# Ring packing and amortized FHEW bootstrapping [*]

Daniele Micciancio
UCSD
daniele@cs.ucsd.edu

Jessica Sorrell
UCSD
jlsorrel@cs.ucsd.edu

May 29, 2018

**Abstract**

The FHEW fully homomorphic encryption scheme (Ducas and Micciancio, Eurocrypt 2015) offers very fast homomorphic NAND-gate computations (on encrypted data) and a relatively fast refreshing procedure that allows to homomorphically evaluate arbitrary NAND boolean circuits. Unfortunately, the refreshing procedure needs to be executed after every single NAND computation, and each refreshing operates on a single encrypted bit, greatly decreasing the overall throughput of the scheme. We give a new refreshing procedure that simultaneously refreshes $n$ FHEW ciphertexts, at a cost comparable to a single-bit FHEW refreshing operation. As a result, the cost of each refreshing is amortized over $n$ encrypted bits, improving the throughput for the homomorphic evaluation of boolean circuits roughly by a factor $n$.

## 1 Introduction

Since Gentry's first construction of a Fully Homomorphic Encryption (FHE) scheme [Gen09], much research has been done to improve both the security and efficiency of FHE. On the security front, a line of works [GH11, BV14a, Bra12, BGV12, GSW13] has led to a FHE scheme of Brakerski and Vaikuntanathan [BV14b] based on worst-case lattice problems for *polynomial approximation factors*, and therefore essentially *as secure as* regular (non-homomorphic) lattice-based *public-key encryption* [Reg05].

On the efficiency front, major progress has been achieved too, but we are still very far from reaching the ideal goal of an FHE scheme *as efficient as public key encryption.* Brakerski, Gentry, and Vaikuntanathan [BGV12] give a scheme for homomorphic evaluation of circuits of depth $L$ and security parameter $\lambda$ requiring $\tilde{O}(\lambda \cdot L^3)$ per-gate computation. Gentry, Halevi, and Smart [GHS12c] used similar techniques to achieve homomorphic evaluation of width-$\Omega(\lambda)$ circuits with only polylog$(\lambda)$ per-gate computation. However, these schemes place restrictions on circuit depth or size, and, more importantly, rely on a substantially stronger assumption than that used for public-key encryption: namely, the worst-case hardness of lattice problems for *quasi-polynomial approximation factors.*

[*]Research supported in part by the Defense Advanced Research Project Agency (DARPA) and the U.S. Army Research Office under the SafeWare program, and the National Science Foundation (NSF) under grant CNS-1528068. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect reflect the views, position or policy of the Government.

Gentry's bootstrapping technique [Gen09] is still the only known method to achieve *fully* homomorphic encryption, i.e., an encryption scheme capable of evaluating homomorphically arbitrary circuits. We recall that Gentry's bootstrapping technique involves the homomorphic computation of the decryption function on an encryption of the decryption key, a rather complex operation, and this operation needs to be performed on all wires, for every few layers of the circuit. Therefore, improving the effectiveness of bootstrapping has been the main goal of many papers aimed at making FHE faster.

Improvements to bootstrapping have been pursued following two different approaches. The first approach, extensively studied in [BGH13, GHS12c, GHS12b, GHS12a, GHPS12, SV14, HS14, HS15, CZ17], involves the construction of FHE schemes that can pack several messages into a single ciphertext, and operate on them in parallel. While bootstrapping such a scheme may still be very expensive, it can simultaneously refresh a large number of ciphertexts in a single bootstrapping execution. This reduces the total number of times that the bootstrapping procedure needs to be executed (during the homomorphic evaluation of a sufficiently wide circuit,) effectively amortizing the bootstrapping cost over a large number of homomorphic operations. However, these schemes typically incur quasi-polynomial noise growth, and, consequently, require worst-case lattice assumptions for superpolynomial approximation factors. The only exception is the work of Chen and Zhang [CZ17], which showed how to refresh encryption schemes with packed ciphertexts using the bootstrapping approach of [AP14] (discussed below). This allows to use worst-case complexity assumptions with polynomial approximation factors (namely, the hardness of $GapSVP_{\tilde{O}(n^{6.5})}$), but at the cost of executing a costly bootstrapping procedure after each (packed) multiplication.

Building on [GSW13], a newer approach explored in [AP13, AP14, DM15, CGGI16] tries to reduce the cost of bootstrapping a single ciphertext as much as possible, even at the price of having to perform a bootstrapping operation for every gate of the circuit. Alperin-Sheriff and Peikert [AP14] introduced a bootstrapping technique requiring $\tilde{O}(\lambda)$ homomorphic operations. Building upon this technique, Ducas and Micciancio [DM15] brought the running time of a single bootstrapping execution down to a fraction of a second, with further improvements from Chillotti, Gama, Georgieva, and Izabachène [CGGI16]. However, these works come with the limitation that the bootstrapping procedure needs to be executed for essentially every gate of the circuit, and do not support packing several messages into a single ciphertext. So bootstrapping is much faster than, say, in HElib [HS14, HS15], but the amortized cost per gate is still quite high.

