FINAL: Faster FHE instantiated with NTRU and LWE

Charlotte Bonte[1] ⋆, Ilia Iliashenko[2], Jeongeun Park[2], Hilder V. L. Pereira[2], and Nigel P. Smart[2,3]

[1] Intel Corporation, Emerging Security Lab
[2] imec-COSIC, KU Leuven, Leuven, Belgium
[3] Zama Inc.
charlotte.bonte@intel.com, ilia@esat.kuleuven.be, Jeongeun.Park@esat.kuleuven.be, HilderVitor.LimaPereira@esat.kuleuven.be, nigel.smart@kuleuven.be

Abstract. The NTRU problem is a promising candidate to build efficient Fully Homomorphic Encryption (FHE). However, all the existing proposals (e.g. LTV, YASHE) need so-called 'overstretched' parameters of NTRU to enable homomorphic operations. It was shown by Albrecht et al. (CRYPTO 2016) that these parameters are vulnerable against subfield lattice attacks.
Based on a recent, more detailed analysis of the overstretched NTRU assumption by Ducas and van Woerden (ASIACRYPT 2021), we construct two FHE schemes whose NTRU parameters lie outside the overstretched range. The first scheme is based solely on NTRU and demonstrates competitive performance against the state-of-the-art FHE schemes including TFHE. Our second scheme, which is based on both the NTRU and LWE assumptions, outperforms TFHE with a 28% faster bootstrapping and 45% smaller bootstrapping and key-switching keys.

Keywords: NTRU, FHE, LWE, bootstrapping

1 Introduction

In the last ten years fully homomorphic encryption based on lattice problems has been a vibrant field of research, with schemes being proposed, sometimes broken and sometimes improved. The initial work of Gentry [18] was truly groundbreaking in that it established not only (what we now call) a compact somewhat homomorphic encryption (SHE) scheme based on lattices, but it also presented a method to bootstrap the compact SHE scheme into a fully homomorphic encryption (FHE) scheme. Gentry's original scheme was based on properties of lattices of ideals of algebraic number fields, which are now considered insecure, but in the intervening years numerous authors have presented FHE schemes based on LWE [6], Ring-LWE [7], NTRU [27] and the approximate integer GCD problem [31].

⋆ Work done while at imec-COSIC, KU Leuven.

NTRU-based schemes seem the most efficient as their ciphertexts can be represented by a single polynomial in comparison to a pair of polynomials in RLWE-based schemes. Hence, these schemes have the potential of halving both the memory requirements and the running time.

In particular, an early FHE scheme based on the NTRU problem, called YASHE [4], was very efficient when compared to similar schemes. However, it was subsequently shown to be insecure due to the parameters being chosen in the so-called 'overstretched' NTRU regime [1]. More specifically, YASHE required the integer modulus q to be exponentially large in n, the degree of the polynomial used as the modulus of the polynomial ring. However, it was discovered [1] that as we "stretch" the parameters by increasing q for a fixed n, the NTRU problem becomes easier, because it becomes possible to exploit a dense sublattice of the NTRU lattice to mount an attack. Therefore, constructing FHE schemes based on the NTRU problem is challenging.

In the initial version of the attack, the subfields of the NTRU field were exploited in order to reduce the dimension of the lattice in which one searches for the secret key. Latter analysis [24] showed that the attack is enabled simply by the existence of a dense sublattice within the NTRU lattice. Thus, this attack stems from the structure of the NTRU lattice and, therefore, cannot be addressed by switching to another polynomial ring. However, it was still difficult to estimate the impact of these sublattice attacks on the security of NTRU and, therefore, it was hard to obtain correct estimates of the security level of NTRU-based schemes. For example, the recent leveled homomorphic encryption scheme for automata [16], which is based in the matrix NTRU problem, used q polynomial in n. Nevertheless, it was quickly shown [26] that this scheme is vulnerable to sublattice attacks.

However, we now have a much better understanding about how the security of the NTRU problem degrades as we increase q. In particular, the recent work of Ducas and van Woerden [13] allows us to estimate the concrete cost of breaking the NTRU problem for any given q. Thus, we now have much more solid ground to try to construct NTRU-based FHE schemes.

Ducas and van Woerden showed that to avoid the aforementioned sublattice attacks one should set $q \in O(n^{2.484})$. This already seems to rule out NTRU-based schemes which follow the blueprint BGV [5] or FV [14]. Therefore, as a starting point, we take the bootstrapping of [3], which is the basis of the FHEW scheme [12] and its extension, TFHE [10].

These schemes have a base homomorphic encryption scheme, in both cases based on LWE, and an accumulator, which is a variant of the GSW scheme [19], instantiate with the RLWE problem. We use the base scheme to evaluate binary gates and the accumulator to refresh the LWE ciphertexts of the base scheme.

The advantage of GSW-like schemes is that noise growth is quasi additive when evaluating long chains of multiplications, thus, the final noise in the refreshed ciphertext can be as small as $\tilde{O}(n)$, which fits the above bound of Ducas and van Woerden.

In this paper, we investigate the construction of FHE schemes based on the NTRU. We show that it is possible to adapt the framework of FHEW [12] to the NTRU setting, by using a matrix version of the NTRU problem to construct the base scheme and the standard NTRU problem to construct a GSW-like scheme. The resulting scheme has a fast bootstrapping algorithm with running times similar to those of the most efficient scheme of this type – TFHE [10]. As the encryption parameters of our scheme can be selected outside of the overstretched regime of NTRU, this allows us to construct competitive FHE based solely on the NTRU assumption. In other words, our result is a positive answer to the open problem of whether it is possible to construct FHE based on NTRU.

In addition, we show that by combining an LWE-based scheme and our NTRU-based GSW-like scheme, called NGS in this paper, we obtain a bootstrapping algorithm that is faster than TFHE's and requires much less key material, which improves the state-of-the-art in FHE constructions.

Concurrently to our own work Kluczniak [25] presented a version of NTRU called NTRU-ν-um which also claims to provide a secure fully homomorphic version of NTRU with small modulus. The scheme is presented to be instantiated over a ring defined by $X^N + 1$ and $X^N - 1$. In [22] Joye shows that the variant defined over $X^N - 1$ is not secure.

1.1 Our techniques and results

Homomorphic scheme based on the matrix NTRU problem. The bootstrapping framework of [12] assumes that the input encryption of the bootstrapping is an LWE ciphertext which means the main step of the decryption is a simple inner product between the ciphertext and the secret key. However, if we want to replace the underlying LWE-based base scheme with one based on NTRU then complications arise. The NTRU decryption involves a polynomial multiplication, which is much more complicated than the inner product.

One way of simplifying the decryption function is by assuming that each NTRU ciphertext encrypts an integer m_0 instead of a polynomial of degree $N-1$. Let R_q be a polynomial ring. We can encrypt m_0 as $c = g/f + \Delta \cdot m_0 \in R_q$ where g is a random element of R_q, f is the secret key and $\Delta \simeq q/4$. Note that we don't add any additional noise other than g in a ciphertext unlike other NTRU based schemes [27,4] in order to keep noise growth small as discussed in Section 3. To decrypt, we compute the inner product of the coefficient vector of c, denoted by $\boldsymbol{\phi}(c)$, and the first column of the anti-circulant matrix of f which we will denote as $\boldsymbol{\Phi}(f)$. Given that the secret key of the NTRU scheme will be defined as $f = 1 + 4 \cdot f'$, one can notice that in R_q $c \cdot f = g + 4 \cdot f' \cdot \epsilon + \Delta \cdot m_0$, for some small ϵ, which implies that

$$\boldsymbol{\phi}(c) \cdot \boldsymbol{\Phi}(f) = \boldsymbol{\phi}(g) + 4 \cdot \epsilon \cdot \boldsymbol{\phi}(f') + \Delta \cdot (m_0, 0, \ldots, 0).$$

Hence, $\boldsymbol{\phi}(c) \cdot \mathsf{col}_0(\boldsymbol{\Phi}(f)) = g_0 + 4 \cdot \epsilon \cdot f'_0 + \Delta \cdot m_0$, which is enough to recover m_0.

Similarly to [12], one can use NTRU defined over power-of-two cyclotomic rings. However, these rings provide little flexibility in terms of choosing parameters to achieve a certain security level. For example, even if the ring dimension

$N = 600$ already satisfies the desired security level, one has to choose $N = 2^{10}$ as this is the smallest power of 2 larger than 600. This problem can be solved by using cyclotomic rings of other orders instead of power-of-two, but in this case, the matrix $\boldsymbol{\Phi}(f)$ loses its anti-circulant property, which, as we will see in Section 5, helps to significantly speed up the bootstrapping and reduce encryption parameters.

Driven by the above limitations, we resort to the matrix NTRU problem (MNTRU) instead of its ring-based version. Our MNTRU base scheme is described as follows. We replace the polynomial ratio g/f by the matrix product $\mathbf{G} \cdot \mathbf{F}^{-1}$, where both $\mathbf{G}$ and $\mathbf{F}$ are unstructured random matrices. Hence, a ciphertext of some plaintext matrix $\mathbf{M}$ has the form $\mathbf{C} = \mathbf{G} \cdot \mathbf{F}^{-1} + \Delta \cdot \mathbf{M} \in \mathbb{Z}_q^{n \times n}$ or $\mathbf{C} = (\mathbf{G} + \Delta \cdot \mathbf{M}) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^{n \times n}$. A single integer $m \in \{0, 1\}$ is encrypted by a ciphertext of the form

$$\mathbf{c} := (\mathbf{g} + \Delta \cdot \mathbf{m}) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$$

where $\mathbf{g}$ is a random vector from $\mathbb{Z}_q^n$ and $\mathbf{m} := (m, 0, \ldots, 0) \in \mathbb{Z}^n$. This guarantees that the decryption can be done by the inner product of $\mathbf{c}$ and the first column of the secret matrix $\mathbf{F}$. Therefore, it is simple enough for the bootstrapping algorithm to handle it efficiently. Furthermore, it is easy to adapt the homomorphic NAND gate from [12] to our scheme.

Notice that we are dividing both the noise term ($\mathbf{g}$) and the message ($\Delta \cdot m$) by $\mathbf{F}$, because this reduces the impact of the noise growth due to the multiplication by $\mathbf{F}$ during the decryption, as explained in Section 3.

GSW-like scheme based on the NTRU problem. The bootstrapping framework of FHEW [12] uses a GSW-like scheme based on the RLWE problem to evaluate the decryption function of the base scheme efficiently and with low noise growth. Thus, to follow this blueprint, we propose an NTRU-based GSW-like scheme, which we call NGS. As the GSW-like scheme of [30], NGS can encrypt a polynomial $m \in R := \mathbb{Z}[X]/\langle X^N + 1 \rangle$ in two ciphertext formats:

- *Scalar*: it is a standard NTRU ciphertext encrypting m as $g/f + \Delta m \in R_Q$.
- *Vector*: we encrypt m as $\mathbf{c} = \mathbf{g}/f + \mathfrak{g} \cdot m \in R_Q^\ell$, where $\mathfrak{g}$ is a gadget vector and $\ell \approx \log(Q)$.

As in TFHE [10], we define an external product between these two ciphertexts types, which outputs another scalar ciphertext. Notice that we need only ℓ ring elements per vector ciphertext. Thus, our external product is computed with ℓ products in R_Q, while the ciphertexts of the GSW scheme used in TFHE are composed by $4 \cdot \ell'$ ring elements. Therefore, they need $4 \cdot \ell'$ multiplications per external product. Thus, NGS external product can achieve better running times and memory usage for similar parameters.

Here, we focus on using the NGS scheme as an accumulator to homomorphically evaluate the decrytpion function of a base scheme and compare the performance with TFHE bootstrapping. But it is worth to notice that several other applications that use the GSW scheme could take advantage of the faster

homomorphic operations of NGS. For example, by simply replacing GSW by NGS, one could speed up the transciphering for TFHE [20], or the homomorphic evaluation of maximum and minimum functions from [11], or the tree-based private information retrieval from [29].

Fast bootstrapping with non-overstretched parameters. Given our NGS-based external product, we show that MNTRU ciphertexts can be homomorphically decrypted, or bootstrapped, using the NGS scheme with a similar running time as in TFHE. Hence we are able to construct FHE based solely on the NTRU assumption, with similar performance as TFHE.

Given a ciphertext $\mathbf{c} \in \mathbb{Z}_q^n$ of the base scheme, we use the NGS based external product to multiply it with the vector $\mathbf{f}_0 := \mathsf{col}_0(\mathbf{F}) \in \mathbb{Z}^n$, i.e., the first column of the MNTRU secret key. This generates a scalar ciphertext which is then transformed back to an MNTRU ciphertext.

