound the expectation by $E(|f \cdot r|) \leq \mathsf{Ha}(f) \cdot 1 \cdot \mathsf{U}(c)$ where we bound the infinity norm of r by $\mathsf{U}(c)$ and on f is 1. Finally, if $|e| < \frac{Q}{2 \cdot t} - |f \cdot r|$, then

$$\lfloor \frac{t}{Q} \cdot (e + f \cdot r + \frac{Q}{t} \cdot f \cdot m) \rceil = \lfloor \frac{t}{Q} \cdot (e + f \cdot r) + f \cdot m \rceil = f \cdot m$$

because $\frac{t}{Q} \cdot (e + f \cdot r) < \frac{1}{2}$.

Below we analyze the variance of the errors for elementary homomorphic operations.

Lemma 3 (Affine Functions on Encrypted Data) *Let $c_1 = \mathsf{NTRU}_{\sigma_1}(f, m_1)$ and $c_2 = \mathsf{NTRU}_{\sigma_2}(f, m_2)$, and $a \in \mathcal{R}_Q$. We have the following.*
Addition: *We have $c_1 + a = \mathsf{NTRU}_{\sigma_1}(f, m_1 + a)$ and $c_1 + c_2 = \mathsf{NTRU}_{\sigma_{\mathsf{out}}}(f, m_1 + m_2)$, where $\sigma^2_{\mathsf{out}} = \sigma_1^2 + \sigma_2^2$. Furthermore, $\mathsf{U}(c_1 + c_2) \leq \mathsf{U}(c_1) + \mathsf{U}(c_2)$.*
Scalar Multiplication: *We have $c_1 \cdot a = \mathsf{NTRU}_{\sigma_{\mathsf{out}}}(f, m_1 \cdot a)$, where $\sigma^2_{\mathsf{out}} \leq \mathsf{Ha}(a) \cdot ||a||^2_\infty \cdot \sigma_1^2$. Furthermore, $\mathsf{U}(c_1 \cdot a) = \mathsf{U}(c_1) \cdot \mathsf{Ha}(a)$.*

Proof 3 *From definition 3 we have $c_1 = e_1/f + e_2 + r + m_1$ and $c_2 = \tilde{e}_1/f + \tilde{e}_1 + \tilde{r} + m_2$. Addition with scalar a follows trivially from $c_1 + a = e_1/f + e_2 + m_1 + a$. Addition of two ciphertexts follows from $c_1 + c_2 = e_1/f + e_2 + r + m_1 + \tilde{e}_1/f + \tilde{e}_1 + \tilde{r} + m_2 = (e_1 + \tilde{e}_1)/f + e_1 + \tilde{e}_1 + r + \tilde{r} + m_1 + m_2$. Hence the variance of the output noise is the sum of the input noise variances. For scalar multiplication we have $c_1 \cdot a = a \cdot e_1/f_1 + a \cdot e_2 + a \cdot r + a \cdot m_1$. Crucially multiplying ring elements from $\mathcal{R}_Q$ corresponds to computing their negacyclic convolutions. Similarly as in the proof of Lemma 2, we can represent the dth coefficient of $a \cdot e_k$ (or $a \cdot r$) for $k \in [2]$ as a negacyclic convolution and bound its variance by by $\mathsf{Ha}(a) \cdot ||a||^2_\infty \cdot \sigma^2$, and $\mathsf{U}(c \cdot a) \leq \mathsf{U}(c_1) \cdot \mathsf{Ha}(a)$.*

NTRU to LWE Key Switching.

The key switching procedure below takes as input an NTRU ciphertext, and an LWE key switching key outputs an LWE ciphertext that encrypts a chosen coefficient of the NTRU plaintext.

Definition 4 (NTRU to LWE Key Switching) *Let $\mathsf{L}_{\mathsf{Ksk}} \in \mathbb{N}$ be a decomposition parameter for the key-switching procedure and denote $\ell_{\mathsf{Ksk}} = \lceil \log_{\mathsf{L}} Q \rceil$. The NTRU-to-LWE key switching setup $\mathsf{KSSetup}(f, \mathbf{s})$ takes as input f and $\mathbf{s}$, computes and returns the key switching key*

$$\mathsf{Ksk}[i,*] \leftarrow \left[\mathsf{LWE}_{\sigma_{\mathsf{Ksk}}}\left(\mathbf{s}, f[i] \cdot \mathsf{L}_{\mathsf{Ksk}}^{k-1}\right)\right]_{k=1}^{\ell_{\mathsf{Ksk}}} \tag{1}$$

for $i \in [N]$. The key switching procedure $\mathsf{KSwitch}(\mathsf{Ksk}, c, d)$ computes

$$\sum_{\substack{i=1,j=1,\\ i+j-2 \bmod N=d}}^{N} \left\langle \mathsf{Ksk}[i,*], \mathsf{G}^{-1}\left(\psi(i,j) \cdot c[j], \mathsf{L}_{\mathsf{Ksk}}\right)\right\rangle, \tag{2}$$

where $c \in \mathcal{R}_Q$ is a NTRU ciphertext with respect to the secret key f, $d \in [N]$, and $\psi(i,j)$ returns 1 if $i+j \geq N$ and -1 otherwise. Remind that $\mathsf{G}^{-1}\big(\psi(i,j) \cdot c[j], \mathsf{L}_{\mathsf{Ksk}}\big)$ computes the base $\mathsf{L}_{\mathsf{Ksk}}$ decomposition of $\psi(i,j) \cdot c[j] \in \mathbb{Z}_Q$.

Lemma 4 (Correctness of NTRU to LWE Key Switching) *Let $c = \mathsf{NTRU}_\sigma(f, m)$ be a NTRU ciphertext and $\mathsf{Ksk} \leftarrow \mathsf{KSSetup}(f, \mathbf{s})$ as in the formula 1. If $\mathbf{c} = \mathsf{KSwitch}(\mathsf{Ksk}, c, d)$ for $d \in [N]$, then $\mathbf{c}_{\mathsf{out}} = \mathsf{LWE}_{\sigma_{\mathsf{out}}}(\mathbf{s}, (f \cdot m)[d])$, where*

$$\sigma_{\mathsf{out}}^2 \leq \sigma_{\mathsf{Dec}}^2 + N \cdot \ell_{\mathsf{Ksk}} \cdot \mathsf{L}_{\mathsf{Ksk}}^2/4 \cdot \sigma_{\mathsf{Ksk}}^2,$$

where σ_{Dec} and the expectation is as for the noise of c's decryption (see Lemma 2).

Proof 4 *Let us denote a decomposition of $\psi(i,j) \cdot c[j]$ as follows $\big[a_k^{(i,j)}\big]_{k=1}^{\ell_{\mathsf{Ksk}}} = \mathsf{G}^{-1}\Big(\psi(i,j) \cdot c[j], \mathsf{L}_{\mathsf{Ksk}}\Big)$. Note that the decomposed elements $a_k^{(i,j)} \in \big[-\mathsf{L}_{\mathsf{Ksk}}/2, \mathsf{L}_{\mathsf{Ksk}}/2\big)$ are such that $\sum_{k=1}^{\ell_{\mathsf{Ksk}}} a_k^{(i,j)} \cdot \mathsf{L}_{\mathsf{Ksk}}^{k-1} = \psi(i,j) \cdot c[j] \mod Q$.*

Let us first analyze the inner product for each $i \in [N]$ separately. Particularly, we have

$$\begin{aligned}\mathbf{c}_i &= \left\langle \mathsf{Ksk}[i,*], \mathsf{G}^{-1}\left(\psi(i,j) \cdot c[j], \mathsf{L}_{\mathsf{Ksk}}\right)\right\rangle \\ &= \mathsf{LWE}_{\bar{\sigma}}(\mathbf{s}, \sum_{k=1}^{\ell_{\mathsf{Ksk}}} f[i] \cdot \mathsf{L}_{\mathsf{Ksk}}^{k-1} \cdot a_k^{(i,j)}) \\ &= \mathsf{LWE}_{\bar{\sigma}}(\mathbf{s}, f[i] \cdot \psi(i,j) \cdot c[j])\end{aligned}$$

where $\bar{\sigma}^2 \leq \ell_{\mathsf{Ksk}} \cdot \mathsf{L}_{\mathsf{Ksk}}^2/4 \cdot \sigma_{\mathsf{Ksk}}^2$ because we perform ℓ_{Ksk} LWE scalar multiplications.

Then we have that $\mathsf{KSwitch}(\mathsf{Ksk}, c, d)$ computes

$$\begin{aligned}&\sum_{\substack{i=1,j=1,\\ i+j-2 \bmod N=d}}^{N} \mathsf{LWE}_{\bar{\sigma}}\Big(\mathbf{s}, f[i] \cdot \psi(i,j) \cdot c[j]\Big), \\ &\quad = \mathsf{LWE}_{\sigma'}\big(\mathbf{s}, (f \cdot c)[d]\big)\end{aligned}$$

because the sum computes the dth coefficient of the negacyclic convolution of f and c. Then we have

$$\begin{aligned}&\mathsf{LWE}_{\sigma'}\big(\mathbf{s}, (f \cdot c)[d]\big) \\ &\quad = \mathsf{LWE}_{\sigma'}\big(\mathbf{s}, (e_1 \cdot g)[d] + (f \cdot e_2)[d] + (f \cdot r)[d] + (f \cdot m)[d]\big) \\ &\quad = \mathsf{LWE}_{\sigma}\big(\mathbf{s}, (f \cdot m)[d]\big),\end{aligned}$$

where $\sigma'^2 = N \cdot \ell_{\mathsf{Ksk}} \cdot \mathsf{L}_{\mathsf{Ksk}}^2/4 \cdot \sigma_{\mathsf{Ksk}}^2$ because there are exactly N pairs $i, j \in [N]$ such that $i+j-2 \bmod N = d$. Then from correctness of the NTRU decryption procedure we have $\mathsf{Var}(e_1 \cdot g + f \cdot e_2) \le \sigma_{\mathsf{Dec}}^2$ and the expectation magnitude is bounded by $2/3 \cdot N \cdot \mathsf{U}(c)$. Therefore σ_{out}^2 is as in the lemma statement.

Gadget Encryption and Multiplication.

Below we give the NTRU gadget encryption, and multiplication algorithm, which is analogous to the GSW encryption scheme [38].