The goal of this work is to combine the advantages of these two approaches, and show how to simultaneously refresh $O(n)$ messages, but at a cost comparable to that of [AP13, AP14, DM15, CGGI16]. Our starting point is the FHEW bootstrapping method of [DM15]. We remark that FHEW has been improved in some follow-up works: [BR15, BDF18] extended the FHEW scheme to larger gates, and [CGGI16] further reduced the running time of bootstrapping, partly at the cost of making a stronger security assumption on Ring-LWE with binary secrets.[1] However, while practically relevant, both improvements are asymptotically modest: the method of [BR15, BDF18] is limited to gates with at most $O(\log n)$ input wires, and the speed-up achieved in [CGGI16] is just polylogarithmic. In fact, in this paper, we will make bootstrapping even slower than [DM15],

[1]The original FHEW implementation [DM15] was very careful to use binary secrets for regular LWE encryption (which admits some theoretical justification [MP13, BLP+13]), but used Ring-LWE encryption with secrets of size $\sqrt{n}$, which is the smallest bound for which any theoretical result has ever been proved [ACPS09]. In [CGGI16], and the version of our bootstrapping procedure that we present and analyze in this paper, Ring-LWE is used with binary secrets, which may be justifiable based on the best known cryptanalysis methods, but would still benefit from more theoretical investigations.

by a factor $O(n^\epsilon)$. The advantage is that, while the bootstrapping cost gets slightly higher, we will simultaneously refresh $O(n)$ messages, reducing the amortized bootstrapping cost per message by almost a factor of $n$ to $O(3^{1/\epsilon} n^\epsilon)$ for $\epsilon < 1/2$. This is the first FHE scheme based on the hardness of worst-case ideal lattice problems with *polynomial approximation factors* that is bootstrappable with sublinearly many homomorphic operations. A comparison to previous schemes with similar security assumptions may be found in Table 1.

| Bootstrapping Schemes with polynomial noise | | | | |
|---|---|---|---|---|
| Scheme | Total cost | # of ciphertexts | Amortized cost | Noise overhead |
| [AP14] | $\tilde{O}(n)$ | 1 | $\tilde{O}(n)$ | $\tilde{O}(n^2)$ |
| [DM15] | $\tilde{O}(n)$ | 1 | $\tilde{O}(n)$ | $\tilde{O}(n^{1.5})$ |
| [GINX16] | $\tilde{O}(n)$ | 1 | $\tilde{O}(n)$ | $\tilde{O}(n^{1.5})$ |
| [CGGI16] | $O(n)$ [2] | 1 | $O(n)$ | $\tilde{O}(n)$ |
| [CZ17] | $\tilde{O}(n^3)$ | $O(n)$ | $\tilde{O}(n^2)$ | $\tilde{O}(n^3)$ [3] |
| This work | $O(3^{1/\epsilon} n^{1+\epsilon})$ | $O(n)$ | $O(3^{1/\epsilon} n^\epsilon)$ | $\tilde{O}(n^{2+3/\epsilon})$ |

Table 1: Comparison of bootstrapping schemes based on hardness of lattice problems with polynomial approximation factors. The columns total running time of bootstrapping, the number of ciphertexts required to amortize the bootstrapping cost, the amortized bootstrapping running time (per ciphertext), and the required polynomial inapproximability factor. All running times hide polylogarithmic factors, and are expressed in terms of "basic" cryptographic operations on RingLWE (or similar) ciphertexts.

In the remainder of the introduction, we provide a detailed description of the FHEW bootstrapping problem, followed by a technical overview of the high level structure of our solution.

## 1.1 The FHEW bootstrapping problem

The starting point of this work is the "FHEW" fully homomorphic encryption scheme of [DM15]. In this overview, we assume some basic familiarity with LWE encryption, and use notation $\mathsf{LWE}_n^{t/q}[m, \delta]$ for the LWE encryption of a message $m \in \mathbb{Z}_t$, with secret key in $\mathbb{Z}_q^n$ and noise level $\delta$. Similarly, we write $\mathsf{RingLWE}_d^{t/q}[m, \delta]$ for Ring LWE ciphertexts over the $d$th cyclotomic ring encrypting a polynomial $m(X)$ of degree $\varphi(d)$.[4] The reader is referred to Section 2.4 for background information on LWE, and for a formal definition of the notation. In its most basic form, FHEW uses a function

$$\mathsf{HomNAND}\colon \mathsf{LWE}_n^{4/q}[\delta] \times \mathsf{LWE}_n^{4/q}[\delta] \to \mathsf{LWE}_n^{2/q}[\Delta] \tag{1}$$

mapping the encryption of two bits $b_0, b_1 \in \{0, 1\} \subset \mathbb{Z}_4$ (encoded as integers modulo 4), to the encryption of their logical "NAND", $b_0 \bar{\wedge} b_1 = \neg(b_0 \wedge b_1)$, but with somewhat larger noise $\Delta$ and encoded as an integer modulo 2. (For completeness, a formal definition and analysis of the $\mathsf{HomNAND}$ function is given in Appendix A.) This operation is extremely efficient, involving just a modest number of arithmetic operations modulo $q$, but it is not composable due to the different input and output encodings. In order to evaluate arbitrary circuits on encrypted data, [DM15] also provides

[4] We will be using primarily only two cyclotomic rings $\mathcal{R}_d = \mathbb{Z}[X]/(X^{d/2} + 1) \equiv \mathbb{Z}^{\varphi(d)}$ for $d = 2^k$ and $\varphi(d) = d/2$, and $\mathcal{R}_d = \mathbb{Z}[X]/(X^{2d/3} + X^{d/3} + 1) \equiv \mathbb{Z}^{\varphi(d)}$ for $d = 3^k$ and $\varphi(d) = 2d/3$.

a refreshing procedure

$$\mathsf{Refresh} : \mathsf{LWE}_n^{2/q}[\Delta] \to \mathsf{LWE}_n^{4/q}[\delta] \tag{2}$$

that maps noisy ciphertexts modulo 2, back to ciphertexts modulo 4 with low noise $\delta$. This refreshing involves the homomorphic computation of the decryption function, and it is rather costly: on a first approximation, it involves $\tilde{O}(n)$ homomorphic modular multiplications on data encrypted under a ring/symmetric-key variant of the GSW cryptosystem [GSW13], which we recall in Section 2.5.