In TFHE's bootstrapping, the LWE secret $\mathbf{s}$ is binary, since this allows one to compute an encryption of $X^{a_i s_i}$ using the fact that $X^{a_i \cdot s_i} = 1 + (X^{a_i} - 1) \cdot s_i$ when $s_i \in \{0, 1\}$. This operation is called a `CMux` gate. Since NTRU has ternary secret keys, adapting the `CMux` would require two consecutive external products, as it was noticed in [28]. Thus, we propose a *ternary* `CMux` gate, which can be executed with a single external product. We notice that this ternary `CMux` is of independent interest, as it can also be applied to other bootstrapping procedures, e.g., if one instantiates TFHE with ternary secrets. Bootstrapping TFHE with ternary keys was also considered in the paper [23].

We also prove that the final noise accumulated by the bootstrapping is $\tilde{O}(n)$, which allows us to choose q as a very low degree polynomial in n, e.g., $q = \tilde{O}(n)$, thus, below the 'fatigue' point that characterizes the overstretched regime of NTRU. Namely, it was shown [24,13] that the dense sublattice attacks against NTRU start to be more efficient than the classic key-recovering attacks when $q = n^{2.484+o(1)}$.

Faster bootstrapping by combining LWE and NTRU. Comparing the external product of TFHE with ours, we see that we need less multiplications in R_Q, thus, less fast Fourier transforms (FFT), which is the most expensive building block in the entire bootstrapping. Hence, we would expect our bootstrapping to be faster than theirs by a constant factor. However, the total number of external products is n, the dimension of the base scheme, which is defined by the hardness of the MNTRU problem. Thus, we have to choose n larger than in TFHE and we end up with a bootstrapping that requires essentially the same number of FFTs as in TFHE.

To obtain a smaller value of n, we propose to replace our MNTRU-based scheme by an LWE-based and use the NGS scheme to bootstrap it. Thus, the decryption function of an LWE-based scheme is evaluated by the NGS scheme, which returns an NGS scalar ciphertext. We show that it is possible to adapt existing key-switching procedures to transform this NTRU ciphertext back to an LWE ciphertext, thus completing the bootstrapping.

Therewith, we need essentially the same number of external products as in TFHE, but each external product requires less FFTs, thus leading to a smaller total number of FFTs in our bootstrapping. In addition, our scheme requires much less key material.

Practical results and C++ implementation: We implemented our bootstrapping algorithms and compared them with that of TFHE. As a result, the bootstrapping of MNTRU ciphertexts is about 40% slower than TFHE's bootstrapping and it requires 9% more key material. However, when the LWE problem is used to construct the base scheme, our running time is about 28% faster than TFHE. As a concrete example, running on a single core of a 3.1 GHz processor, TFHE takes 66 ms while ours takes 48 ms. Furthermore, our LWE/NGS scheme almost halves the total size of bootstrapping and key-switching keys: from 71 MB in TFHE to 39.3 MB. It is important to notice that one bootstrapping allows us to evaluate any binary gate homomorphically, thus, the homomorphic evaluation of any circuit consists of essentially running one bootstrapping for each gate, therefore, the speedup we obtained in the bootstrapping procedure translates directly as the same speedup for any binary circuit.

Our code is publicly available. More details can be found in Section 7.

2 Preliminaries

2.1 Vectors, polynomials, and norms

We use lower-case bold letters for vectors and upper-case bold letters for matrices. A zero vector is denoted by $\mathbf{0}$. We denote the $i+1$-th column (resp. row) of a matrix $\mathbf{A}$ by $\mathsf{col}_i(\mathbf{A})$ (resp. $\mathsf{row}_i(\mathbf{A})$). The inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted by $\mathbf{a} \cdot \mathbf{b}$. For any vector $\mathbf{u}$, $\|\mathbf{u}\|$ denotes the infinity norm. Let $[B]$ denote a set $\{1, \ldots, B\}$ for an integer B.

Throughout the paper, N is always a power of two and $R := \mathbb{Z}[X]/\langle X^N + 1\rangle$ is the $(2N)$-th cyclotomic ring. Any element f of R can be always seen as the unique polynomial of degree smaller than N belonging to the coset $f + \langle X^N + 1\rangle$. Hence, writing $f = \sum_{i=0}^{N-1} f_i \cdot X^i$ is unambiguous and we can then define the coefficient vector of f as $\boldsymbol{\phi}(f) := (f_0, \ldots, f_{N-1}) \in \mathbb{Z}^N$. Therefore, we can define the infinity norm of f as $\|f\| := \|\boldsymbol{\phi}(f)\|$. We also define the anti-circulant matrix of f as $\boldsymbol{\Phi}(f) \in \mathbb{Z}^{N \times N}$ such that $\mathsf{row}_i(\boldsymbol{\Phi}(f)) = \boldsymbol{\phi}(f \cdot X^i)$ for $0 \leq i \leq N-1$. Notice that $\forall (k, f, g) \in \mathbb{Z} \times R \times R$, $\boldsymbol{\phi}(k \cdot f \cdot g) = k \cdot \boldsymbol{\phi}(f) \cdot \boldsymbol{\Phi}(g)$. For any $Q \in \mathbb{Z}$, let $R_Q := R/QR = \mathbb{Z}_Q[X]/\langle X^N + 1\rangle$.

Finally, we define $\mathbb{M} := \{\pm b \cdot X^k : b \in \{0, 1\} \text{ and } k \in \mathbb{N}\}$, which will be used as the plaintext space of the vector ciphertexts defined in Section 4.

2.2 Distributions

Discrete Gaussian distribution. We first describe the discrete Gaussian distribution where our secret elements are sampled from. Typically, a discrete Gaussian distribution is defined as a distribution over $\mathbb{Z}$, where every element in $\mathbb{Z}$

is sampled with probability proportional to its probability mass function value under a Gaussian distribution over $\mathbb{R}$. We first define the Gaussian function as $\rho_{\sigma,c}(x) = \exp(-\frac{|x-c|^2}{2\cdot\sigma^2})$ for $\sigma, c \in \mathbb{R} > 0$. Hence, $\rho_{\sigma,c}(\mathbb{Z}) = \sum_{i=-\infty}^{\infty} \rho_{\sigma,c}(i)$. The discrete Gaussian distribution with standard deviation σ and mean c is a distribution on $\mathbb{Z}$ with the probability of $x \in \mathbb{Z}$ given by $\rho_{\sigma,c}(x)/\rho_{\sigma,c}(\mathbb{Z})$. If $c = 0$, we denote this distribution by χ_σ.

Subgaussian distribution. For the analysis of encryption parameters, we need subgaussian random variables over $\mathbb{R}$.

Definition 1. *A random variable V over $\mathbb{R}$ is α-subgaussian if its moment generating function satisfies*

$$\mathbb{E}[\exp(t \cdot V)] \leq \frac{1}{2}\exp(\alpha^2 \cdot t^2)$$

for all $t \in \mathbb{R}$.

From the definition, we can prove that the variance of V, denoted by $\mathsf{Var}(V)$ is bounded by α^2, i.e. $\mathsf{Var}(V) \leq \alpha^2$. Informally, the tails of V are dominated by a Gaussian function with standard deviation α. The following lemma is adapted from [17] to our definition.

Lemma 1. *If $\mathbf{x}$ is a discrete random vector over $\mathbb{R}^n$ such that each component x_i of $\mathbf{x}$ is α_i-subgaussian, then the vector $\mathbf{x}$ is a β-subgaussian vector where $\beta = \max_{i\in[n]} \alpha_i$.*

Subgaussian random variables have an important property called Pythagorean additivity. Given two random variables, α-subgaussian X and β-subgaussian Y, and $a, b \in \mathbb{Z}$, the random variable $a \cdot X + b \cdot Y$ is $\sqrt{a^2 \cdot \alpha^2 + b^2 \cdot \beta^2}$-subgaussian. It implies that

$$\mathsf{Var}(a \cdot X) + \mathsf{Var}(b \cdot Y) \leq a^2 \cdot \mathsf{Var}(X) + b^2 \cdot \mathsf{Var}(Y) \leq a^2 \cdot \alpha^2 + b^2 \cdot \beta^2.$$

For $a \in R$ (resp. $\mathbf{x} \in \mathbb{Z}^n$), we denote by $\mathsf{Var}(a)$ (resp. $\mathsf{Var}(\mathbf{x})$) the maximum variance of each coefficient (resp. component) of a (resp. $\mathbf{x}$). The variance of the product of two polynomials $a, b \in R$ is $\mathsf{Var}(a \cdot b) = n \cdot \mathsf{Var}(a) \cdot \mathsf{Var}(b)$. Similarly, we denote by $\mathsf{Var}(\mathbf{X})$ the maximum variance of each column of a matrix $\mathbf{X}$.

2.3 Decompositions

For fixed integers q and B, we set $\ell := \lceil \log_B q \rceil$ and define $\mathbf{g}_{q,B} := (B^0, \ldots, B^{\ell-1})$. When q and B are clear from the context, we write $\mathbf{g}$. Then, for any $k \in \mathbb{Z}_q$, we represent k by an integer in $[-q/2, q/2)$ and define its signed decomposition in base B as $\mathbf{g}^{-1}(k) = (k_0, \ldots, k_{\ell-1})$ for each integer $|k_i| \leq B/2$ for $i \in [\ell]$. It is easy to see that $\mathbf{g}^{-1}(k) \cdot \mathbf{g} = k$. For any $f \in R_Q$, we define $\mathbf{g}^{-1}(f) := \sum_{i=0}^{N-1} \mathbf{g}^{-1}(f_i)X^i$. It is clear that

$$\mathbf{g}^{-1}(f) \cdot \mathbf{g} = \sum_{i=0}^{N-1} \mathbf{g}^{-1}(f_i) \cdot \mathbf{g} \cdot X^i = \sum_{i=0}^{N-1} f_i \cdot X^i = f.$$

The digit decomposition $\mathbf{g}^{-1}$ can be deterministic or randomized [17,21].

2.4 NTRU problems

It is usual to instantiate the NTRU problem with ternary secrets. In our constructions, we generate the secrets from a distribution on $\{-1, 0, 1\}$ such that zero occurs with probability $1/2$, and 1 and -1 occur with probability $1/4$. This approximates a discrete Gaussian with standard deviation $\sigma = 1/\sqrt{2}$.

Following [13], we can define the *anti-circulant* and the *matrix* versions of the NTRU problem. Each version has a computational and a decisional variant.

Definition 2 (NTRU). *Let $N > 0, Q > 1$ be integers and $R := \mathbb{Z}[X]/\langle X^N + 1\rangle$. Let $\sigma > 0$ be a real number, $g, f \leftarrow \chi_\sigma^N$ and f be invertible in R_Q.*

The (computational) (N, Q, σ)-NTRU problem is to recover f and g given $h := g \cdot f^{-1} \bmod Q$. The (N, Q, σ)-decisional-NTRU problem is to distinguish between h and a uniformly random polynomial sampled from R_Q.

Definition 3 (Matrix NTRU). *Let $n > 0, q > 1$ be integers and $\sigma > 0$ is a real number. Let $\mathbf{G}, \mathbf{F} \leftarrow \chi_\sigma^{n\times n}$ and $\mathbf{F}$ be invertible modulo q.*

The (computational) (n, q, σ)-matrix-NTRU problem is to recover $\mathbf{F}$ and $\mathbf{G}$ given $\mathbf{H} := \mathbf{G} \cdot \mathbf{F}^{-1} \bmod q$. The (n, q, σ)-decisional-matrix-NTRU problem is to distinguish between $\mathbf{H}$ and a uniformly random matrix from $\mathbb{Z}_q^{n\times n}$.

3 Matrix-NTRU base encryption scheme

Our base encryption scheme is based on the matrix NTRU (MNTRU) problem. It encrypts a bit $m \in \{0, 1\}$ as if it were an element of $\mathbb{Z}_4$; i.e. we multiply m by $\Delta := \lfloor q/4 \rceil$.

As such we can evaluate a NAND gate by adding two ciphertexts encrypting a bit and considering the result modulo 4. The result is $m = 2$ if $\mathsf{NAND}(m_0, m_1) = 0$ and $m \in \{0, 1\}$ if $\mathsf{NAND}(m_0, m_1) = 1$. We can transform this ciphertext with the result modulo 4 back to an encryption of $\mathsf{NAND}(m_0, m_1)$ with a simple affine transformation, as shown below. This ensures that after one homomorphic NAND gate, we obtain a message defined in $\mathbb{Z}_2$, i.e. multiplied by $\lfloor q/2 \rceil$. Since the message is only one bit, we can define its ciphertext as a vector in $\mathbb{Z}_q^n$ as shown in Introduction.