Definition 5 (NTRU Gadget Encryption and Multiplication) *We define a gadget NTRU sample of a message $m_G \in R_Q$ as*

$$\mathbf{c}_G = \mathsf{G\text{-}NTRU}_{\sigma_{\mathsf{Bk}}}(f, m_G) = \left[\mathsf{NTRU}_{\sigma_{\mathsf{Bk}}}(f, m_G \cdot \mathsf{L}_{\mathsf{Bk}}^{i-1})\right]_{i=1}^{\ell_{\mathsf{Bk}}}, \tag{3}$$

where $\ell_{\mathsf{Bk}} = \lceil \log_{\mathsf{L}_{\mathsf{Bk}}}(Q) \rceil$. We define the gadget multiplication procedure GMul as

$$\mathsf{GMul}(\mathbf{c}_G, c) = \langle \mathbf{c}_G, \mathsf{G}^{-1}(c, \mathsf{L}_{\mathsf{Bk}}) \rangle, \tag{4}$$

where $c \in \mathcal{R}_Q$ and in particular $c = \mathsf{NTRU}_\sigma(f, m)$.

Note that the gadget ciphertext at Eq. 3 is nothing more than a vector of NTRU ciphertexts of $m_G \cdot \mathsf{L}_{\mathsf{Bk}}^{i-1}$. Addition and scalar multiplication for individual gadget ciphertexts are as for NTRU. In this paper, we will never directly decrypt gadget ciphertexts; hence we omit to describe the decryption algorithm.

Below we analyze the correctness of the gadget multiplication algorithm. We use only the case where the message m_G is a monomial with its coefficient in $\mathbb{Z}_2$. Hence we will focus the analysis only on this special case for simplicity. We give the generalized analysis in Appendix 8 for completeness.

Lemma 5 (Correctness of NTRU Gadget Multiplication) *If $c_{\mathsf{out}} = \mathsf{GMul}(\mathbf{c}_G, c)$ and $m_G \in \mathbb{Z}_2$, then $c_{\mathsf{out}} = \mathsf{NTRU}_{\sigma_{\mathsf{out}}}(f, c \cdot m_G)$, where $\sigma_{\mathsf{out}}^2 \le N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}}^2/4 \cdot \sigma_{\mathsf{Bk}}^2$.*

If additionally $c = \mathsf{NTRU}_\sigma(f, m)$, then the output ciphertexts is $c_G = \mathsf{NTRU}_{\sigma_{\mathsf{out}}}(f, m \cdot m_G)$, where $\sigma_{\mathsf{out}}^2 \le \sigma^2 + N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}}^2/4 \cdot \sigma_{\mathsf{Bk}}^2$, and $\mathsf{U}(c_{\mathsf{out}}) \le \mathsf{U}(c) + N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}} \cdot \mathsf{U}(c_G)$.

Proof 5 *Let us denote the base-L_{Bk} decomposition of c as $[c_k]_{k=1}^{\ell_{\mathsf{Bk}}} = \mathsf{G}^{-1}(c, \mathsf{L}_{\mathsf{Bk}})$ which is such that $\sum_{k=1}^{\ell_{\mathsf{Bk}}} c_k \cdot \mathsf{L}_{\mathsf{Bk}}^{k-1} = c \mod Q$. From definition we have*

$$\begin{aligned} c_{\mathsf{out}} &= \left\langle \mathbf{c}_G, \mathsf{G}^{-1}(c, \mathsf{L}_{\mathsf{Bk}}) \right\rangle \\ &= \sum_{k=1}^{\ell_{\mathsf{Bk}}} \mathsf{NTRU}_{\sigma_{\mathsf{Bk}}}(f, m_G \cdot \mathsf{L}_{\mathsf{Bk}}^{k-1}) \cdot c_k \\ &= \mathsf{NTRU}_{\tilde{\sigma}}\Big(f, m_G \cdot \sum_{k=1}^{\ell_{\mathsf{Bk}}} c_k \cdot \mathsf{L}_{\mathsf{Bk}}^{k-1}\Big) \\ &= \mathsf{NTRU}_{\tilde{\sigma}}(f, m_G \cdot c). \end{aligned}$$

From linear homomorphism (see Lemma 3) of NTRU we have $\tilde{\sigma}^2 \le N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}}^2/4 \cdot \sigma_{\mathsf{Bk}}^2$. Note that in the above c_k are ring elements with coefficients in $\big[-\mathsf{L}_{\mathsf{Bk}}/2, \mathsf{L}_{\mathsf{Bk}}/2\big)$. Denote $\mathsf{NTRU}_{\tilde{\sigma}}\big(f, m_G \cdot c\big) =$

$e_1'/f + e_2' + m_G \cdot c$. Since the coefficient of m_G is smaller than or equal 1 we have

$$\begin{aligned} c_{\mathsf{out}} &= e_1'/f + e_2' + r' + m_G \cdot c \\ &= e_1'/f + e_2' + r' + m_G \cdot \big(e_1/f + e_2 + r + m\big) \\ &\leq \big(e_1' + e_1\big)/f + e_2' + e_2 + r' + r + m_G \cdot m. \end{aligned}$$

The same reasoning is applied to the analysis of the random variable r'. To summarize we have $\sigma^2_{\mathsf{out}} \leq \sigma^2 + N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}^2_{\mathsf{Bk}}/4 \cdot \sigma^2_{\mathsf{Bk}}$, and $\mathsf{U}(c_{\mathsf{out}}) \leq \mathsf{U}(c) + N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}} \cdot \mathsf{U}(c_G)$.

Modulus Switching.

Here we analyze the modulus switching procedure for NTRU and LWE ciphertexts.

Lemma 6 (Modulus Switching for NTRU) *Let us denote $c = \mathsf{NTRU}_\sigma(f, \frac{Q}{t} \cdot m)$. We define the NTRU modulus switching procedure as*

$$\mathsf{ModSwitch}(c, q) = \left\lfloor c \cdot \frac{q}{Q} \right\rceil,$$

where $q \leq Q$.

If $c_{\mathsf{out}} = \mathsf{ModSwitch}(c, q)$, then $c_{\mathsf{out}} = \mathsf{NTRU}_{\sigma_{\mathsf{out}}}(f, \frac{q}{t} \cdot m)$, where $\sigma^2_{\mathsf{out}} = \left(\frac{q}{Q} \cdot \sigma\right)^2$ and $\mathsf{U}(c_{\mathsf{out}}) \leq \frac{q}{Q} \cdot \mathsf{U}(c) + 1$.

Proof 6 *Denote $c = e_1/f + e_2 + \frac{Q}{t} \cdot m$. From definition we have:*

$$\begin{aligned} c_{\mathsf{out}} = \left\lfloor \frac{q}{Q} \cdot c \right\rceil &= \frac{q}{Q} \cdot c + r' \\ &= \frac{q}{Q} \cdot e_1/f + \frac{q}{Q} \cdot (e_2 + r) + r' + \frac{q}{Q} \cdot \frac{Q}{t} \cdot m \\ &\leq \frac{q}{Q} \cdot e_1/f + \frac{q}{Q} \cdot e_2 + \frac{q}{Q} \cdot r + r' + \frac{q}{t} \cdot m, \end{aligned}$$

where r has coefficients in $[-1/2, 1/2]$. Note however, that the signs of the coefficients are the same as the signs of e_2. Therefore, the rounding error shifts the expectation of the $e_2 + r$ from 0 at most 1. Hence $\sigma^2_{\mathsf{out}} \leq \left(\frac{q}{Q} \cdot \sigma\right)^2$, and the infinity norm on the expectation of the noise terms is bounded by 1 due to the rounding error.

Below we remind the modulus switching algorithm LWE. Since this is a standard algorithm, we give its correctness proof in Appendix 8 for completeness.

Lemma 7 (Modulus Switching for LWE) *Let us define the following ciphertext $\mathbf{c} = \mathsf{LWE}_\sigma\big(\mathbf{s}, \frac{Q}{t} \cdot m\big)$, where $\mathbf{s} \in \mathbb{Z}_Q^n$. Let us define the LWE modulus switching procedure as*

$$\mathsf{ModSwitch}(\mathbf{c}, q) = \left\lfloor \left[\mathbf{c} \cdot \frac{q}{Q}\right]_{i=1}^{n+1} \right\rceil$$

for $q \leq Q$.

If $\mathbf{c}_{\mathsf{out}} \leftarrow \mathsf{ModSwitch}(\mathbf{c}, q)$, then $\mathbf{c}_{\mathsf{out}} = \mathsf{LWE}_{\sigma_{\mathsf{out}}}\big(\mathbf{s}, \frac{q}{t} \cdot m\big)$, where

$$\sigma^2_{\mathsf{out}} \leq \left(\frac{q}{Q} \cdot \sigma\right)^2 + \mathsf{Ha}(\mathbf{s}) \cdot \mathsf{Var}(s)$$

the bound on the expectation of the noise term is at most 1 due to the rounding error.

```
$\mathsf{BootKeyGen}(\mathbf{s}, f, P)$:
Input:
- LWE secret key $\mathbf{s} \in \{0,1\}$.
- NTRU secret key $f \in \mathcal{R}_Q$.
- Integer $P$ s.t. $Q < P$.

1 : For $i \in [n]$:
2 :     Set $\mathsf{Bk}[i] \leftarrow \mathsf{G\text{-}NTRU}_{\sigma_{\mathsf{Bk}}}\big(f, \mathbf{s}[i]\big) \in \mathcal{R}_P$.
3 : $\mathsf{Ksk} \leftarrow \mathsf{KSSetup}\big(f, \mathbf{s}\big)$.
4 : Return $\mathsf{Ksk}$ and $\mathsf{Bk}$.
```

```
$\mathsf{BlindRotate}(\mathbf{c}, \mathsf{Bk}, c_{\mathsf{acc}})$:
Input:
- Ciphertext $\mathbf{c} = \mathsf{LWE}_\sigma\Big(\mathbf{s}, \frac{2N}{t_1} \cdot m\Big)$.
- Blind rotation key $\mathsf{Bk}$.
- Accumulator $c_{\mathsf{acc}} = \mathsf{NTRU}_{\sigma_{\mathsf{acc}}}\big(f, \frac{P}{t_2} \cdot a_{\mathsf{rot}} \cdot f^{-1}\big)$.

1 : $c_{\mathsf{acc},1} \leftarrow c_{\mathsf{acc}} \cdot X^{\mathbf{c}[1]}$.
2 : For $i \in [n]$:
3 :     $c_{\mathsf{acc},i+1} \leftarrow \mathsf{GMul}\big(\mathsf{Bk}[i],$
            $c_{\mathsf{acc},i} \cdot X^{\mathbf{c}[i+1]} - c_{\mathsf{acc},i}\big) + c_{\mathsf{acc},i}$.
4 : Return $c_{\mathsf{acc},n+1}$.
```

Figure 1: Bootstrapping Key Generation and Blind Rotation

4 Computing on Ciphertexts and Bootstrapping

In this section, we give our bootstrapping algorithm. At Figure 1 we give the adaptation of TFHE-style blind rotation [26] and the plug it into the FHEW-style bootstrapping [32, 56] and functional bootstrapping [61, 53] algorithm. Figures 1 and 2 give the basic schemes. In particular, we assume that the bootstrapping algorithm gets as input an accumulator holding a rotation polynomial. At the end of this section, we discuss how to build such an accumulator for a chosen rotation polynomial. For simplicity, we describe only the version that uses blind rotation (see Figure 1) LWE ciphertexts with binary secret keys. We note that we may extend the bootstrapping algorithm to bootstrap LWE ciphertexts with ternary or Gaussian distributed secret keys via standard techniques [32, 56]. Finally, at Table 2 we summarize all noise of ciphertexts output by the procedures from this section and Section 3.