Our goal is to show that one can *simultaneously* refresh a large number of ciphertexts (say, $O(n)$) at a cost comparable to a single FHEW refreshing: $\tilde{O}(3^{1/\epsilon}n^{1+\epsilon})$ homomorphic modular multiplications on GSW ciphertexts. This reduces the amortized cost of refreshing to just $O(3^{1\epsilon}n^{\epsilon})$ homomorphic (GSW) multiplications per ciphertext, rather than $O(n)$ as in the original FHEW cryptosystem. For any constant $\epsilon > 0$, this results in small polynomial amorized cost $\tilde{O}(n^{\epsilon})$ and polynomial approximation factors. It is also possible to set $\epsilon = 1/O(\sqrt{\log n})$, achieving subpolynomial amortized cost $\exp(O\sqrt{\log b}))$, but at the cost of a slightly superpolynomial approximation factor $n^{O(\sqrt{\log n})}$.

**Theorem 1** *For every* $0 < \epsilon < 1/2$*, there exists an algorithm* Refresh *which on input* $O(n)$ $\mathsf{LWE}_n^{2/q}[\Delta]$ *ciphertexts, refreshes them to* $\mathsf{LWE}_n^{4/q}[\delta]$ *ciphertexts of larger message space and smaller error, using* $\tilde{O}(3^{1/\epsilon}n^{1+\epsilon})$ *homomorphic operations, for* $0 < \epsilon < 1/2$.

## 1.2 High level outline

Our scheme involves a number of different parameters. As in the FHEW cryptosystem, we will use a "small" modulus $q$ and dimension $n = 2^{l-1}$ as parameters for the input ciphertexts. (We write $n = 2^{l-1}$ as we will frequently need to refer to $l = \log n + 1$). A larger modulus $Q$ is used by intermediate ciphertexts. We will give a procedure to simultaneously refresh $\varphi(d) = 2 \cdot 3^{k-1}$ FHEW ciphertexts, where $Q > q > n > d$. Details follow.

We start with $\varphi(d)$ (high noise) ciphertexts in $\mathsf{LWE}_n^{2/q}[\Delta]$, as produced by the FHEW HomNAND operation, working on LWE encryption in dimension $n$. The key idea required to simultaneously refresh all of them is to first combine them into a single RingLWE ciphertext, in a polynomial ring of degree $\varphi(d)$. Specifically, as a first step, we use a variant of the key switching technique from [BV14a] to evaluate a function[5]

$$\mathsf{PackLWE} : \left[\mathsf{LWE}_n^{2/q}[m_i, \Delta]\right]_{i<\varphi(d)} \to \mathsf{RingLWE}_d^{2/q}[m(X), \Delta'] \tag{3}$$

which maps $\varphi(d)$ *arbitrary* LWE ciphertexts (encrypting scalar messages $m_0, \ldots, m_{\varphi(d)-1}$) to a single ciphertext encrypting the polynomial $m(X) = \sum_{i<\varphi(d)} m_i \cdot X^i$. The details of this packing step are given in Section 3.

Before proceeding with homomorphic decryption of the packed ciphertext, we must switch its ring elements to a different, smaller, modulus. The final step of the homomorphic decryption procedure produces LWE ciphertexts of dimension equal to half the modulus of the packed ciphertexts,

[5]The function is also implicitly parameterized by a "packing key". But for simplicity, we do not show this parameter explicitly. Other functions in this section also receive similar implicit parameters, which, together, form the FHE public evaluation key.

and therefore to simplify the composition of HomNAND and Refresh operations, we choose the new modulus of the packed ciphertext to be $2n$. We use a ring version of the modulus switching technique of [BV14a] to compute a function

$$\mathcal{R}\text{-}\mathsf{ModSwitch}\colon \mathsf{RingLWE}_d^{2/q}[m(x), \Delta'] \to \mathsf{RingLWE}_d^{2/2n}[m(x), \Delta''] \tag{4}$$

mapping ciphertexts modulo $q$ under key $z \in \mathcal{R}_d/q$ to ciphertexts modulo $2n$. Details of the modulus switching operation are provided in Section 2.6.

We can now move on to homomorphically decrypt this Ring LWE ciphertext. Following the general bootstrapping framework of [Gen09], refreshing is performed by evaluating the decryption function homomorphically, on an encryption of the secret key. The homomorphic registers encrypting the entries of the secret key are implemented using a symmetric-key/ring variant of the GSW cryptosystem, similar to the FHEW accumulators [DM15]. These registers are denoted abstractly by $\mathsf{REG}^{2n/Q}[\cdot]$. Note that the message modulus $2n$ matches the ciphertext modulus of $\mathsf{RingLWE}_d^{2/2n}$, which is required for entrywise encryption of the RingLWE decryption key.

Using this notation, we can describe the refreshing procedure as the combination of two steps. The first is the primary technical contribution of this work, the homomorphic decryption function

$$\mathsf{RingDecrypt} : \mathsf{RingLWE}_d^{2/2n}[m(x), \Delta''] \to \left[\mathsf{REG}^{2n/Q}[\tilde{m}_i, \delta']\right]_{i<\varphi(d)} \tag{5}$$

which takes (as an implicit parameter) also the encryption $\mathsf{REG}^{2n/Q}[\mathsf{encode}(z)]$ of (a suitably encoded version of) the $\mathsf{RingLWE}^{2/2n}$ secret key $z \in \mathbb{Z}_{2n}^{\varphi(d)}$. The RingDecrypt function is homomorphic in $z$, and it is computed simply by evaluating (the linear component) of the RingLWE decryption function homomorphically on $\mathsf{REG}^{2n/Q}[\mathsf{encode}(z)]$.