A standard MNTRU ciphertext would have the form $\mathbf{g} \cdot \mathbf{F}^{-1} + \Delta \cdot \mathbf{m}$, i.e., with only the noise term being divided by secret key $\mathbf{F}$, however, this would introduce a new noise term in the decrypting, when we multiply the ciphertext by $\mathsf{col}_0(\mathbf{F})$. In more detail, the key switching procedure presented in Section 4.5, which transforms a ciphertext from NGS to MNTRU, would output a ciphertext of the form $\mathbf{c}' = \mathbf{g} \cdot \mathbf{F}^{-1} + \mathbf{e} + \Delta \cdot \mathbf{m}$, where $\|\mathbf{e}\| = \Omega(n)$. Then, the decryption would produce $\mathbf{e} \cdot \mathsf{col}_0(\mathbf{F})$, whose norm would be $\Omega(n^2)$. Thus, to avoid such large noise, we define a MNTRU ciphertext with the form $(\mathbf{g} + \Delta \cdot \mathbf{m}) \cdot \mathbf{F}^{-1}$.

Hence, the MNTRU scheme is defined by the following four algorithms. Note that the decryption procedure below is valid for the ciphertexts produced by a NAND gate.

- $\mathsf{MNTRU.ParamGen}(1^\lambda)$: Receives the security parameter and outputs (n, q, σ).

- $\mathsf{MNTRU.KeyGen}$: Sample $\mathbf{F} \leftarrow \chi_\sigma^{n\times n}$ until $\mathbf{F}^{-1}$ exists in $\mathbb{Z}_q^{n\times n}$. Define $\mathsf{sk} := \mathbf{F}$. Create a public evaluation key as $\mathsf{evk} := (\mathbf{g} + \lfloor 5\cdot q/8 \rceil \cdot (1,\mathbf{0}))\cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$, where $\mathbf{g} \leftarrow \chi_\sigma^n$. Output $(\mathsf{evk}, \mathsf{sk})$.
- $\mathsf{MNTRU.Enc}(m, \mathsf{sk})$: Given $m \in \{0,1\}$, sample $\mathbf{g} \leftarrow \chi_\sigma^n$. Let $\Delta := \lfloor q/4 \rceil$ and output
$$c = (\mathbf{g} + \Delta \cdot (m, \mathbf{0}))\cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n.$$
We call it a fresh MNTRU ciphertext.
- $\mathsf{MNTRU.Dec}(\mathbf{c}, \mathsf{sk})$: Given the secret key $\mathsf{sk} = \mathbf{F}$ and a ciphertext $\mathbf{c} \in \mathbb{Z}_q^n$, which is of the form $(\mathbf{g} + \lfloor q/2 \rceil \cdot (m, \mathbf{0}))\cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$, this algorithm computes $r = \mathbf{c} \cdot \mathsf{col}_0(\mathbf{F}) \bmod q$ and outputs
$$\left\lfloor \frac{2\cdot r}{q} \right\rceil \bmod 2.$$
- $\mathsf{MNTRU.Nand}(\mathbf{c}_0, \mathbf{c}_1, \mathsf{evk})$: Given the evaluation key evk and two ciphertexts of the form $(\mathbf{g}_i + \lfloor q/4 \rceil \cdot (m_i, \mathbf{0}))\cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$, where $m_i \in \{0,1\}$ output
$$\mathbf{c}_{\mathsf{NAND}} := \mathsf{evk} - \mathbf{c}_0 - \mathbf{c}_1.$$

This homomorphic NAND gate is basically the same as the one presented in [12]. Thus, its output is $\mathbf{c}_{\mathsf{NAND}} = \left(\mathbf{g} - \mathbf{g}_0 - \mathbf{g}_1 + (e \pm q/8)\cdot(1,\mathbf{0}) + \frac{q}{2}\cdot(m,\mathbf{0})\right)\cdot \mathbf{F}^{-1}$ where $|e| \le \frac{3}{2}$ and $m = \mathsf{NAND}(m_0, m_1) = 1 - m_0 \cdot m_1$. One can see this through the following computation.

Let $\mathbf{f} := \mathsf{col}_0(\mathbf{F})$, g_0 be the first element of $\mathbf{g}$, $g_{0,0}$ be the first element of $\mathbf{g}_0$ and $g_{1,0}$ be the first element of $\mathbf{g}_1$ then

$$\begin{aligned}
\mathbf{c}_{\mathsf{NAND}}\cdot\mathbf{f} - (1 - m_0\cdot m_1)\frac{q}{2} &= (\mathsf{evk} - \mathbf{c}_0 - \mathbf{c}_1)\cdot\mathbf{f} - (1 - m_0\cdot m_1)\frac{q}{2} \\
&= g_0 - g_{0,0} - g_{1,0} + \left\lfloor \frac{5q}{8} \right\rceil - \left\lfloor \frac{q}{4} \right\rceil m_0 - \left\lfloor \frac{q}{4} \right\rceil m_1 \\
&\quad - \frac{q}{2} + \frac{q}{2} m_0 \cdot m_1 \\
&= g_0 - g_{0,0} - g_{1,0} + \frac{q}{8} + \epsilon - \frac{q}{4}(m_0 + m_1 - 2m_0\cdot m_1) \\
&\quad + \epsilon_0 \cdot m_0 + \epsilon_1 \cdot m_1 \\
&= g_0 - g_{0,0} - g_{1,0} + \frac{q}{8} + \epsilon - \frac{q}{4}(m_0 - m_1)^2 \\
&\quad + \epsilon_0 \cdot m_0 + \epsilon_1 \cdot m_1,
\end{aligned}$$

where $\epsilon, \epsilon_0, \epsilon_1$ are round-off errors whose absolute value is less or equal to $1/2$. If we set $e = \epsilon + \epsilon_0 \cdot m_0 + \epsilon_1 \cdot m_1$, we have $|e| \le \frac{3}{2}$.

We now show that decrypting the output of a NAND gate gives the correct answer, as long as the sum of three input noises $\mathbf{g} - \mathbf{g}_0 - \mathbf{g}_1$ is not too large. For simplicity, we consider the ternary noise for the following lemma since we instantiate our scheme with ternary secrets as we mentioned above. Therefore, the noise of evaluation key always satisfies that $\|\mathbf{g}\| = 1$. The noise contained in a fresh ciphertext or an evaluation key is called *fresh*.

Lemma 2 (Correctness of decryption). *For* $0 \leq i \leq 1$, *let* $\mathbf{c}_i := (\mathbf{g}_i + \lfloor q/4 \rceil \cdot (m_i, \mathbf{0})) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$ *be an encryption of* $m_i \in \{0,1\}$. *Consider that* evk *is generated with a ternary* $\mathbf{g}$ *and let* $\mathbf{c} := \mathsf{MNTRU.Nand}(\mathbf{c}_0, \mathbf{c}_1, \mathsf{evk})$. *If* $\|\mathbf{g}_0 + \mathbf{g}_1\| < (q-20)/8$, *then* $\mathsf{MNTRU.Dec}(\mathbf{c}, \mathsf{sk})$ *outputs* $\mathsf{NAND}(m_0, m_1)$.

Proof. From the above analysis, we know that

$$\mathbf{c} = \Big(\mathbf{g} - \mathbf{g}_0 - \mathbf{g}_1 + e(1,\mathbf{0}) \pm q/8 \cdot (1,\mathbf{0}) + (q/2) \cdot (m,\mathbf{0})\Big) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$$

where $m := \mathsf{NAND}(m_0, m_1)$.

Let $\mathbf{f} := \mathsf{col}_0(\mathbf{F})$. To decrypt $\mathbf{c}$, we compute $r := \mathbf{c} \cdot \mathbf{f} \bmod q$. Notice that for some $u \in \mathbb{Z}$, we have

$$r = g - g_0 - g_1 + e \pm q/8 + (q/2) \cdot m - u \cdot q,$$

where g, g_0 and g_1 are the first components of $\mathbf{g}, \mathbf{g}_0$ and $\mathbf{g}_1$, respectively. Thus, the second step of the decryption operation gives us

$$\left\lfloor \frac{2 \cdot r}{q} \right\rceil = \left\lfloor \frac{2 \cdot (g - g_0 - g_1)}{q} + \frac{2 \cdot e}{q} \pm \frac{1}{4} \right\rceil + m - 2 \cdot u$$

which is equal to m modulo 2 as long as $|2 \cdot (g - g_0 - g_1)/q + 3/q \pm 1/4| < 1/2$. Thus, the inequality simply implies that

$$\|\mathbf{g} - \mathbf{g}_0 - \mathbf{g}_1\| < \left(\frac{1}{2} - \frac{1}{4} - \frac{3}{q}\right) \cdot \frac{q}{2} = \frac{q-12}{8}.$$

Since the noise of evaluation key is always fresh and sampled from ternary elements, $\|\mathbf{g}\| = 1$. It implies that if $\|\mathbf{g}_0 + \mathbf{g}_1\| < (q-12)/8 - 1 = (q-20)/8$, then the result holds.

□

4 NGS: NTRU-based GSW-like scheme

In this section, we present a (ring-based) NTRU-based scheme that has two encryption functions. The first one encrypts a plaintext m which is a ternary polynomial as an element of R_Q, whilst the second one encrypts it as a vector over R_Q using "gadget vectors". To simplify the noise analysis, we assume that all the messages encrypted by the vector ciphertexts belong to the following set of monomials: $\mathbb{M} = \{\pm b \cdot X^k : b \in \{0,1\} \text{ and } k \in \mathbb{N}\}$. We notice that this assumption holds for our bootstrapping procedures.

Our scheme has quasi-additive noise-growth as the GSW scheme [19]. In fact, it is inspired by the simplified variant of GSW proposed in [12]. We call this scheme NGS, which stands for NTRU-GSW-like encryption Scheme. In Section 5, the NGS scheme is used as the accumulator to homomorphically evaluate the decryption of another, much simpler scheme based on the matrix NTRU problem. Following the idea of [10] to speed up the bootstrapping, we define an

external product that multiplies scalar NTRU ciphertexts, i.e. elements of R_Q, and vector NTRU ciphertexts, i.e. vectors over R_Q. This is the framework used to obtain a fast bootstrapping in FHEW [12] and TFHE [10].

Usually NTRU schemes are defined as asymmetric ciphers by publishing a public key $h := g/f \bmod Q$. Since such public keys are not involved in bootstrapping, we present a symmetric version of this scheme. Notice that any encryption of zero could be used as a public key. Moreover, since the NGS ciphertexts are never decrypted in the bootstrapping pipeline, we omit the decryption procedure.

4.1 Basic procedures

The NTRU-based encryption scheme is defined as follows.

- $\mathsf{NGS.ParamGen}(1^\lambda)$: Receives the security parameter and outputs the tuple $(N, Q, \varsigma, B, \ell)$, where B is a base used to decompose the ciphertexts and $\ell := \lceil \log_B(Q) \rceil$.
- $\mathsf{NGS.KeyGen}$: Sample $f' \leftarrow \chi_\varsigma^N$ and set $f := 1 + 4 \cdot f'$ until f^{-1} exists in R_Q. Output $\mathsf{sk} := f$.
- $\mathsf{NGS.EncS}(\mathsf{sk}, m)$: Given a ternary polynomial m , sample $g \leftarrow \chi_\varsigma^N$, define $\Delta := \lfloor Q/4 \rceil$, and output $c = g/f + \Delta \cdot m \in R_Q$. We call c a *scalar encryption* of m.
- $\mathsf{NGS.EncVec}(\mathsf{sk}, m)$: Given $m \in \mathbb{M}$, sample $g_i \leftarrow \chi_\varsigma^N$ for $0 \leq i \leq \ell - 1$. Define $\mathbf{g} := (g_0, \ldots, g_{\ell-1})$ and $\mathfrak{g} = (B^0, B^1, \ldots, B^{\ell-1})$. Output $\mathbf{c} = \mathbf{g}/f + \mathfrak{g} \cdot m \in R_Q^\ell$. We call $\mathbf{c}$ a *vector encryption* of m.

4.2 External product

Having defined two types of encryptions, scalar and vector ciphertexts, we can define the "external product" between them as proposed in TFHE [10]. The external product is cheaper than the NGS homomorphic multiplication (i.e. the convolution of two vector ciphertexts).

Suppose we have a scalar encryption $c := g/f + \Delta \cdot u \in R_Q$ of a ternary polynomial u and a vector encryption $\mathbf{c} := \mathbf{g}/f + \mathfrak{g} \cdot v \in R_Q^\ell$ of a message $v \in \mathbb{M}$. Then, the external product of c and $\mathbf{c}$ is defined as follows

$$c \boxdot \mathbf{c} := \mathfrak{g}^{-1}(c) \cdot \mathbf{c} \in R_Q.$$

Since $\mathfrak{g}^{-1}(c) \cdot \mathfrak{g} = c$, it is clear that $c_{mult} = c \boxdot \mathbf{c}$ is equal to

$$c_{mult} := (\mathfrak{g}^{-1}(c) \cdot \mathbf{g})/f + (\mathfrak{g}^{-1}(c) \cdot \mathfrak{g} \cdot v) = \underbrace{(\mathfrak{g}^{-1}(c) \cdot \mathbf{g} + g \cdot v)}_{g_{mult}}/f + \Delta \cdot u \cdot v.$$

Hence, c_{mult} is a valid scalar encryption of the product $u \cdot v$ as long as the noise term g_{mult} is small enough. We formalize this notion in the next section. Notice that it is important that $\|u \cdot v\| < 4$, otherwise, multiplying it by Δ introduces a round-off error and produces an ill-formed ciphertext. Since we are assuming that $v \in \mathbb{M}$, we have $\|u \cdot v\| \leq \|u\| < 2$.