Setting up the Rotation Polynomial and the Accumulator. Before giving the formal analysis of the bootstrapping algorithm, let us briefly explain how to choose the rotation polynomial a_{rot}. Suppose we want to bootstrap a ciphertext that holds the message $m \in \mathbb{Z}_{t_1}$, and along the way, we want to compute the function $F : \mathbb{Z}_{t_1} \mapsto \mathbb{Z}_{t_2}$. To do so, we need to construct a rotation polynomial $a_{\mathsf{rot}} \in \mathcal{R}_Q$. We set $a_{\mathsf{rot}}[N-y] = -F(\lfloor \frac{t_1 \cdot y}{2 \cdot N} \rceil) \mod P$ for all $y \in [1, N]$ and $a_{\mathsf{rot}}[1] = F(0)$.

When using the $\mathsf{Bootstrap}$ algorithm, we can compute functions F such that $F(x + t_1/2 \mod t_1) = -F(x \mod t_1) \mod t_1$ for $x \in \mathbb{Z}_{t_1}$ and even t_1. When running $\mathsf{F\text{-}Bootstrap}$ we do not have the above restriction. In particular, we can compute any $F : \mathbb{Z}_{t_1} \mapsto \mathbb{Z}_{t_2}$ and any $t_1 < N$. For the $\mathsf{F\text{-}Bootstrap}$ algorithm we have another rotation polynomial a_{sgn}, whose coefficients are all set to $6N/4$. We compute $a_{\mathsf{sgn}} \cdot X^y$, which constant coefficient is $2N/4$ for $y \in [1, N]$ and $6N/4$ for $y \in [N+1, 2N]$. When we modulus switch the ciphertext $\mathbf{c}_{\mathsf{pre}}$ to $\mathbb{Z}_N$ instead of $\mathbb{Z}_{2N}$, then we have $y = b_{\mathsf{pre}} - \mathbf{a}_{\mathsf{pre}}^\top \mathbf{s} = m_{\mathsf{pre}} + e_{\mathsf{pre}} + kN \mod 2N$. Now when $k = 0$, then a_{sgn} returns $2N/4$, hence we do not add anything in step 6. For $k = 1$, we add $6N/4 - 2N/4 = N$ and we get rid of the $k \cdot N$ term in y. Thereby, we ensure that the second blind rotation rotates a_{rot} by numbers from $\mathbb{Z}_N$ instead of $\mathbb{Z}_{2N}$. See the proof of Theorem 1 for more details.

```
$\mathsf{Bootstrap}(c, \mathsf{Ksk}, \mathsf{Bk}, c_{\mathsf{acc}}, d)$:
Input:
 - Ciphertext $c = \mathsf{NTRU}_\sigma\Big(f, \frac{Q}{t_1} \cdot m \cdot f^{-1}\Big)$.
 - NTRU to LWE key-switching key $\mathsf{Ksk}$.
 - Blind rotation key $\mathsf{Bk}$.
 - Accumulator $c_{\mathsf{acc}} = \mathsf{NTRU}_{\sigma_{\mathsf{acc}}}\Big(f, \frac{P}{t_2} \cdot a_{\mathsf{rot}} \cdot f^{-1}\Big)$.
 - Index $d \in [N]$.

1 :  $\mathbf{c}_{\mathsf{LWE}} \leftarrow \mathsf{KSwitch}\big(\mathsf{Ksk}, c, d\big)$.
2 :  $\mathbf{c}_{\mathsf{in}} \leftarrow \mathsf{ModSwitch}\big(\mathbf{c}_{\mathsf{LWE}}, N\big)$.
3 :  $c_{\mathsf{acc,out}} \leftarrow \mathsf{BlindRotate}\big(\mathbf{c}_{\mathsf{in}}, \mathsf{Bk}, c_{\mathsf{acc}}\big)$.
4 :  Return $c_{\mathsf{out}} \leftarrow \mathsf{ModSwitch}\big(c_{\mathsf{acc}}, Q\big)$.
```

```
$\mathsf{F\text{-}Bootstrap}(c, \mathsf{Ksk}, \mathsf{Bk}, c_{\mathsf{acc}}, c_{\mathsf{sgn}}, d, t)$:
Input:
 - Ciphertext $c = \mathsf{NTRU}_\sigma\Big(f, \frac{Q}{t_1} \cdot m \cdot f^{-1}\Big)$.
 - NTRU to LWE key-switching key $\mathsf{Ksk}$.
 - Blind rotation key $\mathsf{Bk}$.
 - Accumulator $c_{\mathsf{acc}} = \mathsf{NTRU}_{\sigma_{\mathsf{acc}}}\Big(f, \frac{P}{t_2} \cdot a_{\mathsf{rot}} \cdot f^{-1}\Big)$.
 - Accumulator $c_{\mathsf{sgn}} = \mathsf{NTRU}_{\sigma_{\mathsf{acc}}}\Big(f, \frac{P}{t_2} \cdot a_{\mathsf{sgn}} \cdot f^{-1}\Big)$.
 - Index $d \in [N]$ and integer $t \in \mathbb{N}$.

1 :  $\mathbf{c}_{\mathsf{Ksk}} \leftarrow \mathsf{KSwitch}\big(\mathsf{Ksk}, c, d\big) \in \mathbb{Z}_Q^{n+1}$.
2 :  $\mathbf{c}_{\mathsf{pre}} \leftarrow \mathsf{ModSwitch}\big(\mathbf{c}_{\mathsf{Ksk}}, N\big) + \Big[\lfloor \frac{N}{2t} \rceil, \mathbf{0}\Big]$.
3 :  $c_{\mathsf{acc,msg}} \leftarrow \mathsf{BlindRotate}\big(\mathsf{Bk}, c_{\mathsf{sgn}}, \mathbf{c}_{\mathsf{pre}}\big)$.
4 :  $\mathbf{c}_{\mathsf{msg}} \leftarrow \mathsf{KSwitch}\big(\mathsf{Bk}, \mathbf{c}_{\mathsf{msg}}, 1\big) \in \mathbb{Z}_Q^{n+1}$.
5 :  $\mathbf{c}_{\widehat{\mathsf{msg}}} \leftarrow \mathsf{ModSwitch}\big(\mathbf{c}_{\mathsf{msg}}, 2N\big) \in \mathbb{Z}_{2N}^{n+1}$.
6 :  $\mathbf{c}_{\mathsf{in}} \leftarrow \mathbf{c}_{\mathsf{pre}} + \mathbf{c}_{\widehat{\mathsf{msg}}} - \frac{2N}{4} \in \mathbb{Z}_{2N}^{n+1}$.
7 :  $c_{\mathsf{acc,out}} \leftarrow \mathsf{BlindRotate}\big(\mathsf{Bk}, a_{\mathsf{rot}}, \mathbf{c}_{\mathsf{in}}\big)$.
8 :  Return $c_{\mathsf{out}} \leftarrow \mathsf{ModSwitch}\big(c_{\mathsf{acc}}, Q\big)$.
```

Figure 2: Bootstrap and Full Domain Functional Bootstrapping.

Algorithm	Noise variance of the output ciphertext
$\mathsf{Dec}(c)$	$\sigma^2_{\mathsf{Dec}} \le 4/3 \cdot N \cdot \sigma^2$
$\mathsf{KSwitch}(\mathsf{Ksk}, c, d)$	$\sigma^2_{\mathsf{Dec}} + N \cdot \ell_{\mathsf{Ksk}} \cdot \mathsf{L}^2_{\mathsf{Ksk}}/4 \cdot \sigma^2_{\mathsf{Ksk}}$
$\mathsf{GMul}(\mathbf{c}_G, c)$	$\sigma^2 + N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}^2_{\mathsf{Bk}}/4 \cdot \sigma^2_{\mathsf{Bk}}$
$\mathsf{ModSwitch}(c, q)$	$\left(\frac{q}{Q} \cdot \sigma\right)^2$
$\mathsf{ModSwitch}(\mathbf{c}, q)$	$\left(\frac{q}{Q} \cdot \sigma\right)^2 + \mathsf{Ha}(\mathbf{s}) \cdot \mathsf{Var}(s)$
$\mathsf{Bootstrap}(c, \mathsf{Ksk}, \mathsf{Bk}, c_{\mathsf{acc}}, d)$	$\sigma^2_{\mathsf{out}} \le \left(\frac{q}{Q}\right)^2 \cdot \left(\sigma^2_{\mathsf{in}} \le \sigma^2_{\mathsf{acc}} + n \cdot N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}^2_{\mathsf{Bk}}/4 \cdot \sigma^2_{\mathsf{Bk}}\right)$ $\sigma^2_{\mathsf{in}} \le \left(\frac{q}{Q}\right)^2 \cdot \left(\sigma^2_{\mathsf{Dec}} + N \cdot \ell_{\mathsf{Ksk}} \cdot \mathsf{L}^2_{\mathsf{Ksk}}/4 \cdot \sigma^2_{\mathsf{Ksk}}\right) + \mathsf{Ha}(\mathbf{s}) \cdot \mathsf{Var}(s)$
$\mathsf{F\text{-}Bootstrap}(c, \mathsf{Ksk}, \mathsf{Bk}, c_{\mathsf{acc}}, c_{\mathsf{sgn}}, d, t)$	$\sigma^2_{\mathsf{out}} \le \left(\frac{q}{Q}\right)^2 \cdot \left(\sigma^2_{\mathsf{in}} \le \sigma^2_{\mathsf{acc}} + n \cdot N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}^2_{\mathsf{Bk}}/4 \cdot \sigma^2_{\mathsf{Bk}}\right)$ $\sigma^2_{\mathsf{in}} \le 2 \cdot \left(\frac{q}{Q}\right)^2 \cdot \left(\sigma^2_{\mathsf{Dec}} + N \cdot \ell_{\mathsf{Ksk}} \cdot \mathsf{L}^2_{\mathsf{Ksk}}/4 \cdot \sigma^2_{\mathsf{Ksk}}\right) + 2 \cdot \mathsf{Ha}(\mathbf{s}) \cdot \mathsf{Var}(s)$

Table 2: Summary of the Noise Variances.We assume all algorithms and their inputs are generated as the respective definition. For the inout ciphertexts we assume c and $\mathbf{c}$ to be an NTRU and LWE a ciphertext both with Gaussian parameter σ.