At this point, two important remarks should be made about the RingDecrypt function, which is the core of our bootstrapping procedure:

- First, notice that, as long as $\Delta''$ is within the decryption bounds of $\mathsf{RingLWE}_d^{2/2n}$, this will produce a valid decryption, with an amount of noise $\delta'$ that depends only on the evaluation key $\mathsf{REG}^{2n/Q}[\mathsf{encode}(z)]$. So, this step is where the actual refreshing happens, producing output ciphertexts with much smaller noise $\delta' < \Delta''$ than the input.

- Second, since $\mathsf{REG}[\cdot]$ has $2n$ as a message modulus, the result of the computation must also be (the coordinate-wise encryption of) a polynomial $\tilde{m}(X)$ with coefficients $\tilde{m}_i \in \mathbb{Z}_{2n}$ (rather than $m_i \in \mathbb{Z}_2$ as the original message $m(X)$). The output of RingDecrypt will encrypt a noisy, scaled version of $m(X)$, satisfying $\tilde{m}(X) \approx n \cdot m(X)$.

Since $\mathsf{REG}[\cdot]$ encrypts each coefficient of $\tilde{m}(X)$ individually, the values $\mathsf{REG}^{2n/Q}[\tilde{m}_i, \delta']$ are already (refreshed, low-noise) ciphertexts of the original messages $m_0, \ldots, m_{d-1}$, but using a (noisy) input encoding $\tilde{m}_i$, a different cryptosystem and a large modulus $Q$. Each one of them is very similar to the intermediate output of the original FHEW refreshing procedure (2), as if we had computed it on each $\mathsf{LWE}_{2n}^{2/q}[m_i, \Delta]$ ciphertext individualy. So, they can be mapped to $\mathsf{LWE}_{2n}^{4/q}$ ciphertexts as in the original FHEW scheme by calling a "most-significant-bit" extraction function

$$\mathsf{msbExtract}\colon \mathsf{REG}^{2n/Q}[\tilde{m}_i, \delta'] \to \mathsf{LWE}_n^{4/Q}[m_i, \delta''] \tag{6}$$

and the standard modulus switching procedure

$$\mathsf{ModSwitch}\colon \mathsf{LWE}_n^{4/Q}[\delta''] \to \mathsf{LWE}_n^{4/q}[\delta'''] \tag{7}$$

for a LWE ciphertext under key $\mathbf{s} \in \mathbb{Z}_Q^{2n}$, which increases the noise $\delta''' \approx \frac{q}{Q}\delta'' + \sqrt{2\pi(1 + \|\mathbf{s}\|^2)}$ by a small additive term. Parameters will be set in such a way that $\delta''' \leq \delta$ is small enough to apply the HomNAND function (1) and keep computing on encrypted data. This completes the high level description of our bootstrapping method.

Since RingDecrypt is where the actual refreshing (i.e., noise reduction) takes place, and it is highly nontrivial, in the next section we give a detailed description of the challenges faced by an efficient evaluation of RingDecrypt, and the techniques we use to address them.

## 1.3 The challenge of homomorphic ring decryption

In the description above, we have claimed that RingLWE decryption can be performed using $\tilde{O}(d)$ arithmetic operations. This is indeed true, and can be achieved via standard FFT-based polynomial multiplication algorithms. However, in our application, this operation needs to be performed homomorphically on an encryption of the decryption key, which is a much more challenging task. The problem is that when encryption is performed in the exponent, as in the Ring-GSW-based FHEW accumulators, one is extremely limited in the type of linear operations that can be computed.

We recall that FHEW accumulators (and the cryptographic registers used in this paper) encode a message $v \in \mathbb{Z}_N$ using a ring variant of the GSW cryptosystem with, as a message space, the set of polynomials in a formal variable $X$ of degree bounded by $\varphi(N)$. These polynomials are used to encode scalar values, mapping each $v \in \mathbb{Z}_N$ to the monomial $X^v$. Encoding $v$ in the exponent limits the operations available to a candidate refreshing algorithm. Addition may be performed, using the multiplicatively homomorphic property of the GSW cryptosystem to compute $X^v \cdot X^w = X^{v+w}$, but other operations are not so straightforward. Multiplication by (known) scalars (mapping $X^v \mapsto X^{vc}$) requires homomorphic exponentiation, and even a simple subtraction or negation (mapping $X^v \mapsto X^{-v}$) would require homomorphic inversion, all operations unsupported by the GSW or any other known cryptosystem.

Standard FFT algorithms, on the other hand, require both the evaluation of addition and subtractions (in each "butterfly" of the FFT), as well as scalar multiplication by "twiddle" factors, i.e., powers of the root of unity used to compute the FFT. In order to support subtraction, we represent each register in a redundant way, holding both an encryption of $X^v$ and an encryption of $X^{-v}$. To reduce the number of scalar multiplications required by our refreshing procedure, we resort to a variant of Nussbaumer negacyclic convolution algorithm [Nus80].