4.3 Noise analysis

Instead of performing a worst-case analysis of the noise growth, which boils down to bounding every element by its infinity norm, we provide a more realistic average-case noise analysis. To do so, we can instantiate $\mathbf{g}^{-1}$ with a randomized gadget decomposition algorithm [17,21], or we can use a deterministic decomposition and heuristically assume that all the coefficients of the errors of MNTRU and NGS samples are independent and concentrated; thus, they are subgaussian random variables. The first approach is used in FHEW [12], while the latter is present in TFHE [10,8]. Both methods return a subgaussian random variable. Therefore, our analysis assumes that for all $a \in R_Q$, $\mathbf{g}^{-1}_{q,B_{\mathrm{ksk}}}(a)$ is a γ-subgaussian for some $\gamma = O(B)$.

Definition 4 (Noise of a scalar ciphertext). *Let $c = g/f + \Delta \cdot m \in R_Q$. We define the noise of c as $\mathsf{err}(c) := c \cdot f - \Delta \cdot m \in R_Q$ and interpret it as a polynomial over $\mathbb{Z}[X]$ with coefficients in $[-Q/2, Q/2]$.*

We also define the noise of a vector ciphertext below for our noise analysis.

Definition 5 (Noise of a vector ciphertext). *Let $\mathbf{c} = \mathbf{g}/f + \mathfrak{g} \cdot m \in R_Q^\ell$. We define the noise of $\mathbf{c}$ as $\mathsf{err}(\mathbf{c}) := \mathbf{c} \cdot f - \mathfrak{g} \cdot m \cdot f \in R_Q$ and interpret it as a vector of polynomials over $\mathbb{Z}[X]$ with coefficients in $[-Q/2, Q/2]$.*

We first bound the noise of ciphertexts of a special form, namely fresh ones that encrypt monomials. This includes the important special case of $m \in \{0, 1\}$.

Lemma 3 (Bound on the noise of a (fresh) scalar ciphertext). *Let $c = g/f + \Delta \cdot m \in R_Q$ be a ciphertext of m. If m is a monomial of the form $\pm b \cdot X^k$ for some $b \in \{0, 1\}$, then*

$$\mathsf{Var}(\mathsf{err}(c)) \leq \mathsf{Var}(g) + 4 \cdot \varsigma^2.$$

If m is a ternary polynomial with degree at most $N - 1$, then

$$\mathsf{Var}(\mathsf{err}(c)) \leq \mathsf{Var}(g) + 4 \cdot N \cdot \varsigma^2.$$

Moreover, if c is a fresh ciphertext, then $\mathsf{Var}(\mathsf{err}(c)) \leq 5 \cdot \varsigma^2$ for a monomial m and the variance is bounded by $(4 \cdot N + 1) \cdot \varsigma^2$ for a ternary polynomial m.

Proof. Let $\Delta = Q/4 + \epsilon$ for some $\epsilon \in \mathbb{R}$ such that $|\epsilon| \leq 1/2$. Since in R_Q it holds that

$$c \cdot f = g + (1 + 4 \cdot f') \cdot (Q/4 + \epsilon) \cdot m = g + 4 \cdot f' \cdot \epsilon \cdot m + \Delta \cdot m,$$

we have $\mathsf{err}(c) := c \cdot f - \Delta \cdot m = g + 4 \cdot f' \cdot \epsilon \cdot m$. Notice that, if $m \in \{0, \pm 1, \pm X, \ldots, \pm X^{N-1}\}$, we have $\mathsf{Var}(f' \cdot m) \leq \mathsf{Var}(f')$, thus

$$\mathsf{Var}(\mathsf{err}(c)) \leq \mathsf{Var}(g) + (4 \cdot \epsilon)^2 \cdot \mathsf{Var}(f') \leq \mathsf{Var}(g) + 4 \cdot \varsigma^2.$$

If m is a ternary polynomial of degree at most $N - 1$,

$$\mathsf{Var}(\mathsf{err}(c)) \leq \mathsf{Var}(g) + (4 \cdot \epsilon)^2 \cdot \|m\|_2^2 \cdot \mathsf{Var}(f') \leq \mathsf{Var}(g) + 4 \cdot N \cdot \varsigma^2.$$

If c is a fresh ciphertext, then $\mathsf{Var}(g) = \varsigma^2$ and the rest of the lemma follows. □

We now analyze how the external product increases the noise.

Lemma 4 (Noise growth of external product). *Let $c := g/f + \Delta \cdot u \in R_Q$ and $\mathbf{c} := \mathbf{g}/f + \mathfrak{g} \cdot v \in R_Q^\ell$. Define $c_{mult} := c \boxdot \mathbf{c}$ as above. Then*

$$\mathsf{Var}(\mathsf{err}(c_{mult})) \leq N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathbf{g}) + \|v\|_2^2 \cdot \mathsf{Var}(g) + 4 \cdot \varsigma^2.$$

If $v \in \mathbb{M} := \{\pm b \cdot X^k : b \in \{0,1\}$ and $k \in \mathbb{N}\}$, then

$$\mathsf{Var}(\mathsf{err}(c_{mult})) \leq N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c})) + \mathsf{Var}(\mathsf{err}(c))$$

Proof. From the analysis in Section 4.2, we know that $c_{mult} = g_{mult}/f + \Delta \cdot m$, where $g_{mult} := \mathfrak{g}^{-1}(c) \cdot \mathbf{g} + g \cdot v$ and $m := v \cdot u$. Thus, by Lemma 3, we have $\mathsf{Var}(\mathsf{err}(c_{mult})) \leq \mathsf{Var}(g_{mult}) + 4 \cdot \varsigma^2$. Since

$$\mathsf{Var}(g_{mult}) \leq \mathsf{Var}(\langle \mathfrak{g}^{-1}(c), \mathbf{g} \rangle) + \mathsf{Var}(g \cdot v) \leq N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathbf{g}) + \|v\|_2^2 \cdot \mathsf{Var}(g),$$

the result follows. If $v \in \mathbb{M}$, then $\|v\|_2^2 \leq 1$, and the value $\mathsf{Var}(g_{mult}) + 4 \cdot \varsigma^2$ is bounded by $N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathbf{g}) + \mathsf{Var}(g) + 4 \cdot \varsigma^2$, which is $N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c})) + \mathsf{Var}(\mathsf{err}(c))$ by Definition 5 and Lemma 3. □

Our goal now is to analyze the noise growth caused by a sequence of k such external products, i.e., $c' = c \boxdot_{i=1}^k \mathbf{c}_i = (\dots((c \boxdot \mathbf{c}_1) \boxdot \mathbf{c}_2) \dots \boxdot \mathbf{c}_k)$. Since in our bootstrapping the messages encrypted by vector ciphertexts are of the form $\pm b \cdot X^m$ for some bit b, we simplify the analysis by supposing that the messages encrypted by $\mathbf{c}_1, \dots, \mathbf{c}_k$ belong to $\mathbb{M}$. This allows us to ignore the term $\|v\|_2^2$ in Lemma 4 as it is bounded by 1.

Lemma 5 (Noise of a sequence of external products). *For $1 \leq i \leq k$, let $\mathbf{c}_i := \mathbf{g}_i/f + \mathfrak{g} \cdot m_i \in R_Q^\ell$ with $m_i \in \mathbb{M}$. Let $c_0 = g_0/f + \Delta \cdot m_0 \in R_Q$ with a ternary polynomial m_0. If $c' := c \boxdot_{i=1}^k \mathbf{c}_i$, then*

$$\mathsf{Var}(\mathsf{err}(c')) \leq N \cdot \ell \cdot \gamma^2 \cdot \sum_{i=1}^{k} \mathsf{Var}(\mathbf{g}_i) + \mathsf{Var}(g_0) + 4 \cdot \varsigma^2.$$

Proof. Let $c_i := c_{i-1} \boxdot \mathbf{c}_i = g_i/f + \Delta \cdot m_i'$ for $1 \leq i \leq k$. It is clear that $c' = c_k$. Using the fact that $v_1, \dots, v_k \in \mathbb{M}$, we apply Lemma 4 k times and obtain

$$\begin{aligned}
\mathsf{Var}(\mathsf{err}(c_k)) &\leq N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_k)) + \mathsf{Var}(\mathsf{err}(c_{k-1})) \\
&\leq N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_k)) + N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_{k-1})) + \mathsf{Var}(\mathsf{err}(c_{k-2})) \\
&\quad \vdots \\
&\leq N \cdot \ell \cdot \gamma^2 \cdot \sum_{i=1}^{k} \mathsf{Var}(\mathsf{err}(\mathbf{c}_i)) + \mathsf{Var}(\mathsf{err}(c_0)) \\
&= N \cdot \ell \cdot \gamma^2 \cdot \sum_{i=1}^{k} \mathsf{Var}(\mathbf{g}_i) + \mathsf{Var}(g_0) + 4 \cdot \varsigma^2.
\end{aligned}$$

□

Corollary 1. *Using the notation of Lemma 5, if all the ciphertexts are fresh, then*

$$\mathsf{Var}(\mathsf{err}(c')) \leq (4 + (k+1) \cdot N \cdot \ell \cdot \gamma^2) \cdot \varsigma^2.$$

4.4 Modulus-switching

In this section, we show that the modulus-switching technique for (R)LWE-based schemes can be adapted to NTRU-based schemes. Given a ciphertext $c = g/f + \Delta \cdot \mu \in R_Q$ for some message μ which is a ternary polynomial, we can multiply c by q/Q and round it to obtain a ciphertext defined modulo q. Since $\lfloor y \rceil = y + \epsilon$, the modulus switching essentially scales the ciphertext and adds a small rounding error, which is then multiplied by the secret key f during decryption. As in the analysis of [12], we define the following randomized rounding function.

Definition 6. *Let $Q, q \in \mathbb{Z}$ and $1 < q < Q$. The randomized rounding function $[\cdot]_{Q:q} : \mathbb{Z}_Q \to \mathbb{Z}_q$ is defined as $[z]_{Q:q} := \lfloor q \cdot z/Q \rfloor + B$ where $B \in \{0, 1\}$ is a Bernoulli random variable with $\Pr[B = 1] = (q \cdot z/Q) - \lfloor q \cdot z/Q \rfloor \in [0, 1]$.*

Notice that the the rounding error $\epsilon := [z]_{Q:q} - (q \cdot z/Q)$ is 1-subgaussian. We extend the definition to polynomials, vectors, and matrices by applying the rounding entry-wise. Thus, the modulus switching is defined as

$$\mathsf{ModSwitch}(c) = \sum_{i=0}^{N-1} [c_i]_{Q:q} \cdot X^i \in R_q.$$

Lemma 6. *Let $c = g/f + \lfloor Q/4 \rceil \cdot \mu \in R_Q$. Then, $\mathsf{ModSwitch}(c)$ is a scalar encryption of μ in R_q. Moreover,*

$$\mathsf{Var}(\mathsf{err}(\mathsf{ModSwitch}(c))) \leq (q/Q)^2 \cdot \mathsf{Var}(\mathsf{err}(c)) + 1 + 16 \cdot N \cdot \varsigma^2.$$

Proof. Just notice that $\mathsf{ModSwitch}(c) = (q \cdot g/Q)/f + \epsilon + \Delta \cdot \mu \in R_q$, where $\Delta = \lfloor q/4 \rfloor$ and ϵ is a polynomial with infinite norm bounded by 1, therefore, $\mathsf{err}(\mathsf{ModSwitch}(c)) = q \cdot \mathsf{err}(c)/Q + \epsilon \cdot f = q \cdot \mathsf{err}(c)/Q + \epsilon \cdot (1 + 4f')$. Then the variance of the noise is as follows:

$$\begin{aligned}
\mathsf{Var}(\mathsf{err}(\mathsf{ModSwitch}(c))) &= \mathsf{Var}(q \cdot \mathsf{err}(c)/Q + \epsilon + 4 \cdot \epsilon \cdot f') \\
&= \mathsf{Var}(q \cdot \mathsf{err}(c)/Q) + \mathsf{Var}(\epsilon) + 16 \cdot \mathsf{Var}(\epsilon \cdot f') \\
&\leq (q/Q)^2 \cdot \mathsf{Var}(\mathsf{err}(c)) + \mathsf{Var}(\epsilon) + 16 \cdot N \cdot \mathsf{Var}(\epsilon) \cdot \mathsf{Var}(f') \\
&\leq (q/Q)^2 \cdot \mathsf{Var}(\mathsf{err}(c)) + 1 + 16 \cdot N \cdot \mathsf{Var}(f').
\end{aligned}$$

The last inequality holds since ϵ is 1-subgaussian.