4.1 The Bootstrapping Algorithms

Below we give the correctness and noise analysis of our bootstrapping algorithm.

Theorem 1 (Correctness of the Bootstrapping Algorithm) *Let $a_{\mathsf{rot}} \in \mathcal{R}_P$ be such that $(a_{\mathsf{rot}} \cdot X^y)[1] = F(\lfloor \frac{t_1}{N} \cdot y \rceil)$, where $y \in \mathbb{Z}_N$ and $F : \mathbb{Z}_{t_1} \mapsto \mathbb{Z}_{t_2}$.*

Let $\mathbf{c}_{\mathsf{in}}$ and c_{out} be as in step 2 and 4 of the Bootstrap *(resp. step 6 and 8 of the* F-*Bootstrap) algorithm at figure 2. Then*

$$\mathbf{c}_{\mathsf{in}} = \mathsf{LWE}_{\sigma_{\mathsf{in}}}\big(\mathbf{s}, \frac{N}{t_1} \cdot m \cdot f^{-1}\big)$$

and

$$c_{\mathsf{out}} = \mathsf{NTRU}_{\sigma_{\mathsf{out}}}\big(f, \frac{Q}{t_2} \cdot F(m) \cdot f^{-1}\big)$$

where σ_{in} and σ_{out} are given by Table 2, and $q = 2N$ for the Bootstrap *(resp. $q = N$ for the* F-*Bootstrap) algorithm.*

Proof 7 *We will divide the proof into four parts. First we give the analysis of the* BlindRotate *algorithm from Figure 1. Then we give the analysis of c_{out} that is returned by* Bootstrap *and* F-*Bootstrap. Finally, we give the correctness and the noise analysis of $\mathbf{c}_{\mathsf{in}}$. Note that correctness and the noise analysis of the three last parts mostly follows from the correctness of the underlying algorithms. The most involved part of the proof is blind rotation given below.*
Blind Rotation. *Let us first inspect the ith iteration for the blind rotation's* ***For*** *loop. Denote $c_{\mathsf{acc},i} = \mathsf{NTRU}_{\sigma_{\mathsf{acc},i}}\big(f, M_i\big)$, where $M_i \in \mathcal{R}_P$ is the message in the current iteration. Remind that $\mathsf{Bk}[i]$ encrypts $\mathbf{s}[i] \in \{0,1\}$. Hence there are two cases we must consider.*

1. *The case for $\mathbf{s}[i] = 0$. In this case, the* GMul *algorithm outputs an NTRU encryption of 0 with Gaussian parameter σ_G to which we add $c_{\mathsf{acc},i}$. Hence we have that*

$$\begin{aligned} c_{\mathsf{acc},i+1} &= \mathsf{NTRU}_{\sigma_G}\big(f, 0\big) + c_{\mathsf{acc},i} \\ &= \mathsf{NTRU}_{\sigma_G}\big(f, c_{\mathsf{acc},i}\big). \end{aligned}$$

   Remind that from Lemma 5, we have $\sigma_G^2 \leq N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}}^2/4 \cdot \sigma_{\mathsf{Bk}}^2$. Therefore, this step adds σ_G^2 to the error variance $\sigma_{\mathsf{acc},i}^2$ of $c_{\mathsf{acc},i}$.

2. *The case for $\mathbf{s}[i] = 1$. In this case* GMul *output an NTRU ciphertext $c_{G,i} = \mathsf{NTRU}_{\sigma_G}\big(f, c_{\mathsf{acc},i} \cdot X^{\mathbf{c}_{\mathsf{in}}[i]} - c_{\mathsf{acc},i}\big)$ to which we add $c_{\mathsf{acc},i}$. Note that from additive homomorphism the $c_{\mathsf{acc},i}$ cancel out and we have*

$$c_{\mathsf{acc},i+1} = \mathsf{NTRU}_{\sigma_G}\big(f, c_{\mathsf{acc},i} \cdot X^{\mathbf{c}_{\mathsf{in}}[i]}\big).$$

   Furthermore, since the scalar $X^{\mathbf{c}_{\mathsf{in}}[i]}$ has infinity norm equal one, we have that the ciphertext $c_{\mathsf{acc},i} \cdot X^{\mathbf{c}_{\mathsf{in}}[i]}$ has the same noise variance as the ciphertext $c_{\mathsf{acc},i}$. As in the previous case, the iteration add σ_G^2 to the error variance $\sigma_{\mathsf{acc},i}^2$ of $c_{\mathsf{acc},i}$.

Remind that we initialize the ciphertext $c_{\mathsf{acc},1}$ with the message $M \cdot X^{\mathbf{c}_{\mathsf{in}}[1]}$, where $M = \frac{P}{t_2} \cdot a_{\mathsf{rot}} \cdot f^{-1}$. Furthermore, the $c_{\mathsf{acc},1}$ has noise parameter σ_{acc}, which is the same noise parameter as c_{acc}. After the nth iteration the ciphertext $c_{\mathsf{acc},n+1}$ encrypts $M \cdot \prod_{i=1}^{n} X^{\mathbf{c}[i]} = M \cdot X^{\mathsf{Phase}(\mathbf{c}_{\mathsf{in}})}$. The bound on the *final noise term follows from the noise analysis of* GMul *in Lemma 5. Namely, each iteration of the loop adds the additive term* $\ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}}^2/4 \cdot \sigma_{\mathsf{Bk}}^2$ *to the total noise. To summarize we have*

$$\sigma_{\mathsf{acc},n+1}^2 \leq \sigma_{\mathsf{acc}}^2 + n \cdot N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}}^2/4 \cdot \sigma_{\mathsf{Bk}}^2.$$

Bootstrap. *The correctness of this part immediately follows from the correctness of* BlindRotate *and* ModSwitch*. However to summarize, we have that* BlindRotate *outputs a NTRU ciphertext of* $\frac{P}{t_2} \cdot a_{\mathsf{rot}} \cdot f^{-1} \cdot X^{\mathsf{Phase}(\mathbf{c}_{\mathsf{in}})}$*, which after modulus switching becomes a ciphertext of* $\frac{Q}{t_2} \cdot a_{\mathsf{rot}} \cdot f^{-1} \cdot X^{\mathsf{Phase}(\mathbf{c}_{\mathsf{in}})}$*. From the assumption on* a_{rot} *we have that the constant coefficient of* c_{out}*'s plaintext is* $\frac{Q}{t_2} \cdot F(\lfloor \frac{t_1}{N} \cdot \mathsf{Phase}(\mathbf{c}_{\mathsf{in}}) \rceil)$*.*
Functional Bootstrap. *Correctness of the full domain functional bootstrapping is as follows. Denote* $\mathbf{c}_{\mathsf{pre}} = [b_{\mathsf{pre}}, \mathbf{a}_{\mathsf{pre}}]$ *such that* $b_{\mathsf{pre}} = \mathbf{a}_{\mathsf{pre}}^\top \mathbf{s} + m_{\mathsf{pre}} + e_{\mathsf{pre}} + \in \mathbb{Z}_N$*. Note that since we add* $\left[\lfloor \frac{N}{2t} \rceil, \mathbf{0}\right]$ *we ensure that* $0 \leq m_{\mathsf{pre}} + e_{\mathsf{pre}} < N$*. This shifting operation is important as otherwise we would not be able to choose an appropriate rotation polynomial. We set all coefficients of the rotation polynomial* a_{sgn} *to* $6N/4$ *except for the constant coefficient that is set to* $2N/4$*. We blind rotate* $\mathbf{c}_{\mathsf{pre}}$ *modulo* $2N$ *with* a_{sgn}*, so* $b_{\mathsf{pre}} - \mathbf{a}_{\mathsf{pre}}^\top \mathbf{s} = m_{\mathsf{pre}} + e_{\mathsf{pre}} + kN \mod 2N$ *for some* $k \in \{0,1\}$*, where* m_{pre} *is the modulus switching of the message* m *that is encoded in* $\mathbf{c}$*. From correctness of blind rotation, and the key and modulus switching we have that* $\mathbf{c}_{\widehat{\mathsf{msg}}}$ *decrypts to* $\frac{2N}{4}$ *if* $k=0$ *and* $\frac{3 \cdot 2N}{4}$ *if* $k=1$*. We can write the decryption of* $\mathbf{c}_{\widehat{\mathsf{msg}}}$ *as* $k \cdot \frac{6N}{4} + (1-k) \cdot \frac{2N}{4}$*. So when we add* $\mathbf{c}_{\mathsf{pre}} + \mathbf{c}_{\widehat{\mathsf{msg}}} - \frac{2N}{4}$*, the term* $kN + k \cdot \frac{6N}{4} + (1-k) \cdot \frac{2N}{4} - \frac{2N}{4}$ *is zero for both* $k \in \{0,1\}$*. Hence we have* $b_{\mathsf{in}} = \mathbf{a}^\top \mathbf{s} + m_{\mathsf{pre}} + e_{\mathsf{in}} \mod 2N$*, where* $m_{\mathsf{pre}} + e < N$*. Therefore, we can choose the coefficients of the rotation polynomial such that* $(a_{\mathsf{rot}} \cdot X^{\frac{N}{t} m_{\mathsf{pre}} + e})[1] = F(\frac{N}{t} \cdot m_{\mathsf{pre}} + e)$*. Note that we will only multiply the rotation polynomials by* $X^{m_{\mathsf{pre}}+e}$*, where* $0 \leq m_{\mathsf{pre}} + e < N$*. In particular, the negacyclicity problem never occurs. In other words, we directly set the coefficient to encode the lookup table, and we do not worry that the rotation exceeds the number of coefficients and changes the sign of the output. Finally, the variance* σ_{out}^2 *follows from the analysis of blind rotation and modulus switching as for the* Bootstrap *algorithm.*
The "Small" Modulus LWE Ciphertext. *The correctnes and the noise variance for* $\mathbf{c}_{\mathsf{in}}$ *follows from correctness of* KSwitch *(Lemma 4) and the LWE* ModSwitch *(Lemma 7). In the case of* F-*Bootstrap we actually add* $\mathbf{c}_{\mathsf{pre}}$ *and* $\mathbf{c}_{\widehat{\mathsf{msg}}}$ *to obtain* $\mathbf{c}_{\mathsf{in}}$*. Note that the noise terms of both* $\mathbf{c}_{\mathsf{pre}}$ *and* $\mathbf{c}_{\widehat{\mathsf{msg}}}$ *are the same noise variance as* $\mathbf{c}_{\mathsf{in}}$ *in the* Bootstrap *algorithm, because both are products of key switching and LWE modulus reduction on NTRU ciphertexts returned blind rotation and key switched from* P *to the* Q *modulus. To summarize, for the full domain functional bootstrapping the variance of* $\mathbf{c}_{\mathsf{in}}$*'s noise is twice as high as for* Bootstrap*.*

4.2 Computing on Encrypted Data and Packing

To compute on encrypted data, we can use the homomorphism of NTRU ciphertexts to compute affine functions over $\mathbb{Z}_{t_1}$. After bootstrapping a ciphertext holding a message m at position d we obtain an NTRU ciphertext encrypting $g = \frac{Q}{t_2} a_{\mathsf{rot}} \cdot f^{-1} \cdot X^{\frac{N}{t_2} m + e}$. The extraction and key switching steps return a LWE ciphertext of $(f \cdot g)[1] = \frac{Q}{t_2} F(m)$. Note that we can still compute affine functions on these ciphertexts with monomials of degree zero. But any further bootstrapping must extract the LWE from position $d = 1$. Note that it may be tempting to key switch to LWE right after the blind rotation step. Unfortunately, this requires us to take the LWE with the larger modulus Q or pay a high price in terms of lower correctness when trying to bootstrap such ciphertext again.