In its original formulation, the Nussbaumer algorithm operates on polynomials in $X$ (where $X^{2^k} = 1$) by writing them as bivariate polynomials in $X, Y$, with both $X$ and $Y$ of degree at most $\sqrt{n} = \sqrt{2^k}$. Alternatively, one can look at these bivariate polynomials as univariate polynomials in $X$ with coefficients in $\mathbb{Z}[Y]$. By taking $Y = X^{\sqrt{n}}$, the coefficients belong to the ring $R = \mathbb{Z}[Y]/(Y^{\sqrt{n}} + 1)$, which admits $Y$ as a $2\sqrt{n}$th root of unity which can be used to compute the FFT of polynomials in $R[X]$. Multiplication by roots of unity can now be expressed simply by additions, subtractions and rotations of values in $\mathbb{Z}[Y]/(Y^{\sqrt{n}} + 1)$. Furthermore, because the FFT outputs elements of $R$, the algorithm can be recursively applied to each of the pointwise multiplications following the FFT.

Our refreshing procedure will require some modifications to the Nussbaumer algorithm as described above, which are detailed in Section 4.4. Most critically, if the algorithm is to be used recursively, we must be careful in bounding its recursive depth. The noise of the refreshing algorithm depends exponentially on the depth of the circuit computing it, and we must restrict ourselves to constant depth to achieve polynomial noise overhead. So rather than setting $Y = X^{\sqrt{n_i}}$ for $n_i = \sqrt[2^i]{n}$ at the $i$th level of recursion, we fix $Y = X^{n^\epsilon}$ for all steps. Therefore, for all $\epsilon$, the algorithm has recursive depth bounded by $1/\epsilon$. This is the main intuition behind our bootstrapping procedure: the Nussbaumer algorithm reduces polynomial multiplication to multiplications of many lower-degree polynomials, without introducing multiplications by constants or excessive noise overhead. A naive multiplication algorithm can then be used to compute the smaller polynomial products, and the transform inverted to give the encrypted product required for homomorphic ring decryption.

## 2 Preliminaries

### 2.1 Basic notation

We write column vectors over a ring $\mathcal{R}$ with bold font $\mathbf{a} \in \mathcal{R}^n$. Matrices are similarly written in capitalized bold font as $\mathbf{A} \in \mathcal{R}^{n\times m}$. The $L^2$ norm of a vector $\mathbf{a} = (a_1, \ldots, a_n) \in \mathcal{R}^n$ is $\|\mathbf{a}\| = \sqrt{\sum_i |a_i|^2}$. The concatenation of elements $a, b, \ldots$ into a row vector is written as $[a, b, \ldots]$. We write $(a, b, \ldots)$ for concatenation as a column vector.

### 2.2 Distributions

A random variable $X$ has *subgaussian* distribution over $\mathbb{R}$ of parameter $\alpha$ if its tails are dominated by a Gaussian of parameter $\alpha$, so that $\Pr\{|X| \geq t\} \leq 2e^{-\pi t^2/\alpha^2}$ for all $t \geq 0$. A subgaussian variable $X$ with parameter $\alpha > 0$ satisfies $E[e^{2\pi t X}] \leq e^{\pi\alpha^2 t^2}$, for all $t \in \mathbb{R}$. We note that a centered random variable $X$, where $|X| \leq \beta$ always holds, is subgaussian, specifically with parameter $\beta\sqrt{2\pi}$. For example the randomized rounding function $\lceil(x)\rfloor_\$$ (which takes value $\lfloor x \rfloor$ with probability $\lceil x \rceil - x$, and equals $\lceil x \rceil$ otherwise) is $\sqrt{2\pi}$-subgaussian. A random vector $\mathbf{x}$ of dimension $n$ is subgaussian of parameter $\alpha$ if for all unit vectors $\mathbf{u} \in \mathbb{R}^n$, its one-dimensional marginals $\langle \mathbf{u}, \mathbf{x}\rangle$ are also subgaussian of parameter $\alpha$. This extends to random matrices, where $\mathbf{X}^{m\times n}$ is subgaussian of parameter $\alpha$ if for all unit vectors $\mathbf{u} \in \mathbb{R}^m, \mathbf{v} \in \mathbb{R}^n$, $\mathbf{u}^t\mathbf{X}\mathbf{v}$ is subgaussian of parameter $\alpha$. It follows immediately from these definitions that the concatenation of independent subgaussian vectors, all with parameter $\alpha$, interpreted as either a vector or matrix, is also subgaussian with parameter $\alpha$.

### 2.3 Cyclotomic Rings

For any positive integer $N$, let $\Phi_N(X) = \prod_{j\in\mathbb{Z}_N^*}(X - \omega_N^j)$ be the $N$th cyclotomic polynomial, where $\omega_N = e^{2\pi \mathsf{i}/N} \in \mathbb{C}$ is the complex $N$th principal root of unity, and $\mathbb{Z}_N$ is the group of invertible integers modulo $N$. We recall that $\Phi_N(X) \in \mathbb{Z}[X]$ is a monic polynomial of degree $\varphi(N) = |\mathbb{Z}_N^*|$ with integer coefficients. The corresponding ring $\mathcal{R}_N = \mathbb{Z}[X]/\Phi_N(X)$ of integer polynomials modulo $\Phi_N$ is called the $N$th cyclotomic ring. This ring can be identified with $\mathcal{R}_N \equiv \mathbb{Z}^{\varphi(N)}$ (as additive groups) representing each element $a \in \mathcal{R}_N$ by a polynomial of degree less than $\varphi(N)$, and mapping this polynomial $a(X) = \sum_{j<\varphi(N)} a_j \cdot X^j$ to its coefficient vector $(\!(a)\!) = (a_0, \ldots, a_{\varphi(N)-1}) \in \mathbb{Z}^{\varphi(N)}$. For any ring element $a \in \mathcal{R}_N$, $\|a\|$ is taken to mean the $L^2$ norm of the corresponding vector $(\!(a)\!) \in \mathbb{Z}^{\varphi(N)}$. Ring elements $a, b \in \mathcal{R}_N$ also admit a matrix representation $\mathbf{M}_a \in \mathbb{Z}^{\varphi(N)\times\varphi(N)} = [(\!(a \cdot X^0)\!), (\!(a \cdot X^1)\!), \ldots, (\!(a \cdot X^{\varphi(N)-1})\!)]$ (used in Section 4.5,) such that $\mathbf{M}_a \cdot (\!(b)\!) = (\!(a \cdot b)\!)$. For any positive integer $q$, we write $\mathcal{R}_N/q$ for the quotient $\mathcal{R}_N/(q\mathcal{R}_N)$, i.e., the ring of polynomials $\mathcal{R}_N$ with coefficients reduced modulo $q$. Notice that $\mathcal{R}_N/q \equiv \mathbb{Z}_q^{\varphi(N)}$ as additive groups.