4.5 Key-switching from NGS to the base scheme

As we will see in Section 5, our bootstrapping procedure starts with a ciphertext $(\mathbf{g} + \Delta \cdot (m, \mathbf{0})) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$ of the base scheme. After modulus-switching, it produces an NTRU encryption $c = g/f + \epsilon + \Delta \cdot \mu \in R_q$, where μ is a polynomial whose constant term is equal to $m \in \{0, 1\}$ and $\Delta := \lfloor q/4 \rceil$. To finish the bootstrapping, we want to obtain again a base scheme ciphertext of the form $\mathbf{c}' = (\mathbf{g}' + \Delta \cdot (m, \mathbf{0})) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$. To achieve this, we define the following key-switching operation.

- Key-switching key generation: The input of this procedure is composed by the secret keys $f \in R$ and $\mathbf{F} \in \mathbb{Z}^{n \times n}$, and the parameters $\sigma_{\mathtt{ksk}}$, q, and $B_{\mathtt{ksk}}$. Let $L = \lceil \log_{B_{\mathtt{ksk}}}(q) \rceil$. Define $\mathbf{P} \in \mathbb{Z}^{(N \cdot L) \times N}$ as the gadget matrix $\mathbf{I}_N \otimes \mathfrak{g}_{q, B_{\mathtt{ksk}}}$, i.e. each "diagonal element" of $\mathbf{P}$ is equal to $\mathfrak{g}_{q, B_{\mathtt{ksk}}} \in \mathbb{Z}^L$. Also, let $\mathbf{E} \in \mathbb{Z}^{N \times n}$ be the matrix whose entries are zeros except for $\mathbf{E}_{0,0} = 1$.
  Then, sample $\mathbf{G} \leftarrow \chi_{\sigma_{\mathtt{ksk}}}^{(N \cdot L) \times n}$ and output
  $$\mathtt{ksk} := (\mathbf{G} + \mathbf{P} \cdot \boldsymbol{\Phi}(f) \cdot \mathbf{E}) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^{(N \cdot L) \times n},$$
  where $\boldsymbol{\Phi}(f)$ is the anti-circulant matrix of f.
- Key-Switching algorithm: Given an output of modulus-switching, $c = g/f + \epsilon + \Delta \cdot \mu \in R_q$, and a key-switching key $\mathtt{ksk}$, let
  $$\mathsf{KeySwitch}(c, \mathtt{ksk}) := \mathbf{y} \cdot \mathtt{ksk} \in \mathbb{Z}_q^n$$
  where $\mathbf{y} := (\mathfrak{g}_{q, B_{\mathtt{ksk}}}^{-1}(c_0), \ldots, \mathfrak{g}_{q, B_{\mathtt{ksk}}}^{-1}(c_{N-1})) \in \mathbb{Z}^{N \cdot L}$.

Lemma 7 (Correctness of key-switching). *Let $c = g/f + \epsilon + \Delta \cdot \mu \in R_q$ be a scalar encryption of a ternary polynomial μ, with $\Delta = \lfloor q/4 \rceil$, and $\mathtt{ksk}$ a key-switching key from $f = 1 + 4 \cdot f'$ to $\mathbf{F} \in \mathbb{Z}^{n \times n}$. Then, $\mathsf{KeySwitch}(c, \mathtt{ksk})$ outputs a base scheme ciphertext $\mathbf{c}' = (\mathbf{g} + \Delta \cdot (\mu_0, \mathbf{0})) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^{n \times n}$, where μ_0 is the constant term of μ. Moreover, its time complexity is $O(N \cdot n \cdot \log q)$ operations on $\mathbb{Z}_q$.*

Proof. Let $|\epsilon'| \leq 1/2$ such that $\Delta = q/4 + \epsilon'$. Since
$$\mathbf{y} \cdot \mathbf{P} = \boldsymbol{\phi}(c) = \boldsymbol{\phi}(g) \cdot \boldsymbol{\Phi}(f)^{-1} + \boldsymbol{\phi}(\epsilon) + \Delta \cdot \boldsymbol{\phi}(\mu),$$
it is clear that
$$\mathbf{y} \cdot \mathbf{P} \cdot \boldsymbol{\Phi}(f) = \boldsymbol{\phi}(g) + \boldsymbol{\phi}(\epsilon) \cdot \boldsymbol{\Phi}(f) + \epsilon' \cdot \boldsymbol{\phi}(\mu) \cdot 4 \cdot \boldsymbol{\Phi}(f') + \Delta \cdot \boldsymbol{\phi}(\mu) \in \mathbb{Z}_q^N.$$
Therefore, by defining $\mathbf{g}' := \boldsymbol{\phi}(\epsilon) \cdot \boldsymbol{\Phi}(f) + \epsilon' \cdot \boldsymbol{\phi}(\mu) \cdot 4 \cdot \boldsymbol{\Phi}(f')$, the following equality holds modulo q:
$$\mathbf{c}' = \Big(\mathbf{y} \cdot \mathbf{G} + \Big(\boldsymbol{\phi}(g) + \mathbf{g}' + \Delta \cdot \boldsymbol{\phi}(\mu)\Big) \cdot \mathbf{E}\Big) \cdot \mathbf{F}^{-1}.$$
And because $\mathbf{v} \cdot \mathbf{E} = (v_0, \mathbf{0}) \in \mathbb{Z}^n$ for any $\mathbf{v} \in \mathbb{Z}^N$, we finally obtain
$$\mathbf{c}' = (\mathbf{y} \cdot \mathbf{G} + (g_0, \mathbf{0}) + (g_0', \mathbf{0}) + \Delta \cdot (\mu_0, \mathbf{0})) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n.$$
If we set $\mathbf{g} = \mathbf{y} \cdot \mathbf{G} + (g_0, \mathbf{0}) + (g_0', \mathbf{0})$, the result holds. Moreover, since the procedure consists in multiplying $\mathbf{y} \in \mathbb{Z}^{N \cdot L}$ by each of the n columns of $\mathtt{ksk}$, it is clear that it costs $O(N \cdot n \cdot \log q)$ operations on $\mathbb{Z}_q$. □

Noise analysis on the matrix key switching procedure. We first see that the noise of c which is an output of modulus-switching equals to $g + \epsilon + 4 \cdot \epsilon \cdot f' + 4 \cdot f' \cdot \epsilon' \cdot \mu$ by Definition 4. Then the variance of the noise is following by Lemma 3:

$$\begin{aligned}\mathsf{Var}(\mathsf{err}(c)) &\leq \mathsf{Var}(g) + \mathsf{Var}(\epsilon) + 16 \cdot N \cdot \mathsf{Var}(\epsilon) \cdot \mathsf{Var}(f') + 4 \cdot \|\mu\|_2^2 \cdot \mathsf{Var}(f') \\ &\leq \mathsf{Var}(g) + 1 + 16 \cdot N \cdot \mathsf{Var}(f') + 4 \cdot N \cdot \mathsf{Var}(f') \\ &= \mathsf{Var}(g) + 1 + 20 \cdot N \cdot \varsigma^2\end{aligned}$$

The noise contained in $\mathbf{c}'$ is $\mathbf{y} \cdot \mathbf{G} + (g_0, \mathbf{0}) + (g_0', \mathbf{0})$. In fact, $\mathbf{G}$ is the noise of the key switching key $\mathtt{ksk}$, and $g_0 + g_0'$ is very close to the noise originally contained in c, before key-switching. Notice that

$$\begin{aligned}\mathsf{Var}(g_0') &\leq \mathsf{Var}(\boldsymbol{\phi}(\epsilon) \cdot \boldsymbol{\Phi}(f)) + (4\epsilon')^2 \cdot \mathsf{Var}(\boldsymbol{\phi}(\mu) \cdot \boldsymbol{\Phi}(f')) \\ &\leq \mathsf{Var}(\epsilon) + 16 \cdot N \cdot \mathsf{Var}(\epsilon) \cdot \mathsf{Var}(\boldsymbol{\Phi}(f')) + 4 \cdot \|\boldsymbol{\phi}(\mu)\|_2^2 \cdot \mathsf{Var}(\boldsymbol{\Phi}(f')) \\ &\leq 1 + 16 \cdot N \cdot \mathsf{Var}(\boldsymbol{\Phi}(f')) + 4 \cdot N \cdot \mathsf{Var}(\boldsymbol{\Phi}(f')) \\ &\leq 1 + 20 \cdot N \cdot \varsigma^2.\end{aligned}$$

Thus, assuming the outputs of decomposition $\mathfrak{g}^{-1}(\cdot)$ is γ-subgaussian, the variance of $\mathsf{err}(\mathbf{c}')$ is following:

$$\begin{aligned}\mathsf{Var}(\mathsf{err}(\mathbf{c}')) &= \mathsf{Var}(\mathbf{y} \cdot \mathbf{G}) + \mathsf{Var}(g_0, \mathbf{0})) + \mathsf{Var}((g_0', \mathbf{0})) \\ &\leq N \cdot L \cdot \mathsf{Var}(\mathfrak{g}^{-1}(\phi(c))) \cdot \mathsf{Var}(\mathbf{G}) + \mathsf{Var}(\boldsymbol{\phi}(g)) + 1 + 20 \cdot N \cdot \varsigma^2 \\ &\leq N \cdot L \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathtt{ksk})) + \mathsf{Var}(g) + 1 + 20 \cdot N \cdot \varsigma^2 \\ &= N \cdot L \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathtt{ksk})) + \mathsf{Var}(\mathsf{err}(c)).\end{aligned}$$

5 Bootstrapping

As explained in the introduction, to cope with the ternary secrets inherent in NTRU we utilize a ternary $\mathtt{CMux}$ gate. Our *ternary* $\mathtt{CMux}$ gate is defined as follows: For a given $f_i \in \{-1, 0, 1\}$, we define two keys $\mathtt{bsk}_{i,0}$ and $\mathtt{bsk}_{i,1}$:

$$\begin{cases} f_i = -1 & \implies \mathtt{bsk}_{i,0} := \mathsf{NGS.EncVec}(0) \wedge \mathtt{bsk}_{i,1} := \mathsf{NGS.EncVec}(1) \\ f_i = 0 & \implies \mathtt{bsk}_{i,0} := \mathsf{NGS.EncVec}(0) \wedge \mathtt{bsk}_{i,1} := \mathsf{NGS.EncVec}(0) \\ f_i = 1 & \implies \mathtt{bsk}_{i,0} := \mathsf{NGS.EncVec}(1) \wedge \mathtt{bsk}_{i,1} := \mathsf{NGS.EncVec}(0) \end{cases} \quad (1)$$

Then, our $\mathtt{CMux}$ gate is defined as

$$\mathtt{CMux}_i(c_i) := \mathbf{1} + (X^{c_i} - 1) \cdot \mathtt{bsk}_{i,0} + (X^{-c_i} - 1) \cdot \mathtt{bsk}_{i,1},$$

where $\mathbf{1}$ is a trivial, noiseless, encryption of one, i.e. simply $\mathfrak{g}$. It is easy to see that $\mathtt{CMux}_i(c_i) = \mathsf{NGS.EncVec}(X^{c_i \cdot f_i})$. In particular, the message encrypted by $\mathtt{CMux}_i(c_i)$ belongs to $\mathbb{M}$ as required by our external product from Section 4.2.

Algorithm 1: Bootstrapping key generation.

Input: $\mathbf{F} \in \mathbb{Z}_q^{n \times n}$ – the secret key of the base scheme.
Output: $\mathtt{bsk}$ – the bootstrapping key.

$(f_0, \dots f_{n-1}) \leftarrow \mathsf{col}_0(\mathbf{F})$
for $i \leftarrow 0$ **to** $n-1$ **do**
  Compute $\mathtt{bsk}_{i,0}$ and $\mathtt{bsk}_{i,1}$ accordingly to Equation 1.
Return $\mathtt{bsk} := \{(\mathtt{bsk}_{i,0}, \mathtt{bsk}_{i,1}) : 0 \le i \le n-1\}$.

Algorithm 2: Bootstrapping algorithm.

Input:
$\mathtt{ct} \in \mathbb{Z}_q^n$ – a base scheme ciphertext encrypting $m \in \{0,1\}$
$\{\mathtt{bsk}_{i,j}\}_{0 \le i \le n-1, 0 \le j \le 1}$ – bootstrapping keys, where each $\mathtt{bsk}_{i,j} \in R_{Q,N}^{\ell}$
$\mathtt{ksk}$ – a key-switching key from the NGS secret key $f \in R$ to the base scheme secret key $\mathbf{F} \in \mathbb{Z}^{n \times n}$.
Output: $\mathtt{ct}' \in \mathbb{Z}_q^n$ – a base-scheme ciphertext encrypting the same m.