When working over $\mathbb{Z}_Q[X]/(X^N + 1)$, we can compute any negacyclic function F on $\mathbb{Z}_{t_1}$. Furthermore, when applying the F-Bootstrap technique from [61, 53] we can generalize the method to compute any function on $\mathbb{Z}_{t_1}$. In this case we can for example correctly compute $x^2 \mod t_1$

or $x^{-1} \mod t_2$ for $x \in \mathbb{Z}_{t_1}$. If furthermore $\mathbb{Z}_{t_1}$ contains inverses of 4 then we can compute $x \cdot y = (\frac{x+y}{2})^2 - (\frac{x-y}{2})^2 \mod t_1$ with only two invocations of the bootstrapping algorithm. This way we can efficiently compute arithmetic circuits. Furthermore, the arithmetic can be easily extended to composite $\mathbb{Z}_p$ with $p = \prod_{i=1}^{m} t_{1,i}$ where all $t_{1,i}$ are pairwise co-prime from the Chinese remainder theorem.

Building Accumulators. Note that the accumulator we give as input to the bootstrapping procedure is key-dependent. We do the following to allow the evaluator to choose its rotation polynomials, thus selecting the circuit to be computed. We publish additionally a gadget encryption of f^{-1}. The evaluator obtains the accumulator by gadget multiplying the encryption with the rotation polynomial of its choice. We note that the error due to gadget multiplication is a tiny fraction of the error due to blind rotation. As we will show in Section 5, in practice, we have over 2^9 gadget multiplications in the blind rotation step. Furthermore, this multiplication has to be executed only once per rotation polynomial because the evaluator can store and reuse his accumulators.

Encrypting Data. To send encrypted data, we have a few options. We can either send LWE ciphertexts, with modulus N and error distribution being a Gaussian with standard deviation matching the standard deviation of the error of the LWE ciphertext after switching the modulus. Then, instead of key switching the input NTRU ciphertexts, we can immediately start to compute step four on the LWE ciphertexts. The downside of this method is that we require $n+1$ elements in $\mathbb{Z}_N$ to transmit a single message. Another method is to set a message into a coefficient of the NTRU ciphertext. This way, we may transmit N messages at the cost of only N elements in $\mathbb{Z}_Q$. We note that for the initial NTRU ciphertext, we may actually take a smaller modulus and obtain an even better ciphertext rate. The moduli Q and P are chosen to support the error induced by the blind rotation as well as the NTRU to LWE key switching part.

Finally, the naive way to return the outcome of the computation is to return the NTRU ciphertext from the last invocation of bootstrapping (or after additionally computing some affine functions on a vector of NTRU ciphertexts). In this case, the ciphertext rate for the result is rather weak since we transmit N elements in $\mathbb{Z}_Q$ per message. What we can do, is either run the NTRU to LWE switching procedure to reduce the rate to $n+1$ elements in $\mathbb{Z}_q$, or we can try to pack the outcome of multiple NTRU ciphertexts into a single NTRU ciphertext. For this purpose, we need an additional packing key that works as the NTRU to LWE key switching procedure but has NTRU ciphertexts instead of LWE ciphertexts.

Amortization. We can use similar techniques as showed by Carpov et al. [20] to compute multiple functions on the same input with only a single invocation of the bootstrapping algorithm. Namely instead of setting the accumulator to be a NTRU ciphertext of $\frac{P}{t_2} \cdot a_{\mathsf{rot}} \cdot f^{-1}$, we set it to $\frac{P}{t_2} \cdot f^{-1}$. Then before step 7 of the bootstrapping procedure, we multiply the blind rotated accumulator with a_{rot}. It is easy to see that the resulting message is the same, but we note that the noise rate will be greater and dependent on the norm of a_{rot}. Nevertheless, for applications like binary decomposition of field elements, we set a_{rot} to have only binary coefficients.

5 Security, Parameters and Correctness

We present all parameter sets on Table 3. All our parameters are targeted to achieve at least 128-bits of security. The NTRU secret key f is always assumed to have coefficients from $\{-1, 0, 1\}$. We chose e_2 such that $\mathsf{Var}(e_2) = \frac{1}{16}$. For every NTRU ciphertext we sample a fresh g with coefficients in $\{-1, 0, 1\}$. We set e_1 such that $\mathsf{Var}(e_1) = 2/3$. We take Q as close to N as possible, but we are limited by the error of the key switching procedure. The smaller Q, the smaller the key switching key, and the bigger the error in the resulting LWE ciphertext that stems from the key switching

	n	$\log(P)$	$\log(Q)$	N	L_{Bk}	ℓ_{Bk}	ℓ_{Ksk}	$\mathsf{stddev}_{\mathsf{Ksk}}$
Binary LWE Secret Key								
NTRU-ν-um-11-B	$2^9 + 125$	2^{30}	2^{25}	2^{11}	2^6	5	25	2^{10}
NTRU-ν-um-12-B	$2^9 + 238$	2^{45}	2^{33}	2^{12}	2^{15}	3	33	2^{14}
NTRU-ν-um-13-B	$2^9 + 315$	2^{45}	2^{34}	2^{13}	2^{15}	3	34	2^{14}
NTRU-ν-um-14-B	$2^9 + 390$	2^{45}	2^{36}	2^{14}	2^{15}	3	36	2^{14}

Table 3: Parameter Sets. The hamming weight of these parameters is not enforced, and all coefficients are from the same distribution. In other words we have $\mathsf{Ha}(f) = N$ and $\mathsf{Ha}(\mathbf{s}) = n$.

Binary LWE Secret Key				
Set:	11-B	12-B	13-B	14-B
$\log(t) \setminus$ sk	10.45%	64.43%	93.36%	82.96%
4	$(0,\ 2^{-13})$	$(0,\ 0)$	$(0,\ 0)$	$(0,\ 0)$
5	$(2^{-33},\ 2^{-4})$	$(0,\ 0)$	$(0,\ 0)$	$(0,\ 0)$
6	$(2^{-9},\ 0.32)$	$(0,\ 2^{-15})$	$(0,\ 0)$	$(0,\ 0)$
7	$(0.10,\ 0.62)$	$(0,\ 2^{-5})$	$(0,\ 2^{-12})$	$(0,\ 2^{-37})$
8	$(0.41,\ 0.80)$	$(0,\ 0.28)$	$(0,\ 2^{-4})$	$(2^{-39},\ 2^{-10})$
9	$(0.68,\ 0.90)$	$(0,\ 0.59)$	$(2^{-42},\ 0.33)$	$(2^{-11},\ 2^{-4})$
10	$(0.83,\ 0.95)$	$(2^{-46},\ 0.72)$	$(2^{-12},\ 0.63)$	$(0.07,\ 0.38)$
11	$(0.91,\ 0.97)$	$(2^{-13},\ 0.89)$	$(0.06,\ 0.81)$	$(0.37,\ 0.66)$

Table 4: Correctness Estimates. Each entry contains two probabilities of failing to bootstrap correctly. The first is the probability that the ciphertext c_{out} yields an incorrect output. The second is that the ciphertext $\mathbf{c}_{\mathsf{in}}$ is erroneous due to noise from a previous bootstrapping, key switching, and modulus switching. Given the variance of a random variable, we calculate the probability of failure by the erf function. Remind that t is the plaintext modulus, and the percentage gives the contribution of the rounding error $\mathbf{c}_{\mathsf{in}}$.

Set	uSVP	Avg β	λ
NTRU-ν-um-11-B	178.7	360.84	136
NTRU-ν-um-12-B	319.6	361.0	137
NTRU-ν-um-13-B	710.2	1068.0	344
NTRU-ν-um-14-B	1575.2	3048.68	923

Table 5: Security Estimations for the DSPRA-assumption (NTRU) and RLWE.The second column gives the cost estimations of usvp against the RLWE assumption (this is the lowest cost that we obtained). β is the BKZ-block size against DSPRA. λ gives us the estimated security parameter. To summarize, our parameters give larger security than we need allowing us to increase the modulus P if necessary.

procedure. The key switching key is instantiated over the smaller modulus Q. When choosing the LWE parameters for the key switching key, we observe that the LWE dimension n is one of the most crucial parameters for the system's performance. Hence we try to minimize n. But, on the other hand, we need to take the modulus Q large enough to fit the key switching error. Furthermore, we take the decomposition base for the key switching key to be 2 across all parameter sets. However, we note that choosing a larger decomposition base for key switching would cost us either valuable precision, or a larger modulus Q but smaller key size overall. We decided to go with larger keys.

5.1 Estimating Security.

Assumptions in This and Concurrent Work. Let us remind that we do not publish a public key in the form of an element $c = g/f$ is the case for LTV [54] and YASHE [14]. Crucially, note that if we would publish a public key $h = g/f$, and encrypt a message as $c = e \cdot g/h + m$, then for two ciphertexts encrypting the same message, an adversary could easily compute $(c_1 - c_2)/h = (e_1 - e_2)$ which is small. In our case, the attack would be disastrous because the adversary would be allowed to query the bits of the LWE secret key encrypted in the bootstrapping key. But in our scheme, like in the concurrent work [13], h is never published. The work [13] does not use g as we do. Specifically their NTRU ciphertext takes the form $c_i = g_i/f + m_i$, where g_i is from the same distribution as f, and is freshly chosen for each ciphertext. Hence we cannot reduce the problem to the standard DSPRA-assumption. From the decision small polynomial ratio assumption, we have that a ciphertext in the form $h = g/f$ is indistinguishable from uniform random. However, it is not entirely clear whether a sequence $c_1 = e_1/f, \ldots, c_2 = e_m/f$, where $e_1, \ldots, e_m$ are sampled independently and $1/f$ is reused across all $c_1, \ldots, c_m$ is indinsinghishable from random as well. To summarize, it seems that [13] must assume a modified decision small polynomial ratio assumption 2 to argue indistinguishability.