For concreteness, in this paper we only use cyclotomic rings $\mathcal{R}_N$ for two special types of the index $N$:

- $N = n = 2^l$, giving the polynomial ring $\mathcal{R}_n = \mathbb{Z}[X]/(X^{n/2} + 1)$ of degree $\varphi(n) = n/2$, and
- $N = d = 3^k$, giving the polynomial ring $\mathcal{R}_d = \mathbb{Z}[X]/(X^{2d/3} + X^{d/3} + 1)$ of degree $\varphi(d) = 2d/3$.

In particular, $\mathcal{R}_d/q \equiv \mathbb{Z}_q^{2d/3}$ (for $d = 3^k$) and $\mathcal{R}_n/q \equiv \mathbb{Z}_q^{n/2}$ (for $n = 2^l$).

**Fact 1 (Recall from [DM14], Fact 6)** *If $\mathcal{D}$ is a subgaussian distribution of parameter $\alpha$ over $\mathcal{R}_N$, and $\mathbf{R} \leftarrow \mathcal{D}^{w\times k}$ has independent coefficients drawn from $\mathcal{D}$, then, with overwhelming probability, we have $s_1(\mathbf{R}) \leq \alpha\sqrt{N} \cdot O(\sqrt{w} + \sqrt{k} + \omega(\sqrt{\log N}))$.*

## 2.4 (Ring) LWE Symmetric Encryption

In this subsection we introduce definitions and notation for the basic LWE encryption scheme that our bootstrapping procedure operates on. Following [DM15], we formulate the LWE encryption procedure using a randomized *rounding function* $\chi\colon \mathbb{R}^n \to \mathbb{Z}^n$ mapping each $\mathbf{x} \in \mathbb{R}^n$ to a distribution over $\mathbb{Z}^n$ such that $\chi(\mathbf{x} + \mathbf{v}) = \chi(\mathbf{x}) + \mathbf{v}$ for all integer vectors $\mathbf{v} \in \mathbb{Z}^n$. For any $\mathbf{x} \in \mathbb{R}^n$, the random variable $\chi(\mathbf{x}) - \mathbf{x}$ is called the rounding error of $\chi(\mathbf{x})$. As a special case, when the domain of $\chi$ is restricted to $\mathbb{Z}^n$, we have $\chi(\mathbf{x}) = \mathbf{x} + \chi(\mathbf{0})$, i.e., the randomized rounding function simply adds a fixed "noise" distribution $\chi(\mathbf{0})$ to the input $\mathbf{x} \in \mathbb{Z}^n$.

We consider a ring version of LWE symmetric encryption, parametrized by a (cyclotomic) ring $\mathcal{R}_N$, a dimension $n$, message-modulus $t \geq 2$, ciphertext modulus $q$ and randomized rounding function $\chi$. The message space of the scheme is $\mathcal{R}_N/t \equiv \mathbb{Z}_t^{\varphi(N)}$. (Typically, $\mathcal{R} = \mathbb{Z}$, the rounding function has error distribution $|\chi(x) - x| < q/2t$, and $t = 2$ is used to encrypt message bits.) The (secret) key of the encryption scheme is a vector $\mathbf{s} \in \mathcal{R}_N^n$, which may be chosen uniformly at random (modulo $q$), or as a random short vector. The $\mathcal{R}$-LWE encryption of a message $m \in \mathcal{R}_N/t$ under key $\mathbf{s} \in \mathcal{R}_N^n$ is

$$\mathcal{R}\text{-}\mathsf{LWE}_{\mathbf{s}}^{t/q}(m) = (\mathbf{a}, \chi(\mathbf{a}\cdot\mathbf{s} + mq/t) \bmod q) \in (\mathcal{R}_N^n/q) \times (\mathcal{R}_N/q) \equiv \mathcal{R}_N^{n+1}/q \tag{8}$$

where $\mathbf{a} \leftarrow \mathcal{R}_N^n/q$ is chosen uniformly at random. Notice that when $t$ divides $q$, the encryption of $m$ equals $(\mathbf{a}, \mathbf{a}\cdot\mathbf{s} + e + mq/t \bmod q)$, where the error $e$ is chosen according to a fixed noise distribution $\chi(0)$. The error of a ciphertext $(\mathbf{a}, b)$ is the random variable $\mathsf{err}(\mathbf{a}, b) = (b - \mathbf{a}\cdot\mathbf{s} - mq/t) \bmod q$ describing the rounding error, reduced modulo $q$ to the centered interval $[-q/2, q/2]$. Notice that the error $\mathsf{err}(\mathbf{a}, b)$ depends not just on $(\mathbf{a}, b)$, but also on $\mathbf{s}, q, t$ and $m$. Also, in the absence of any restriction on the error, a ciphertext $(\mathbf{a}, b)$ can be the encryption of any message $m$ with respect to any key $\mathbf{s}$. So, we always consider ciphertexts with bounded errors $\|\chi(\mathbf{x}) - \mathbf{x}\| \leq \beta$, as defined below.