$(c_0, \dots, c_{n-1}) \leftarrow \left\lfloor \frac{2 \cdot N \cdot \mathtt{ct}}{q} \right\rceil$
$\mathtt{ACC} \leftarrow \lfloor \frac{Q}{8} \rceil \cdot X^{N/2} \cdot \sum_{i=0}^{N-1} X^i$
for $i \leftarrow 0$ **to** $n-1$ **do**
  $\mathbf{c}_{\mathtt{Mux}} \leftarrow \mathtt{CMux}_i(c_i)$
  $\mathtt{ACC} \leftarrow \mathtt{ACC} \boxdot \mathbf{c}_{\mathtt{Mux}}$
$\mathtt{ACC} \leftarrow \mathtt{ACC} + \lfloor \frac{Q}{8} \rceil \cdot \sum_{i=0}^{N-1} X^i$
$\mathtt{ACC} \leftarrow \mathsf{ModSwitch}(\mathtt{ACC})$
$\mathtt{ct}' \leftarrow \mathsf{KeySwitch}(\mathtt{ACC}, \mathtt{ksk})$
Return $\mathtt{ct}'$.

Recall that our base-scheme ciphertext $\mathbf{c} = (\mathbf{g} + \Delta \cdot (m, \mathbf{0})) \cdot \mathbf{F}^{-1} \in \mathbb{Z}_q^n$ can be decrypted by multiplying it by the first column of $\mathbf{F}$, Thus, our bootstrapping keys are generated using Equation 1 for each entry f_i from the first column of $\mathbf{F}$, see Algorithm 1.

By using the $\mathtt{CMux}$ gate n times and multiplying all the resulting ciphertexts, we obtain an encryption of $X^{\mathbf{c} \cdot \mathsf{col}_0(\mathbf{F})} = X^{g + (N/2) \cdot m}$. We can then multiply this by the (plaintext) "test vector" $T(X) := X^{N/2} \cdot \sum_{i=0}^{N-1} X^i \pmod{X^N + 1}$ to produce a scalar encryption of m. Note, we actually put the test vector in the left most position of the product so that each multiplication is an external product instead of a regular "vector-vector" homomorphic multiplication, i.e. we compute $\lfloor Q/8 \rceil \cdot T(X) \cdot \boxdot_{i=0}^{n-1} \mathtt{CMux}_i(c_i)$, which produces $\mathsf{NGS.EncS}(2 \cdot m - 1)$, but with $\Delta = \lfloor Q/8 \rceil$. Then we add $\mathsf{NGS.EncS}(1)$ to obtain $\mathsf{NGS.EncS}(2 \cdot m)$. The factor two is multiplied by $\lfloor Q/8 \rceil$ and so we obtain $\mathsf{NGS.EncS}(m)$ with $\Delta = \lfloor Q/4 \rceil$, as desired. Finally, we use the key-switching procedure defined in Section 4 to transform this NTRU ciphertext into a matrix NTRU ciphertext of the base scheme. Our bootstrapping is shown in detail in Algorithm 2.

5.1 Bootstrapping noise

Firstly, we analyze the noise growth of our $\mathtt{CMux}$ gate. Let $c_{\mathtt{Mux}} := \mathtt{CMux}_i(c_i)$ for any $0 \leq i \leq n-1$. Then, the following holds:

$$\begin{aligned}\mathsf{Var}(\mathsf{err}(c_{\mathtt{Mux}})) &\leq \|X^{c_i} - 1\|_2^2 \cdot \mathsf{Var}(\mathsf{err}(\mathtt{bsk}_{i,0})) + \|X^{-c_i} - 1\|_2^2 \cdot \mathsf{Var}(\mathsf{err}(\mathtt{bsk}_{i,1})) \\ &\leq 4 \cdot \mathsf{Var}(\mathsf{err}(\mathtt{bsk})),\end{aligned}$$

where $\mathtt{bsk}_{i,0}$ and $\mathtt{bsk}_{i,1}$ are the corresponding bootstrapping keys, which are NGS ciphertexts with noise variance $\mathsf{Var}(\mathsf{err}(\mathtt{bsk}))$.

Now we consider the whole bootstrapping algorithm. In the first line, we scale down the input ciphertext to modulus $2 \cdot N$. We denote the resulting vector by $\mathtt{ct}_{2\cdot N}$. Then we have

$$\left|\mathtt{ct} \cdot \mathsf{col}_0(\mathbf{F}) - \frac{q}{2 \cdot N} \cdot \mathtt{ct}_{2\cdot N} \cdot \mathsf{col}_0(\mathbf{F})\right| \leq \frac{q}{4 \cdot N} \cdot |\mathtt{ct} \cdot \mathsf{col}_0(\mathbf{F})|, \tag{2}$$

where $\mathsf{col}_0(\mathbf{F})$ is the first column of the secret key of $\mathtt{ct}$.

From the line 3 to 5 of Algorithm 2, the output $\mathtt{ACC}$ is obtained by utilizing n external products with $\mathbf{c}_{\mathtt{Mux}}$ whose noise variance is $\mathsf{Var}(\mathsf{err}(c_{\mathtt{Mux}}))$. The variance of the final $\mathsf{err}(\mathtt{ACC})$ based on Lemma 5 is the following:

$$\mathsf{Var}(\mathsf{err}(\mathtt{ACC})) \leq n \cdot N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_{\mathtt{Mux}})) + 4 \cdot \|\mathsf{msg}(\mathtt{ACC})\|_2^2 \cdot \varsigma^2,$$

where $\mathsf{msg}(\mathtt{ACC})$ is $X^{N/2} \cdot \sum_{i=0}^{N-1} X^i$. After line 6, the accumulator $\mathtt{ACC}$ contains a message as a ternary polynomial (say $M(X)$) whose constant term is m. The error term will be changed into

$$\begin{aligned}\mathsf{Var}(\mathsf{err}(\mathtt{ACC})) &\leq n \cdot N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_{\mathtt{Mux}})) + 4 \cdot \|M(X)\|_2^2 \cdot \varsigma^2 \\ &\leq n \cdot N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_{\mathtt{Mux}})) + 4 \cdot N \cdot \varsigma^2\end{aligned}$$

After this step, modulus switching is performed, which results in the noise, by Lemma 6, being

$$\mathsf{Var}(\mathsf{err}(\mathtt{ACC})) \leq (q/Q)^2 \cdot n \cdot N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_{\mathtt{Mux}})) + (q/Q)^2 \cdot 4 \cdot N \cdot \varsigma + 1 + 16 \cdot N \cdot \varsigma$$

After the external product with a key switching key $\mathtt{ksk}$ in line 8, the noise in the resulting $\mathtt{ct}'$ has a variance

$$\begin{aligned}\mathsf{Var}(\mathsf{err}(\mathtt{ct}')) &\leq N \cdot L \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathtt{ksk})) + \mathsf{Var}(\mathsf{err}(\mathtt{ACC})) \\ &\leq N \cdot L \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathtt{ksk})) + (q/Q)^2 \cdot n \cdot N \cdot \ell \cdot \gamma^2 \cdot \mathsf{Var}(\mathsf{err}(\mathbf{c}_{\mathtt{Mux}})) \\ &\quad + (q/Q)^2 \cdot 4 \cdot N \cdot \varsigma^2 + 1 + 16 \cdot N \cdot \varsigma^2\end{aligned}$$

where L is the dimension of the key switching key.

After the $\mathtt{for}$ loop from line 3 to 5, the message of the resulting $\mathtt{ACC}$, $\mathsf{msg}(\mathtt{ACC})$, is $\left\lfloor \frac{Q}{8} \right\rceil \cdot X^{\mathtt{ct}_{2\cdot N} \cdot \mathsf{col}_0(\mathbf{F})} \cdot X^{N/2} \cdot \sum_{i=0}^{N-1} X^i$. If we have $|\mathtt{ct} \cdot \mathsf{col}_0(\mathbf{F})| < q/4$, then the ciphertext $\mathtt{ct}$ is encrypting the value zero. This follows from the fact that then

$-N/2 < |\mathtt{ct}_{2\cdot N} \cdot \mathsf{col}_0(\mathbf{F})| \leq N/2$ and thus the constant term of the $\mathsf{msg}(\mathtt{ACC})$ is $-\lfloor Q/8 \rceil$, i.e. the constant term of $\mathsf{msg}(\mathtt{ACC})$ in line 6 is zero. If, however, $|\mathtt{ct} \cdot \mathsf{col}_0(\mathbf{F})| < 3 \cdot q/4$ then the ciphertext $\mathtt{ct}$ is encrypting the value one. In this case $N/2 < |\mathtt{ct}_{2\cdot N} \cdot \mathsf{col}_0(\mathbf{F})| \leq 3N/2$, hence the $\mathsf{msg}(\mathtt{ACC})$ is $\lfloor Q/8 \rceil$. Therefore, the constant term of $\mathsf{msg}(\mathtt{ACC})$ in line 6 is $\lfloor Q/4 \rceil$.

We now have the following heuristic for the output noise in average case.

Heuristic. Given $\mathtt{ct}$ encrypting a bit m, Algorithm 2 outputs an MNTRU ciphertext $\mathtt{ct}'$ encrypting the same bit. In addition, under the central limit heuristic, the noise contained in the output behaves as a Gaussian distribution, hence, with overwhelming probability, it satisfies the following bound

$$\|\mathsf{err}(\mathtt{ct}')\| \leq 6 \cdot \begin{pmatrix} N \cdot L \cdot \gamma^2 \cdot \mathcal{E}_{\mathtt{ksk}} + 4 \cdot (q/Q)^2 \cdot n \cdot N \cdot \ell \cdot \gamma^2 \cdot \mathcal{E}_{\mathtt{bsk}} \\ +(q/Q)^2 \cdot 4 \cdot N \cdot \varsigma^2 + 1 + 16 \cdot N \cdot \varsigma^2 \end{pmatrix}^{1/2} \tag{3}$$

where $\mathcal{E}_{\mathtt{ksk}} = O(\mathsf{Var}(\mathsf{err}(\mathtt{ksk})))$ and $\mathcal{E}_{\mathtt{bsk}} = O(\mathsf{Var}(\mathsf{err}(\mathtt{bsk})))$.

The following theorem states that our scheme requires a modulus q that is asymptotically less than the fatigue point as stated in [13].

Theorem 1. *If the output of Algorithm 2 satisfies (3) except with negligible probability and $q = \tilde{O}(n)$, the output of Algorithm 2 can be correctly decrypted except with negligible probability.*

Proof. Since $N \in \Theta(n)$, $q/Q, \mathcal{E}_{\mathtt{bsk}}, \mathcal{E}_{\mathtt{ksk}} \in O(1)$, and $\ell, L \in O(\log Q) = O(\log N)$ and (3) is satisfied, the final noise after bootstrapping is $\tilde{O}(n)$ except with negligible probability. For correctness, Lemma 2 imposes that the sum of two input fresh/refreshed ciphertexts noises should be smaller than $(q-20)/8$. Thus the bound of each refreshed noise needs to be less than $(q-20)/16$, which implies we need $\|\mathsf{err}(\mathtt{ct}')\| < (q-20)/16 = q/16 - 5/4$ to recover the correct message. Therefore, it is sufficient to choose $q \in \tilde{O}(n)$.

We will discuss the concrete value q based on the above heuristic and theorem in Section 6.

5.2 Bootstrapping an LWE-based scheme

As mentioned our external product costs only ℓ multiplications on R_Q versus $4 \cdot \ell'$ in TFHE. In general, our base scheme constructed on top of the matrix NTRU problem requires a larger dimension n than in an LWE-based scheme to achieve the same security level. Since the bootstrapping procedure uses n external products, we can obtain a faster FHE scheme by replacing our base scheme by the LWE-based one used in FHEW and TFHE, and using NGS to bootstrap it. This minimizes the number of external products and also makes each one of them cheaper.

Hence, we propose to use our NGS scheme as the accumulator to refresh LWE ciphertexts as opposed to MNTRU ciphertexts. The decryption function is

essentially the same, i.e. the inner product between the ciphertext and the secret key. Since the LWE secret key can be binary, we can use binary homomorphic `CMux` gates instead of the ternary ones. However, at the end of the main loop of the refreshing procedure, we obtain an NTRU ciphertext of the form $c = g/f + \epsilon + \Delta \cdot m \in R_Q$, where ϵ is the rounding error after modulus switching. Then we need to transform it again into an LWE ciphertext. So, we adapt our key-switching from Section 4 to also switch the underlying hard problem from NTRU to LWE.