We choose our parameters more conservatively and add another layer of scrambling with the e_1 error and the e_2 error. This way we can argue security, by arguing for each ciphertext $c_i = e_1 g_i/f + e_2$ that the parts g_i/f are chosen independently at random from the decision small polynomial ratio assumption (Definition 2). Note that it is sufficient to use just one g across all ciphertexts, but we chose it freshly as an additional defense. The DSPRA-assumption states indistinguishability when all the g_i's are equivalent. Then from the RLWE assumption [59, 55] for the ring $\mathbb{Z}_P[X]/(X^N + 1)$, with N-power-of-two, we can argue indistinguishability of c_i even if the parts g_i/f would leak and would reuse the same g_i's. Finally, we stress that the argument above is a standard argument, and for formal proof, we refer to [60]. For completeness we recall the proof in Appendix 8 in the full version [49].

Estimating Security of the Parameters. To estimate security for the (R)LWE samples used the LWE estimator [6], but we note that for our parameters, the dimension of the rings is so large that the RLWE security is much above the 128-bit level. The security bottleneck lies in the key-switching key, which has a relatively small dimension. Below we discuss how we estimate the security for NTRU. However, we observe that the estimated security is far above 128-bits due to the large ring dimensions.

For NTRU, we consider two types of attacks. The first is a direct attack on the NTRU secret key that we call the SKR event (Secret Key Recovery). In this case, we consider recovering a vector of the short lattice basis as a successful attack against NTRU. The second event is recovering a basis vector of the dense sublattice contained in the NTRU lattice. We call this event the DSD event (Dense Sublattice Discovery). For more details on the attacks we refer to [33]. It has been observed that recovering the dense sublattice is much more efficient when the ratio between the modulus P and the ring dimension is large. We call the ratio between P and N after which the event of finding a basis vector of the sense sublattice is more likely than finding a vector as short as the secret key vector of the fatigue point. Two concurrent works by Albrecht, Bai, and Ducas [5], and Cheon, Jeong, and Lee [24] initiate the study of the so-called overstretched regime[3]. Kirchner and Fouque [48] improve their results and set the asymptotic fatigue point at $N^{2.783+o(1)}$. Very recently, Ducas and van Woerden [33] give a tighter prediction and set the fatigue point at $P = N^{2.484+o(1)}$. Importantly, Ducas and van Woerden [33] predict the hardness of the decisional small polynomial ratio problem in the overstretched regime. Namely, they predict the hardness of the DSD event when the modulus P exceeds the fatigue point. Moreover, they back up their prediction with extensive experiments. At Table 5 we give the results of running the estimator from [33][4] and form [6]. Remind that our ciphertexts assume NTRU and RLWE. For RLWE we report on the uSVP cost as returned by the estimator. We note that this is the lowest estimated attack cost.

Below, we describe the estimations for the DSPRA-assumption. The NTRU estimator gives us the block size β at which Progressive BKZ detects the SKR or DSD event. For all our parameter sets, we obtained the DSD event first, meaning that running BKZ with block size β, we get the DSD event with a probability close to 1. On the other hand, the SKR event for the given β was close to 0. Hence at table 5, we list the parameters necessary to brake NTRU by obtaining a DSD event.

Based on thhe BKZ-block size β and the ring dimension N, we estimate the cost of running the lattice reduction by the cost model from [10]. Namely, we estimate the cost in (brute force-equivalent bits) by

$$\lambda = 0.292 \cdot \beta + 16.4 + \log(8 \cdot N).$$

Despite operating in the overstretched regime, we note that our dimensions N are so high that our parameters have much larger security than necessary. For example, for NTRU-ν-um-C-13-B, the NTRU instances are estimated to have 405-bits of security! For the parameters with the smallest dimension, we get 136-bits of security, which is already much higher than our 128-bit goal.

5.2 Correctness of the Parameter Sets.

Below we show correctness estimates for plaintext spaces $\mathbb{Z}_{t_1}$ and $\mathbb{Z}_{t_2}$, for t_1 and $t_2 = 2^4, \ldots, 2^{11}$. Our estimates are depicted at Table 4. The sk row gives the share of variance from the rounding when modulus switching. Specifically, we calculate what percentage of the total variance of $\mathbf{c}_{\mathsf{in}}$ consists of the variance of the rounding error. The reason to point out the rounding's share is to give an intuitive limit on the ciphertext space's size for a given ring dimension and without sparse secret keys.

[3]Although the first attack is due to Gentry and Szydlo [39]

[4]We slightly modified the estimator code to allow us to estimate security for larger dimensions.

$\approx N$	NTRU-ν-um	[50]	[61]	[53]
2^{11}	7644	29400	28672	-
2^{12}	6000	34300	12672	15660
2^{13}	6616	44100	-	-
2^{14}	7216	-	-	-

Table 6: The number of the FFT/NTTs for NTRU-ν-um. We take only the fastest parameters from previous work.

Set	BR [s]	KS [s]	Ksk [MB]	Bk [MB]	ct [KB]
11-B	0.09	0.01	130.08	25.97	8.15
12-B	0.14	0.03	507.18	55.25	20.46
13-B	0.32	0.06	1152.68	121.90	40.94
14-B	0.81	0.2	2662.56	265.96	81.90

Table 7: Performance of NTRU-ν-um. Columns BR and KS refer to blind rotation and key switching respectively. The columns Ksk, Bk give the evaluation key sizes, and ct gives the size of the ciphertext.

6 Implementation and Performance

We implement [3] and test NTRU-ν-um in C++ using the fftw library [35] to compute fast Fourier transforms and Intel HEXL [11] to compute number theoretic transforms. We use FFT's for the NTRU-ν-um with ring size 2^{11}. For all other parameter sets, we use the NTT to compute negacyclic convolutions. Let us briefly describe the main loop of the bootstrapping algorithm. In each iteration, we perform a rotation of the coefficients of c_{acc}, subtraction, addition, and a gadget multiplication. Clearly, gadget multiplication is the most expensive operation. Similarly, as in previous implementations, we precompute the FFT/NTT's of the polynomials of the blind rotation keys. That is, the polynomials are stored in the evaluation form. The accumulator is stored in the coefficient form. Hence, for every gadget multiplication we decompose the NTRU sample $c_{\mathsf{acc}} \cdot X^{\mathbf{c}_{\mathsf{in}}[i+1]} - c_{\mathsf{acc}}$, and we compute a FFT/NTT for every element of the decomposition. Then we compute a multisum with the corresponding bootstrapping key in the evaluation form and compute an inverse FFT/NTT.

We give a brief theoretical comparison of the FFT/NTT's required to compute a bootstrapping between our scheme, FDFB [50] and [61, 53]. The comparison is on Table 6. We limit ourselves only to a theoretical comparison, for the following reasons. All papers [50, 53, 61] provide parameters and report timings on their implementations. Nevertheless, we find that comparing the implementation times may be difficult and unfair. For example FDFB [50, 53] are implemented on top of Palisade, which timings may vary depending on whether hardware acceleration is used or not. Furthermore, some parameter sets like [53] use a 54-bit modulus, whereas we have only 45 bits, what allows them to choose much larger decomposition bases and reduce the number of NTT's. As noted earlier, our ring dimensions actually allow us to increase our modulus. We note that all the schemes can be implemented using the same arithmetic. We can calculate the number of FFT/NTT's in our blind rotation by $n \cdot (\ell_{\mathsf{Bk}} + 1)$. In contrast for the blind rotation used in previous work, that is based on RLWE, we have $2 \cdot n \cdot (\ell_{\mathsf{Bk}} + 1)$. Therefore, we expect of achieve a $2\times$ speedup over prior work. As we can see, NTRU-ν-um appears to achieve a major speedup in terms of the number of FFT/NTT's. For example, our binary set for $N \approx 2^{12}$ is approximately 2.12 times faster than [61, 53] and 5.9 faster than [50]. What is worth noting is that [61, 53] and [50] also use binary keys. Finally, we

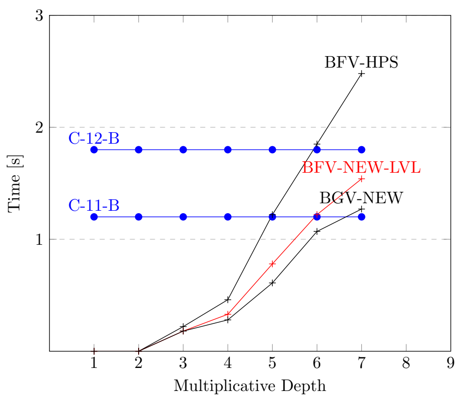

Figure 3: Comparison for timing between NTRU-ν-um and the BGV/BFV to compute $\prod_{i=1}^{k} a_i x^i$. Remind that all BGV/BFV are leveled and do not include bootstrapping and modulus raising.

tested our implementation on a commodity laptop with an Intel(R) Core(TM) i7-11850H 2.50GHz processor and 32.0 GB RAM. We performed all tests on a single core. Table 7 shows the average timing of a single bootstrapping operation and the size of the key material.

6.1 Applications.

This section describes several example applications and reports on timings for those applications.
Computing Univariate Functions over Finite Fields. We show how NTRU-ν-um compares to BGV/BFV-type schemes [17, 19, 18, 34] when computing univariate functions in finite fields. In particular, we take the function $\prod_{i=1}^{k} a_i x^i$. The comparison is given by Figure 3 with the recent results from Kim, Polyakov, and Zucca [46] implemented in PALISADE [2]. There are a few important observations. First, NTRU-ν-um evaluates an arbitrary univariate function over $\mathbb{Z}_t$ in time independent of the depth of the function. Second, the experiments in [46] assume that $|a_i| < 2^4$. The coefficients are much smaller than the plaintext modulus, which is around 16-bits. For larger coefficients, we would need to adjust the parameters for BGV/BFV and take a larger ciphertext modulus. In contrast, NTRU-ν-um is independent of the size of the coefficients. However, to compute correctly for larger plaintext spaces using our parameter sets, we need to represent the plaintexts in the CRT form. Note that a similar optimization is made in [46]. In contrast to [46], NTRU-ν-um outputs an already bootstrapped (noise reduced) ciphertext, which allows an evaluator to resume computation immediately. For BGV/BFV-type schemes to resume computation, we need to raise the modulus and bootstrap the resulting ciphertexts [41, 22, 42]. On the other hand, when an NTRU-ν-um bootstrapping invocation is the last, and the ciphertext is returned to the client, we can take a much larger precision.
Homomorphic Gaussian Elimination. The next application is to compute the Gaussian elimination algorithm. Let us remind that Gaussian elimination requires computing the multiplicative inverse of field elements. Importantly, with BGV/BFV techniques, while it is theoretically possible to compute any polynomial-size arithmetic circuit, in practice evaluating the Extended Euclidean algorithm is a very costly operation. With NTRU-ν-um, we compute multiplicative inverses in just one bootstrapping operation. Figure 4 presents the results of computing Gaussian elimination for several dimensions and parameter sets.