**Definition 1 ((Ring) LWE ciphertexts)** *The set of all (Ring) LWE ciphertexts over (cyclotomic) ring $\mathcal{R}_N$, encrypting message $m \in \mathcal{R}_N/t$, under key $\mathbf{s} \in \mathcal{R}_N^n$, modulo $q$ and with error bound $\beta$, is*

$$\mathcal{R}_N\text{-}\mathsf{LWE}_{\mathbf{s}}^{t/q}[m, \beta] = \{(\mathbf{a}, b) \mid \mathbf{a} \in \mathbb{R}_N^n/q, \|\mathbf{a}\cdot\mathbf{s} - mq/t\| \leq \beta\}.$$

*When the value of the key $\mathbf{s} \in \mathcal{R}_N^n$ is clear from the context or unimportant, we simply write $\mathcal{R}_N\text{-}\mathsf{LWE}_n^{t/q}[m,\beta]$, where the subscript $n$ refers to the dimension of the secret $\mathbf{s} \in \mathcal{R}_N^n$.*

We use some abbreviated notation in the following important special cases:

- When $N = 1$, we omit $\mathcal{R}_N = \mathbb{Z}$, and write $\mathsf{LWE}_{\mathbf{s}}^{t/q}[m,\beta]$ (or $\mathsf{LWE}_n^{t/q}[m,\beta]$) for the set of standard ($\mathbb{Z}$-) LWE ciphertexts with $n$-dimensional secret $\mathbf{s} \in \mathbb{Z}^n$.
- When $n = 1$, and $\mathbf{s} = s \in \mathcal{R}_N$ is a single ring element, we write $\mathsf{RingLWE}_N^{t/q}$ as an abbreviation for $\mathcal{R}_N\text{-}\mathsf{LWE}_1^{t/q}$.

The (Ring) LWE decryption procedure plays a central role in our FHE bootstrapping process. Specifically, the definition of the FHEW bootstrapping problem directly involves the decryption of standard ($\mathbb{Z}$-) LWE ciphertexts, and our efficient bootstrapping procedure makes internal use of Ring LWE decryption. So, we formalize it in the following definition for an arbitrary (cyclotomic) ring $\mathcal{R}_N$.

**Definition 2 ((Ring) LWE decryption)** *The decryption of an $\mathcal{R}_N\text{-}\mathsf{LWE}_{\mathbf{s}}^{t/q}$ ciphertext $(\mathbf{a}, b) \in (\mathcal{R}_N^n, \mathcal{R}_N)/q$ is*

$$\mathsf{Dec}(\mathbf{s},(\mathbf{a},b)) = \lfloor t(b - \mathbf{a}\cdot\mathbf{s})/q \rceil \bmod t \in \mathcal{R}_N/t \equiv \mathbb{Z}_t^{\varphi(N)}. \tag{9}$$

It is easy to check that for all $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t/q}[m, q/2t]$, the decryption procedure correctly recovers the encrypted message:

$$\mathsf{Dec}(\mathbf{s},(\mathbf{a},b)) = \left\lfloor \frac{t}{q}\cdot\left(\frac{q}{t}m + e\right)\right\rceil = \left\lfloor m + \frac{t}{q}e \right\rceil = m \bmod t$$

because $\frac{t}{q}|e| < 1/2$.

## 2.5 Ring-GSW encryption

The cryptographic accumulators of [DM15] (and the extended cryptographic registers defined in our work) make use of a ring variant of the GSW encryption scheme [GSW13], which we now briefly describe. Let $\mathcal{R}_{N/Q}$ be the $N$th cyclotomic ring, modulo some suitably large integer $Q$. The Ring-GSW cryptosystem, encrypts a message $m \in \mathbb{Z}_N$ under key $z \in \mathcal{R}_{N/Q}$ as

$$\mathsf{GSW}_z^{N/Q}(m) = [\mathbf{a}, \mathbf{a}\cdot z + \mathbf{e}] + m\cdot \mathbf{g}\otimes \mathbf{I}_2 \tag{10}$$

where $\mathbf{g} = (B^0, B^1, B^2, \ldots, Q/B)$ for some base $B$. Similarly as for LWE, we write $\mathsf{GSW}_z^{N/Q}[m,\beta]$ for the set of ciphertexts encrypting $m$ under $z$ with error at most $\|\mathbf{e}\| \le \beta$. Decryption follows from the observation that the last row of a ciphertext is a Ring-LWE encryption of $m$ under $z$.

The Ring-GSW cryptosystem supports homomorphic addition and multiplication, and we will primarily use the latter. $\mathsf{GSW}_z^{N/Q}(m_0\cdot m_1) = C_0 \times C_1$ is computed by first expressing $C_0 = \sum_i B^i C_{0,i}$ as a sum of matrices with $B$-bounded (polynomial) entries, and then computing the matrix product $[C_{0,0}, \ldots, C_{0,\log Q}]\cdot C_1$. Letting $\mathbf{e}_0$ (resp. $\mathbf{e}_1$) denote the error vector of $C_0$ (resp. $C_1$), the result can be written $[\mathbf{a}, \mathbf{a}\cdot z + \mathbf{e}] + m_0 m_1 \cdot \mathbf{g}\otimes\mathbf{I}_2$, where $\mathbf{e} = \sum_i C_{0,i} e_{1,i} + m_1 \mathbf{e}_0$ depends asymmetrically on the error of the inputs. To minimize the error growth resulting from a sequence of multiplications of $\mathsf{GSW}$ ciphertexts (with similar initial error), then, the multiplications should be evaluated in a right-associative sequence.