NTRU to LWE key-switching: The goal of the following algorithm is to switch the form of a ciphertext from an NGS ciphertext to an LWE ciphertext encrypting the same message. Let $(\mathbf{A}, \mathbf{b})$ be an LWE sample with a secret key $\mathbf{s}$. Let $c = g/f + \epsilon + \Delta \cdot m$ be a scalar NGS ciphertext with a secret key f, where ϵ is the rounding error after modulus swtiching. Define the key-switching key as the following vector of LWE samples:

$$\mathtt{ksk}_{\mathsf{NTRU}\to\mathsf{LWE}} := (\mathbf{A}, \mathbf{b} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e} + \mathbf{P} \cdot \mathbf{f}_0)$$

with $\mathbf{A} \in \mathbb{Z}_q^{(N\cdot L)\times n}$, $\mathbf{e} \leftarrow \chi_{\sigma_e}^{N\cdot L}$, $\mathbf{f}_0 := \mathsf{col}_0(\boldsymbol{\Phi}(f)) \in \mathbb{Z}^N$, and $\mathbf{P} = \mathbf{I}_N \otimes \mathfrak{g}_{q,B_{\mathtt{ksk}}}$. Then, given a ciphertext $c = g/f + \epsilon + \Delta \cdot m \in R_q$, the key-switching from NTRU to LWE is defined as follows:

- $\mathsf{KeySwitch}_{\mathsf{NTRU}\to\mathsf{LWE}}(c, \mathtt{ksk}_{\mathsf{NTRU}\to\mathsf{LWE}})$:
  1. Parse $\mathtt{ksk}_{\mathsf{NTRU}\to\mathsf{LWE}}$ as $(\mathbf{A}, \mathbf{b})$
  2. $\mathbf{a} \leftarrow \mathsf{KeySwitch}(c, \mathbf{A})$
  3. $b \leftarrow \mathsf{KeySwitch}(c, \mathbf{b})$
  4. Output $\mathbf{c}' := (\mathbf{a}, b)$

That is, we decompose the coefficient vector of c and multiply by both components of $\mathtt{ksk}_{\mathsf{NTRU}\to\mathsf{LWE}}$. Thus, we define $\mathbf{y} := \mathfrak{g}^{-1}(\boldsymbol{\phi}(c)) \in \mathbb{Z}^{N\cdot L}$ and compute

$$\mathbf{c}' := (\mathbf{a}, b) = (\mathbf{y} \cdot \mathbf{A},\ \mathbf{y} \cdot \mathbf{b}) \in \mathbb{Z}_q^{n+1}.$$

Then, we can see that

$$b = \mathbf{a}\cdot\mathbf{s} + \mathbf{y}\cdot\mathbf{e} + \boldsymbol{\phi}(c)\cdot\mathbf{f}_0 = \mathbf{a}\cdot\mathbf{s} + \mathbf{y}\cdot\mathbf{e} + g_0 + \epsilon\cdot((1,\mathbf{0}) + 4\cdot\boldsymbol{\phi}(f')) + 4\cdot\epsilon'\cdot\boldsymbol{\phi}(m)\cdot\boldsymbol{\phi}(f') + \Delta\cdot m_0$$

where $\epsilon \in (-1/2, 1/2]$ and m_0 is the constant term of m. In other words, $(\mathbf{a}, b)$ is a valid LWE ciphertext of m_0.

Noise analysis. We see that the noise of the resulting LWE encryption equals to $\mathbf{y}\cdot\mathbf{e} + g_0 + \epsilon + \epsilon \cdot 4 \cdot \boldsymbol{\phi}(f') + 4 \cdot \epsilon' \cdot \boldsymbol{\phi}(m) \cdot \boldsymbol{\phi}(f')$ as defined in [10], with the variance of the noise satisfying:

$$\begin{aligned}
\mathsf{Var}(\mathsf{err}(\mathbf{c}')) &= \mathsf{Var}(\mathbf{y}\cdot\mathbf{e}) + \mathsf{Var}(g_0) + \mathsf{Var}(\epsilon_0) + 16\cdot\mathsf{Var}(\epsilon\cdot\boldsymbol{\phi}(f')) + 4\cdot\mathsf{Var}(\boldsymbol{\phi}(m)\cdot\boldsymbol{\phi}(f')) \\
&\le N\cdot L\cdot\mathsf{Var}(\mathbf{y})\cdot\mathsf{Var}(\mathbf{e}) + \mathsf{Var}(g) + 1 + 16\cdot N\cdot\mathsf{Var}(\epsilon)\cdot\mathsf{Var}(f') + 4\cdot\|m\|_2^2\cdot\varsigma^2 \\
&\le N\cdot L\cdot\mathsf{Var}(\mathbf{y})\cdot\mathsf{Var}(\mathbf{e}) + \mathsf{Var}(g) + 1 + 16\cdot N\cdot\varsigma^2 + 4\cdot\|m\|_2^2\cdot\varsigma^2 \\
&\le N\cdot L\cdot\mathsf{Var}(\mathbf{y})\cdot\mathsf{Var}(\mathbf{e}) + \mathsf{Var}(g) + 1 + 20\cdot N\cdot\varsigma^2 \\
&\le N\cdot L\cdot\gamma^2\cdot\sigma_e^2 + \mathsf{Var}(\mathsf{err}(c))
\end{aligned}$$

6 Security analysis and parameter selection

The CPA-security of our NGS scheme follows directly from the decisional NTRU problem via a standard hybrid argument. Firstly, notice that $\mathbf{G}/\mathbf{F} + \mathbf{M}$ is a secure encryption as the MNTRU assumption states that $\mathbf{G}/\mathbf{F}$ is uniform mod q, then, using the circular security assumption, it is safe to encrypt $\mathbf{M}/\mathbf{F}$ instead of $\mathbf{M}$, i.e., a message that depends on the secret key $\mathbf{F}$, under the matrix NTRU problem. From this, we obtain the format $(\mathbf{G} + \mathbf{M})/\mathbf{F}$ used in our base scheme. Finally, the security of the bootstrapping follows from the (weak) circular security assumption that the NGS scheme can be used to encrypt the key of the base scheme, which in turn, encrypts the key of the NGS scheme. All these circular security assumptions are standard and are used extensively, e.g., [16,30]. In particular, it is not known how to construct FHE without the weak circular security used here.

Concrete security: Research on the security of the NTRU problem revealed a significant improvement of the performance of lattice reduction attacks on NTRU lattices with large moduli q, which are now known as the overstretched NTRU regime. Several works [1,9,24] showed the susceptibility of the overstretched regimes to attacks. The work of Kirchner and Fouque shows however that the attack is possible due to the choice of parameters and not due to the structure of the fields underlying the NTRU problem. The observation that the choice of parameters causes the attack, started a quest to determine the value of the ciphertext modulus q for which the overstretched regime of NTRU begins and hence the security issue occurs. This turning point is called the fatigue point. Kirchner and Fouque make a first attempt to estimate the fatigue point and their efforts result in an asymptotic upper bound, but it is only the recent work of Ducas and van Woerden [13] that achieves at finding a concrete value for the fatigue point for ternary NTRU.

To determine the fatigue point Ducas and van Woerden identified two events that distinguish the standard regime from the overstretched regime:

- Secret Key Recovery (SKR): The event in which a vector as short as a secret key vector is inserted in the basis of the lattice.
- Dense Sublattice Discovery (DSD): The event in which a vector of the dense sublattice generated by the secret key is inserted in the basis of the lattice. This vector is strictly longer than the secret key, but nevertheless this event leads to a successful attack as either the SKR event follows quickly after the DSD event, the DSD events cascade and generate the dense sublattice from which the secret key can be recovered or the discovered dense sublattice vector is in itself sufficient to decrypt fresh ciphertexts.

Based on an exploration of the occurrence of one of these events, Ducas and van Woerden present an analysis that discovers the fatigue point, which is determined by the value for q for which the DSD attack starts to be more efficient than the SKR attack. To get then an idea of how secure the NTRU problem with

this q value still is, they also determine the precise cost of the attacks in the overstretched regime. Their analysis uses the BKZ lattice reduction algorithm and does not focus on a single position but predicts the most relevant positions in which the vector of the SKR or DSD event can occur and takes all these positions into account. This refined analysis leads to the following asymptotic result; the fatigue point of NTRU with ternary secrets happens at $q = n^{2.484+o(1)}$. As well as the determination of this asymptotic result, they perform an average case analysis based on the volume of the relevant lattices and sublattices to arrive at a concrete prediction of the fatigue point instead of a worst-case bound. This concrete prediction puts the fatigue point at $q \approx 0.004 \cdot n^{2.484+o(1)}$ for $n > 100$. This average case analysis differentiates the circulant version of NTRU from its matrix version, as there are minor deviations in the volumes of the relevant sublattices. Our work uses the anti-circulant and matrix versions of NTRU as defined in Section 2.4. We argue that the change from the circulant to the anti-circulant version of NTRU does not reduce the security of our NTRU instance, since by using $X^N + 1$ instead of $X^N - 1$, we avoid any weaknesses caused by evaluation at one, which the circulant variant could suffer from. In addition, it does not invalidate the analysis made by Ducas and van Woerden, as that is based on the expected volume of the dense sublattice, which remains the same when $X^N - 1$ is replaced by $X^N + 1$.

Parameter selection: Using the analysis by Ducas and van Woerden [13] and the scripts that they provided to estimate the concrete hardness of NTRU[4], given the dimension, the modulus q, the variance σ^2, and taking into account the distribution of the secret key, we are able to find the block size β needed by BKZ to break the (matrix) NTRU problem. To convert β to the security level, we used the same (classical) cost model used by TFHE, namely, the number of operations of BKZ-β in dimension d was estimated as $T(d, \beta) := 2^{0.292 \cdot \beta + 16.4 + \log_2(8 \cdot d)}$, where $d = 2 \cdot n$ for the NTRU in dimension n. Thus, a security level of λ bits means that $T(d, \beta) \geq 2^\lambda$.

Hence, to choose the parameters of the NGS scheme, we fixed $N = 1024$ and ternary secrets, then found the maximum value of $\log Q$ that gives us $\lambda = 128$. For the scheme based on the matrix NTRU problem, we fixed $n = 800$ and also used ternary secrets. We chose the parameters for the LWE problem using the LWE estimator [2]. The decomposition bases used in the external product and in the key-switching were then chosen to guarantee correctness. We remark that instead of using a single basis B for all external products, we used B_1 for the first n_1 products and B_2 for the last n_2 (thus, $n = n_1 + n_2$), as this allowed us to reduce the total number of polynomial products computed during the bootstrapping.

The average-case noise bounds determined in the previous section then allows us to compute concrete parameters for our scheme. All the parameters are shown in Table 1.

[4] https://github.com/WvanWoerden/NTRUFatigue

Table 1. The parameters used in both bootstrappings, depending on whether the underlying problem of the base scheme is the matrix NTRU or the LWE. The columns N and Q refer to the NGS scheme. For each basis B_i we have a different dimension $\ell_i := \lceil \log_{B_i}(Q) \rceil$ for n_i bootstrapping keys.

Base scheme	n	q	N	Q	(B_1, n_1)	(B_2, n_2)	$B_{\mathtt{ksk}}$	ℓ_1	ℓ_2
MNTRU	800	$131071 \approx 2^{17}$	2^{10}	$912829 \approx 2^{19.8}$	$(8, 750)$	$(16, 50)$	3	7	5
LWE	610	$92683 \approx 2^{16.5}$	2^{10}	$912829 \approx 2^{19.8}$	$(8, 140)$	$(16, 470)$	3	7	5

7 Practical results

Among the three schemes that use the framework of fast bootstrapping with a base scheme and an accumulator [12,11,30], the most efficient one is TFHE. Therefore, we compare our practical results only with TFHE. Similar to the gate bootstrapping in TFHE, we are able to compute a binary gate through a bootstrapping. Therefore we use the bootstrapping as benchmark, as any speedup on the bootstrapping translates directly to the same speedup on any binary circuit. Like for TFHE, the encryption parameters of our schemes stay fixed for any binary circuit. We implemented a proof-of-concept of our bootstrapping procedures in C++. For a fair comparison, we chose the TFHE library as it is written in C++ and is up-to-date. Our code is publicly available[5].

We compiled TFHE with the same FFT library we used in our implementation, namely, FFTW [15]. Moreover, we also compiled our code with the same optimization flags already used by the 'optimal' mode of TFHE. Both TFHE and our implementation use a deterministic decomposition for the external product and also a deterministic rounding for the modulus switching, relying thus on the heuristic assumption that the noise terms obtained during the homomorphic evaluations follow independent subgaussian distributions. All the experiments were conducted on a single core of a machine with 8 GB of RAM and a 3.1 GHz Dual-Core Intel Core i5.