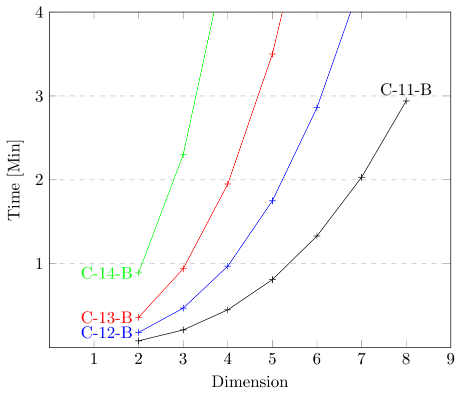

Figure 4: Timings for solving a system of linear equations via the Gaussian elimination algorithm.

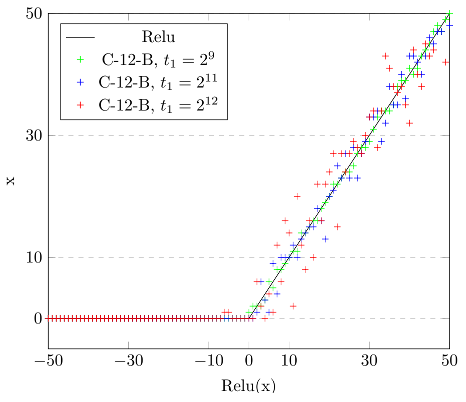

Figure 5: Error for approximate “bootstrapping” of the Relu.

Computing Approximate Functions. We can compute in an approximate mode by taking t_1 even as large as the ring dimension. In this case, a ciphertext that may have been the result of a previous bootstrap operation is modulus reduced to $\mathbb{Z}_N$. Hence if $t_1 = N$, we have essentially no rounding of the ciphertext just like in CKKS [25]. This way, we can compute any approximate function with a single bootstrapping operation. This is very useful when evaluating neural networks on encrypted queries. Neural networks can be seen as circuits with gates of the form $F(b+\sum_{i=1}^{m} a_i \cdot x_i)$ where $[a_i]_{i=1}^{m}$ are called weights and b is the bias. When using NTRU-ν-um we compute the affine part $b + \sum_{i=1}^{m} a_i \cdot x_i$ at nearly no cost. In practice, the size of the affine function rarely exceeds $m \leq 1000$. Hence the computing time is dominated by computing the non-linear function F.

In CKKS-type schemes, to compute any function, we first need to approximate the function, for example, with a Taylor series, and then evaluate the approximation with a CKKS-style approximate homomorphic encryption. Furthermore, similarly to BGV/BFV, after computing a function with CKKS, we need to raise the modulus to resume computation, whereas for NTRU-ν-um we can compute immediately after bootstrapping. Raising the modulus in CKKS-type schemes is currently subject to extensive studies [23, 21, 43, 15, 51], as it is the major efficiency and accuracy bottleneck. For NTRU-ν-um the time to compute a function is as given by Figure 7. We find it instructive to showcase how the approximation error behaves for a functional bootstrapping algorithm like NTRU-ν-um. Notably, the approximation error is very different than the error for schemes like CKKS. As a demonstration, we depict in Figure 5 the result of bootstrapping the Relu function with an oversized plaintext modulus. The Relu function on input $x \in \mathbb{Z}$ output x if $x \geq 0$ and 0 otherwise. As we can see, the larger the size of the plaintext modulus, the larger the impact of the error on the outcome of the computation. What stands unique for NTRU-ν-um and other functional bootstrapping algorithms [27, 16, 20, 50, 61, 13] is that the error depends on the computed function. In the case of Relu, we can see that when the function is constant on a section of the domain, the error does not affect the outcome of the bootstrapping. The reason is as follows. The error may cause the homomorphic rotation of a_{rot} polynomial to be shifted. Suppose the shift sets the constant-coefficient to the same value as for correct computation. In that case, the bootstrapping procedure will output the correct value conditioned that the bootstrapping error does not distort it, but as we can see from Table 4 the bootstrapping error and size of the modulus P allow for a much larger plaintext modulus. Finally, note that as all existing approximate HE schemes, NTRU-ν-um in approximate mode will not be CPA^D-secure [52].

7 Conclusions

We showed that it is possible and advantageous to build fully homomorphic encryption secure in the "overstretched" modulus regime. What is important to note is that the functional bootstrapping technique might be applied in combination with BGV/BFV and CKKS schemes. In particular, it seems that while multiplying two field elements for NTRU-ν-um is possible with two bootstrapping operations, multiplication without bootstrapping is much faster in BGV/BFV. Furthermore, it may be that for the ring dimensions, it may be possible to provide a secure instantiation of LTV and YASHE that could support a small number of multiplications.

Acknowledgements This work has been partially funded/supported by the German Ministry for Education and Research through funding for the project CISPA-Stanford Center for Cybersecurity (Funding number: 16KIS0927).

References

[1] Ieee standard specification for public key cryptographic techniques based on hard problems over lattices. *IEEE Std 1363.1-2008*, pages 1–81, 2009.

[2] PALISADE Lattice Cryptography Library (release 1.11.5). https://palisade-crypto.org/, 2021.

[3] FHE-Deck. https://github.com/FHE-Deck, 2022.

[4] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the nist post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2020.

[5] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 153–178. Springer, Heidelberg, August 2016.

[6] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[7] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Heidelberg, August 2013.

[8] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, Heidelberg, August 2014.

[9] X9 ANSI. 98: Lattice-based polynomial public key establishment algorithm for the financial services industry. Technical report, Technical report, ANSI, 2010.

[10] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24. ACM-SIAM, January 2016.

[11] Fabian Boemer, Sejun Kim, Gelila Seifu, Fillipe DM de Souza, Vinodh Gopal, et al. Intel HEXL (release 1.2). https://github.com/intel/hexl, 2021.

[12] Guillaume Bonnoron, Léo Ducas, and Max Fillinger. Large FHE gates from tensored homomorphic accumulator. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 18: 10th International Conference on Cryptology in Africa*, volume 10831 of *Lecture Notes in Computer Science*, pages 217–251. Springer, Heidelberg, May 2018.

[13] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. FINAL: Faster FHE instantiated with NTRU and LWE. Cryptology ePrint Archive, Report 2022/074, 2022. https://eprint.iacr.org/2022/074.

[14] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and Coding*, pages 45–64, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[15] Jean-Philippe Bossuat, Christian Mouchet, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 587–617. Springer, Heidelberg, October 2021.

[16] Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 483–512. Springer, Heidelberg, August 2018.

[17] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, Heidelberg, August 2012.

[18] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325. Association for Computing Machinery, January 2012.

[19] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE Computer Society Press, October 2011.

[20] Sergiu Carpov, Malika Izabachène, and Victor Mollimard. New techniques for multi-value input homomorphic evaluation and applications. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, volume 11405 of *Lecture Notes in Computer Science*, pages 106–126. Springer, Heidelberg, March 2019.

[21] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 34–54. Springer, Heidelberg, May 2019.

[22] Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved FHE bootstrapping. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 315–337. Springer, Heidelberg, April / May 2018.

[23] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 360–384. Springer, Heidelberg, April / May 2018.

[24] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016.

[25] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer, Heidelberg, December 2017.

[26] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Heidelberg, December 2016.

[27] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 377–408. Springer, Heidelberg, December 2017.

[28] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020.

[29] Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 670–699, Cham, 2021. Springer International Publishing.

[30] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 325–340. Springer, Heidelberg, February / March 2016.

[31] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05: 3rd International Conference on Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer, Heidelberg, June 2005.

[32] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, Heidelberg, April 2015.

[33] Léo Ducas and Wessel van Woerden. Ntru fatigue: How stretched is overstretched? In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 3–32, Cham, 2021. Springer International Publishing.

[34] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

[35] Matteo Frigo and Steven G. Johnson. Fftw. https://www.fftw.org, 2021.

[36] Nicholas Genise, Craig Gentry, Shai Halevi, Baiyu Li, and Daniele Micciancio. Homomorphic encryption for finite automata. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part II*, volume 11922 of *Lecture Notes in Computer Science*, pages 473–502. Springer, Heidelberg, December 2019.

[37] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009.

[38] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A.

Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, Heidelberg, August 2013.

[39] Craig Gentry and Michael Szydlo. Cryptanalysis of the revised NTRU signature scheme. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 299–320. Springer, Heidelberg, April / May 2002.

[40] Antonio Guimarães, Edson Borin, and Diego F. Aranha. Revisiting the functional bootstrap in tfhe. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(2):229–253, Feb. 2021.

[41] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670. Springer, Heidelberg, April 2015.

[42] Shai Halevi and Victor Shoup. Bootstrapping for HElib. *Journal of Cryptology*, 34(1):7, January 2021.

[43] Kyoohyung Han and Dohyeong Ki. Better bootstrapping for approximate homomorphic encryption. In Stanislaw Jarecki, editor, *Topics in Cryptology – CT-RSA 2020*, volume 12006 of *Lecture Notes in Computer Science*, pages 364–390. Springer, Heidelberg, February 2020.

[44] Ryo Hiromasa, Masayuki Abe, and Tatsuaki Okamoto. Packing messages and optimizing bootstrapping in GSW-FHE. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 699–715. Springer, Heidelberg, March / April 2015.

[45] 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.

[46] Andrey Kim, Yuriy Polyakov, and Vincent Zucca. Revisiting homomorphic encryption schemes for finite fields. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 608–639, Cham, 2021. Springer International Publishing.

[47] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, Heidelberg, May 1999.

[48] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 3–26. Springer, Heidelberg, April / May 2017.

[49] Kamil Kluczniak. Ntru-ν-um: Secure fully homomorphic encryption from ntru with small modulus. Cryptology ePrint Archive, Paper 2022/089, 2022. https://eprint.iacr.org/2022/089.

[50] Kamil Kluczniak and Leonard Schild. FDFB: Full domain functional bootstrapping towards practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2021/1135, 2021. https://eprint.iacr.org/2021/1135.

[51] Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. High-precision bootstrapping of RNS-CKKS homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 618–647. Springer, Heidelberg, October 2021.

[52] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 648–677. Springer, Heidelberg, October 2021.

[53] Zeyu Liu, Daniele Micciancio, and Yuriy Polyakov. Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping. Cryptology ePrint Archive, Report 2021/1337, 2021. https://eprint.iacr.org/2021/1337.

[54] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1219–1234. ACM Press, May 2012.

[55] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, Heidelberg, May / June 2010.

[56] Daniele Micciancio and Yuriy Polyakov. *Bootstrapping in FHEW-like Cryptosystems*, page 17–28. Association for Computing Machinery, New York, NY, USA, 2021.

[57] Daniele Micciancio and Jessica Sorrell. Ring packing and amortized FHEW bootstrapping. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018: 45th International Colloquium on Automata, Languages and Programming*, volume 107 of *LIPIcs*, pages 100:1–100:14. Schloss Dagstuhl, July 2018.

[58] Jacques Patarin. The oil and vinegar signature scheme. In *Dagstuhl Workshop on Cryptography September, 1997*, 1997.

[59] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005.

[60] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 27–47. Springer, Heidelberg, May 2011.

[61] Zhaomin Yang, Xiang Xie, Huajie Shen, Shiying Chen, and Jun Zhou. TOTA: Fully homomorphic encryption with smaller parameters and stronger security. Cryptology ePrint Archive, Report 2021/1347, 2021. https://eprint.iacr.org/2021/1347.

8 Omitted Proofs

Below we give the full versions of the lemmas and proves that were omitted in the main body of the paper.

Lemma 8 (Correctness of Gadget Multiplication for General Messages) *If* $c_{\mathsf{out}} = \mathsf{GMul}(\mathbf{c}_G, c)$ *and* $m_G \in \mathcal{R}_Q$*, then* $c_{\mathsf{out}} = \mathsf{NTRU}_{\sigma_{\mathsf{out}}}(f,\, c \cdot m_G)$*, where*

$$\sigma^2_{\mathsf{out}} = N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}^2_{\mathsf{Bk}}/4 \cdot \sigma^2_{\mathsf{Bk}}.$$

If additionally $c = \mathsf{NTRU}_\sigma(f, m)$ *and* $||m_G||_\infty \leq B$*, then*

$$\sigma^2_{\mathsf{out}} \leq N \cdot B^2 \cdot \sigma^2 + N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}^2_{\mathsf{Bk}}/4 \cdot \sigma^2_{\mathsf{Bk}}.$$

Proof 8 *The proof is as the proof of Lemma 5, but we only need to estimate the noise magnitude when multiplying with an NTRU ciphertext and where* $||m_G||_\infty \leq B$*. Similarly as before we have*

$$\begin{aligned} c' &= e'_1/f + e'_2 + r' + m_G \cdot c \\ &= e'_1/f + e'_2 + r' + m_G \cdot (e_1/f + e_2 + r + m) \\ &= (e'_1 + m_G \cdot e_1)/f + e'_2 + m_G \cdot (e_2 + r) + m_G \cdot m. \end{aligned}$$

To summarize we have

$$\sigma^2_{\mathsf{out}} \leq N \cdot B^2 \cdot \sigma^2 + N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}^2_{\mathsf{Bk}}/4 \cdot \sigma^2_{\mathsf{Bk}}$$

because we multiply the Gaussian noise by m_G *that has infinity norm bounded by* B*. Furthermore, we have* $\mathsf{U}(c_{\mathsf{out}}) \leq N \cdot B \cdot \mathsf{U}(\sigma) + N \cdot \ell_{\mathsf{Bk}} \cdot \mathsf{L}_{\mathsf{Bk}}/2 \cdot \mathsf{U}(\mathbf{c}_G)$.

Lemma 9 (Modulus Switching for LWE) *Let us define the following ciphertext* $\mathbf{c} = \mathsf{LWE}_\sigma\big(\mathbf{s}, \frac{Q}{t} \cdot m\big)$*, where* $\mathbf{s} \in \mathbb{Z}^n_Q$*. Let us define the LWE modulus switching procedure as*

$$\mathsf{ModSwitch}(\mathbf{c}, q) = \Big\lfloor \big[\mathbf{c} \cdot \frac{q}{Q}\big]_{i=1}^{n+1} \Big\rceil$$

for $q \leq Q$*.*

If $\mathbf{c}_{\mathsf{out}} \leftarrow \mathsf{ModSwitch}(\mathbf{c}, q)$*, then* $\mathbf{c}_{\mathsf{out}} = \mathsf{LWE}_{\sigma_{\mathsf{out}}}\big(\mathbf{s}, \frac{q}{t} \cdot m\big)$*, where*

$$\sigma^2_{\mathsf{out}} \leq \Big(\frac{q}{Q} \cdot \sigma\Big)^2 + \mathsf{Ha}(\mathbf{s}) \cdot \mathsf{Var}(s)$$

and $\mathsf{U}(c) \leq 1$*.*

Proof 9 *Let* $\mathbf{c}[1] + \mathbf{c}[2:n+1]^\top \cdot \mathbf{s} = e + \Delta_{Q,t} \cdot m$*. From definition we have*

$$\begin{aligned} &\mathbf{c}_{\mathsf{out}}[1] + \mathbf{c}_{\mathsf{out}}[2:n+1]^\top \cdot \mathbf{s} \\ &\qquad = \lfloor \frac{q}{Q} \cdot \mathbf{c}[1] \rceil + \lfloor \frac{q}{Q} \cdot \mathbf{c}_{\mathsf{out}}[2:n+1]^\top \cdot \mathbf{s} \rceil \\ &\qquad = \frac{q}{Q} \cdot \mathbf{c}[1] + r + \frac{q}{Q} \cdot \mathbf{c}_{\mathsf{out}}[2:n+1]^\top \cdot \mathbf{s} + \mathbf{r}^\top \cdot \mathbf{s} \\ &\qquad = \frac{q}{Q} \cdot (\mathbf{c}[1] + \mathbf{c}_{\mathsf{out}}[2:n+1]^\top \cdot \mathbf{s}) + r + \mathbf{r}^\top \cdot \mathbf{s} \\ &\qquad = \frac{q}{Q} \cdot \frac{Q}{t} \cdot m + \frac{q}{Q} \cdot e + r + \mathbf{r}^\top \cdot \mathbf{s} \\ &\qquad = \frac{q}{t} \cdot m + \frac{q}{Q} \cdot e + r + \mathbf{r}^\top \cdot \mathbf{s} \end{aligned}$$

where $r \in [-1/2, 1/2]$ and $\mathbf{r} \in [-1/2, 1/2]^n$. We assume that r and $\mathbf{r}$ are uniform random over their support. We upper-bound the infinity norm of r and $\mathbf{r}$ by 1. Hence $\mathbf{c}_{\mathsf{out}} = \mathsf{LWE}_{e_{\mathsf{out}}}(\mathbf{c}, \frac{q}{t} \cdot m)$, where $e_{\mathsf{out}} = \frac{q}{Q} \cdot (\epsilon \cdot t + e) + r + \mathbf{r}^\top \cdot \mathbf{s}$. Therefore, $\mathsf{Var}(e_{\mathsf{out}}) \leq \mathsf{Var}(\frac{q}{Q} \cdot e) + 1 + \mathsf{Var}(\mathbf{r}^\top \cdot \mathbf{s})$. We set $n - \mathsf{Ha}(\mathbf{s})$ coordinates of $\mathbf{s}$ to zero, so we get

$$\begin{aligned}\mathsf{Var}(\mathbf{r}^\top \cdot \mathbf{s}) &= \sum_{i=1}^{\mathsf{Ha}(\mathbf{s})} \mathsf{Var}(\mathbf{r}[i]) \cdot (\mathsf{Var}(s) + \mathsf{E}(s)^2) \\ &= \mathsf{Ha}(\mathbf{s}) \cdot (\mathsf{Var}(s) + \mathsf{E}(s)^2)\end{aligned}$$

To summarize we have

$$\mathsf{Var}(e_{\mathsf{out}}) = (\frac{q}{Q})^2 \cdot \mathsf{Var}(e) + \mathsf{Ha}(\mathbf{s}) \cdot \mathsf{Var}(s)$$

and $\mathsf{U}(c_{\mathsf{out}}) \leq 1$.

Definition 6 (Ring Learning With Errors) *Let $s \in \mathcal{R}_Q$ be a secret key with coefficients form some distribution $\mathcal{X}_{\mathsf{sk}}$. We define a RLWE sample of a message $m \in \mathcal{R}_Q$ as $\mathbf{c} = \mathsf{RLWE}_\sigma(s, m) \in \mathcal{R}_Q$ where $\mathbf{c}[1] = -\mathbf{c}[2] \cdot s + e + m$, and $\mathbf{c}[2]$ is chosen from the uniform distribution over $\mathcal{R}_Q$. We define the phase of $\mathbf{c}$ as $\mathsf{Phase}(\mathbf{c}) = \mathbf{c}[1] + \mathbf{c}[2] \cdot s$. We define the learning with error distribution $\mathsf{LWE}_{\mathcal{R}_Q, \mathcal{X}_{\mathsf{sk}}, \sigma}$, to consist of LWE samples of zero, as defined above.*

The ring learning with error assumption states that it is hard to distinguish elements sampled from $\mathsf{RLWE}_{\mathcal{R}_Q, \mathcal{X}_{\mathsf{sk}}, \sigma}$ and elements sampled uniformly at random over $\mathcal{R}_Q^2$.

Theorem 2 (Security of NTRU-ν-um.) *NTRU-ν-um is IND-CPA secure assuming the $\mathsf{DSPRA}_{\mathcal{R}_Q}$, the $\mathsf{RLWE}_{\mathcal{R}_Q, \mathcal{X}_{\mathsf{sk}}, \sigma}$ and $\mathsf{LWE}_{n, \sigma', \mathcal{X}'_{\mathsf{sk}}}$ assumptions hold over $\mathcal{R}_Q$.*

Proof 10 *Note that the FHE scheme consists of the blind rotation key Bk and the key switching key Ksk. Clearly as the Ksk key consists of LWE samples, the key is indistinguishable from uniform from the LWE assumption. The Bk key and ciphertexts of the clients are NTRU samples. Hence what is left to show, is that NTRU samples are indistinguishable form uniform. Remind that an NTRu sample of zero is of the form $c = e_1 \cdot g/f + e_2 + r$. From the $\mathsf{DSPRA}_{\mathcal{R}_Q}$ assumption we have that g/f is indistinguishable from a uniformly sampled element $u \in \mathcal{R}_Q$. Hence the sample c is indistinguishable from a sample $c' = e_1 \cdot u + e_2 + r$. Finally, note that the tuple (u, c') constitutes a RLWE sample of r with secret key e_1 and error e_2. Nevertheless, form RLWE assumption we have that $e_1 \cdot u + e_2$ is indistinguishable from a uniformly sampled element $u' \in \mathcal{R}_Q$. Hence we have $(u, u' + r)$ which is distributed as (u, u'') with u''-uniform in $\mathcal{R}_Q$. To summarize, the tuple (u, c') is indistinguishable from (u, u'') assuming $\mathsf{RLWE}_{\mathcal{R}_Q, \mathcal{X}_{\mathsf{sk}}, \sigma}$.*