As the fast Fourier transforms (FFT) and element-wise Hadamard vector products dominate the running time of bootstrapping, we used the following formulas to compute $\mathtt{ACC} \boxdot \mathtt{CMux}(c_i)$ in the bootstrapping algorithm (Algorithm 2)

$$((X^{c_i} - 1) \cdot \mathtt{ACC}) \boxdot (\mathtt{bsk}_{i,0} - \mathtt{bsk}_{i,1} \cdot X^{-c_i}) + \mathtt{ACC} \qquad \text{(MNTRU)},$$
$$((X^{c_i} - 1) \cdot \mathtt{ACC}) \boxdot \mathtt{bsk}_i + \mathtt{ACC} \qquad \text{(LWE)}.$$

The LWE formula is actually used in the TFHE library. Notice that no polynomial multiplication is needed to compute $(X^{c_i} - 1) \cdot \mathtt{ACC}$; it can be done by one negacyclic shift of the coefficients of $\mathtt{ACC}$ and N subtractions in $\mathbb{Z}_Q$. Assuming that the bootstrapping keys are FFT transformed in advance, the external product requires $\ell_i + 1$ FFTs and ℓ_i Hadamard vector products where ℓ_j is the length of $\mathtt{bsk}_i$, $\mathtt{bsk}_{i,0}$ or $\mathtt{bsk}_{i,1}$. In addition, MNTRU requires extra ℓ_j Hadamard vector products to compute $(\mathtt{bsk}_{i,0} - \mathtt{bsk}_{i,1} \cdot X^{-c_i})$.

[5] https://github.com/KULeuven-COSIC/FINAL

In TFHE, the bootstrapping key is composed of $n' := 630$ GSW ciphertexts, where n' is the dimension of the LWE problem used in their base scheme. Moreover, for the GSW ciphertext, they used the ring $R_{q'} := \mathbb{Z}_{q'}[X]/\langle X^{N'}+1\rangle$, where $q' := 2^{32}$ and $N' = 1024$, but they could set a larger decomposition base than the ones we could use, and they can also ignore the least significant bits during the decomposition, since in the RLWE problem, these bits are noisy, thus, they obtain $\ell' := 3$. However, each GSW ciphertext is composed of $4 \cdot \ell'$ elements of $R_{q'}$, on the other hand our NGS ciphertexts only have ℓ ring elements. Thus, the size of the bootstrapping key in TFHE is $4 \cdot n' \cdot \ell' \cdot N' \cdot \log(q') = 31$ MB.

Since each external product costs $4 \cdot \ell'$ products in $R_{q'}$ for TFHE, their total cost is $4 \cdot n' \cdot \ell' = 7560$ ring multiplications. However, the slowest operations of the bootstrapping are forward and backward FFTs. Since the FFTs of the bootstrapping key are precomputed, the 'for' loop of the bootstrapping has to decompose only the RLWE sample that is accumulating the result, obtaining thus $2 \cdot \ell'$ ring elements. Then, it computes the FFT of these elements, performs the external product to obtain a new RLWE sample in the FFT domain and finally apply two inverse FFTs. Hence, TFHE needs $2 \cdot n' \cdot (\ell' + 2) = 6300$ FFTs per bootstrapping.

In our case, the bootstrapping key is composed by $2 \cdot n$ NGS ciphertexts when the base scheme is based on the MNTRU problem and n when the LWE is used. Our ℓ is a little bigger than the $\ell' = 3$ used in TFHE, but we do not have the factor four in the dimension of the NGS ciphertexts.

The size of each ciphertext, the number of ring multiplications, and the amount of FFTs we have to perform when the LWE problem is used in the base scheme ends up being smaller than what is needed by TFHE. In particular, considering the parameters presented in Table 1, the number of FFTs per bootstrapping is $n_1 \cdot (\ell_1 + 1) + n_2 \cdot (\ell_2 + 1)$, where $\ell_i := \lceil \log_{B_i}(Q) \rceil$. A detailed comparison is presented in Table 2. Since every integer in our implementation is represented by the `int` type, we assume every coordinate or coefficient of our keys occupies 32 bits of memory.

We ran our bootstrapping procedures one thousand times and estimated the standard deviation of the noise of refreshed ciphertexts as $\sigma_{\mathsf{LWE}} = 2^{9.46}$ and $\sigma_{\mathsf{NTRU}} = 2^{9.85}$, which gives us the following decryption failure probabilities: $p_{\mathsf{LWE}} = 1 - \mathrm{erf}(92683/(16 \cdot \sigma_{\mathsf{LWE}} \cdot \sqrt{2})) < 2^{-52}$ and $p_{\mathsf{NTRU}} = 1 - \mathrm{erf}(131071/(16 \cdot \sigma_{\mathsf{NTRU}} \cdot \sqrt{2})) < 2^{-60}$.

The number of FFTs and multiplications shown in Table 2 are computed using the parameters of each scheme as described in the abovementioned explanation. For the running times, we measured the average time of the NAND gate plus bootstrapping over 1000 runs.

As shown in Table 2, our bootstrapping algorithm for LWE ciphertexts is 28% faster than TFHE. Furthermore, our method nearly halves the total size of key-switching and bootstrapping keys. Namely, TFHE needs 71 MB of key material whereas our approach generates less than 39.3 MB.

Our bootstrapping algorithm for MNTRU ciphertexts is less efficient than TFHE. The first reason is that MNTRU requires a bigger dimension n than LWE

Table 2. Practical results of TFHE and of our bootstrapping procedures considering the two base schemes. The last collum shows the average bootstrapping running time over 1000 executions.

	Key switching key	Bootstrapping key	Mult. on R_Q	FFTs	Run. time
TFHE [11]	40 MB	31 MB	7560	6300	66 ms
MNTRU	34.4 MB	43 MB	11000	6300	92 ms
LWE	26.3 MB	13 MB	3330	3940	48 ms

to achieve the same security level given that the ciphertext modulus is fixed. In our experiments (see Table 1), $n = 800$ for MNTRU whereas $n' = 630$ in TFHE. The second reason is that the secret key of the MNTRU scheme is ternary. To handle ternary coefficients of the secret key, the `CMux` operation performs more multiplications in the FFT domain, namely $2 \cdot (\ell_1 \cdot n_1 + \ell_2 \cdot n_2)$.

However, the efficiency downgrade of our bootstrapping method for MNTRU ciphertexts is not critical in practice. The bootstrapping takes less than 0.1 seconds on an average commodity laptop with only 9% increase of the key material size. Hence, if one needs an FHE scheme based solely on NTRU, our scheme is a practical candidate for that.

8 Conclusion and future work

We showed that it is possible to construct an efficient FHE scheme based on the NTRU assumption and to instantiate it by setting parameters that are below the "fatigue point" where the sublattice attacks start to apply. This shows that with the current knowledge on the security of NTRU, it seems possible to construct competitive FHE based solely on the NTRU assumption, which motivates further research on NTRU-based FHE schemes. Moreover, we showed that by combining the LWE and the NTRU problems, we can construct an FHE scheme that runs faster and requires less key material than TFHE, which currently has the fastest bootstrapping procedure.

We notice that it would be possible to use better parameters for our scheme, and thus, increase the difference between our running time and TFHE's if we sampled the NTRU secrets f and g with different variances. Namely, the final noise introduced by the bootstrapping depends more on the norm of g than on the norm of f, thus, we could increase the variance of f without having too much impact on the final noise. Intuitively, the NTRU problem should only become harder as the variance of one of its secret increases, thus, this would allow us to increase q. Finally, having a larger value of q for (almost) the same final noise means that we can choose larger decomposition bases, hence, reduce the number of FFTs and Hadamard vector products per external product. However, since there is no formal analysis of the concrete hardness of NTRU with different variances of the secrets, we prefer to leave this as an interesting future work.

As another possible line of work, one could consider the circuit bootstrapping from TFHE, which takes an LWE ciphertext $\mathbf{c} \in \mathbb{Z}_q^{n'+1}$ encrypting a message m

and outputs a GSW ciphertext $\mathbf{C} \in R_q^{2\ell' \times \ell'}$ encrypting m with noise independent of the noise of $\mathbf{c}$. In other words, the circuit bootstrapping refreshes $\mathbf{c}$ and transforms it into a GSW ciphertext. This is done by executing ℓ' bootstrappings and $2\ell'$ key switchings, and requires two key-switching keys. However, in our case we would produce an NGS ciphertext $\mathbf{c} \in R_q^{\ell}$, so just ℓ key switchings are needed instead of $2\ell'$, and also only one key-switching key instead of two. Thus, both the running time and the memory usage can be reduced if we are able to use $\ell < 2\ell'$ in our scheme.

Acknowledgements

We would like to thank Leo Ducas for helpful discussions about the security of the NTRU problem.

This work has been supported in part by ERC Advanced Grant ERC-2015-AdG-IMPaCT, by the Research Foundation – Flanders (FWO) under an Odysseus project GOH9718N and a Junior Postdoctoral Fellowship, by CyberSecurity Research Flanders with reference number VR20192203, and by the Defence Advanced Research Projects Agency (DARPA) under contract No. HR0011-21-C-0034 DARPA DPRIVE BASALISC.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ERC, DARPA, the US Government, Cyber Security Research Flanders or the FWO. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

1. Albrecht, M.R., Bai, S., Ducas, L.: A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 153–178. Springer, Heidelberg (Aug 2016)
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology 9(3), 169–203 (2015), https://doi.org/10.1515/jmc-2015-0016
3. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 297–314. Springer, Heidelberg (Aug 2014)
4. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme. In: Stam, M. (ed.) 14th IMA International Conference on Cryptography and Coding. LNCS, vol. 8308, pp. 45–64. Springer, Heidelberg (Dec 2013)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012)
6. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd FOCS. pp. 97–106. IEEE Computer Society Press (Oct 2011)

7. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (Aug 2011)
8. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 395–412. ACM Press (Nov 2019)
9. Cheon, J.H., Jeong, J., Lee, C.: 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)
10. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 3–33. Springer, Heidelberg (Dec 2016)
11. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology 33(1), 34–91 (Jan 2020)
12. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (Apr 2015)
13. Ducas, L., van Woerden, W.: Ntru fatigue: How stretched is overstretched? In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021. pp. 3–32. Springer International Publishing, Cham (2021)
14. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
15. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE 93(2), 216–231 (2005), special issue on "Program Generation, Optimization, and Platform Adaptation"
16. Genise, N., Gentry, C., Halevi, S., Li, B., Micciancio, D.: Homomorphic encryption for finite automata. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 473–502. Springer, Heidelberg (Dec 2019)
17. Genise, N., Micciancio, D., Polyakov, Y.: Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 655–684. Springer, Heidelberg (May 2019)
18. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009), crypto.stanford.edu/craig
19. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (Aug 2013)
20. Hoffmann, C., Méaux, P., Ricosset, T.: Transciphering, using FiLIP and TFHE for an efficient delegation of computation. In: Bhargavan, K., Oswald, E., Prabhakaran, M. (eds.) INDOCRYPT 2020. LNCS, vol. 12578, pp. 39–61. Springer, Heidelberg (Dec 2020)
21. Jeon, S., Lee, H.S., Park, J.: Efficient lattice gadget decomposition algorithm with bounded uniform distribution. IEEE Access 9, 17429–17437 (2021)
22. Joye, M.: On NTRU-ν-um modulo $X^N - 1$. Cryptology ePrint Archive, Paper 2022/1092 (2022), https://eprint.iacr.org/2022/1092, https://eprint.iacr.org/2022/1092

23. Joye, M., Paillier, P.: Blind rotation in fully homomorphic encryption with extended keys. In: Dolev, S., Katz, J., Meisels, A. (eds.) Cyber Security, Cryptology, and Machine Learning - 6th International Symposium, CSCML 2022, Be'er Sheva, Israel, June 30 - July 1, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13301, pp. 1–18. Springer (2022), https://doi.org/10.1007/978-3-031-07689-3_1
24. Kirchner, P., Fouque, P.A.: Revisiting lattice attacks on overstretched NTRU parameters. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 3–26. Springer, Heidelberg (Apr / May 2017)
25. Kluczniak, K.: 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, https://eprint.iacr.org/2022/089
26. Lee, C., Wallet, A.: Lattice analysis on MiNTRU problem. Cryptology ePrint Archive, Report 2020/230 (2020), https://eprint.iacr.org/2020/230
27. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Karloff, H.J., Pitassi, T. (eds.) 44th ACM STOC. pp. 1219–1234. ACM Press (May 2012)
28. Micciancio, D., Polyakov, Y.: Bootstrapping in fhew-like cryptosystems (2021), https://doi.org/10.1145/3474366.3486924
29. Park, J., Tibouchi, M.: SHECS-PIR: Somewhat homomorphic encryption-based compact and scalable private information retrieval. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Part II. LNCS, vol. 12309, pp. 86–106. Springer, Heidelberg (Sep 2020)
30. Pereira, H.V.L.: Bootstrapping fully homomorphic encryption over the integers in less than one second. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 331–359. Springer, Heidelberg (May 2021)
31. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (May / Jun 2010)