## 2.6 Modulus Switching

In this subsection we describe the modulus switching procedures $\mathsf{ModSwitch}$ required by our bootstrapping algorithm. The methods described here are a minor variant of the modulus switching technique of [BV14a], making use of randomized rounding, and adapted to our notation.

**Lemma 1** *For any positive integers $Q, q \in \mathbb{Z}$, dimension $n$, and secret vector $\mathbf{s} \in \mathbb{Z}^n$, the modulus switching procedure of Algorithm 1 maps ciphertexts $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t/Q}[m, \beta]$ to ciphertexts $\mathsf{ModSwitch}_q(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t/q}[m, \beta']$, where $\beta' = \frac{q}{Q}\beta + \sqrt{2\pi(1 + \|\mathbf{s}\|^2)} = \frac{q}{Q}\beta + O(\|\mathbf{s}\|)$ with high probability over the randomized rounding procedure.*

Proof. The noise of the rounded ciphertext $\mathsf{ModSwitch}_q(\mathbf{a}, b) = (\mathbf{a}', b')$ is

$$\begin{aligned}
\mathsf{err}(\mathbf{a}', b) &= \left|b' - \mathbf{a}' \cdot \mathbf{s} - (q/t)m\right| \\
&\leq |\lceil (q/Q)b \rfloor_\$ - \lceil (q/Q)\mathbf{a} \rfloor_\$ \cdot \mathbf{s} - (q/t)m| \\
&\leq |(q/Q)b - (q/Q)\mathbf{a} \cdot \mathbf{s} - (q/t)m| + |\mathbf{e} \cdot (1, -\mathbf{s})| \\
&\leq \frac{q}{Q} |b - \mathbf{a} \cdot \mathbf{s} - (Q/t)m| + |\mathbf{e} \cdot (1, -\mathbf{s})| \\
&\leq \frac{q}{Q}\beta + |\mathbf{e} \cdot (1, -\mathbf{s})|
\end{aligned}$$

where $\mathbf{e}$ is the rounding vector. Since $\mathbf{e}$ is subgaussian of parameter $\sqrt{2\pi}$, the error is at most $\frac{q}{Q}\beta + \sqrt{2\pi(1 + \|\mathbf{s}\|^2)}$. □

**Lemma 2** *For any integers $n, q$, cyclotomic ring $\mathcal{R}_d$, and secret key $z \in \mathcal{R}_d$, the modulus switching procedure of Algorithm 2 maps ciphertexts $(a, b) \in \mathsf{RingLWE}_z^{t/q}[m, \beta]$ to ciphertexts $\mathsf{ModSwitch}_{R,n}(a, b) \in \mathsf{RingLWE}_z^{t/n}[m, \beta']$, where $\beta' = \frac{n}{q}\beta + \omega(\sqrt{d \log d}) \cdot \|z\|$ with high probability over the randomized rounding.*

Proof. The error terms of the rounded ciphertext are then bounded by

$$\begin{aligned}
\|b' - a' \cdot z - (n/t)m\| &\leq \|\lceil (n/q)b \rfloor_\$ - \lceil (n/q)a \rfloor_\$ z - (n/t)m\| \\
&\leq \|(n/q)b - (n/q)a \cdot z - (n/t)m\| + \|e_0\| + \|e_1 \cdot z\| \\
&\leq \frac{n}{q}\|b - a \cdot z - (q/t)m\| + \|e_0\| + \|e_1 \cdot z\| \\
&\leq \frac{n}{q}\beta + \|e_0\| + \|e_1 \cdot z\|
\end{aligned}$$

where $e_0, e_1 \in \mathcal{R}_d$ are subgaussian rounding vectors of parameter $\sqrt{2\pi}$. It follows that $s_1(e_0), s_1(e_1) \leq \omega(\sqrt{d \log d})$ with high probability. □

# 3 Ciphertext Packing

We describe a variant of the LWE key-switching technique that can be used to convert a set of $\varphi(d)$ LWE ciphertexts $\{(\mathbf{a}_i, b_i)\}$, each encrypting a message $m_i$, to a single "packed" Ring-LWE ciphertext encrypting the message $m(X) = \sum_i m_i X^{i-1}$.

**Algorithm 1** $\mathsf{ModSwitch}_q(\mathbf{a}, b)$

```
Input: $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t/Q}[m]$
Output: $(\mathbf{a}', b') \in \mathsf{LWE}_{\mathbf{s}}^{t/q}[m]$
$(\mathbf{a}', b') \leftarrow (\lceil (q/Q)\mathbf{a} \rfloor_{\$}, \lceil (q/Q)b \rfloor_{\$})$
return $(\mathbf{a}', b')$
```

**Algorithm 2** $\mathsf{ModSwitch}_{R,n}(a, b)$

```
Input: $(a, b) \in \mathsf{RingLWE}_{z}^{t/q}[m]$
Output: $(a', b') \in \mathsf{RingLWE}_{z}^{t/n}[m]$
$(\mathbf{a}', b') \leftarrow (\lceil (n/q)a \rfloor_{\$}, \lceil (n/q)b \rfloor_{\$})$
return $(a', b')$
```

**Lemma 3** *Algorithm 3 is a quasi-linear time algorithm that on input $\varphi(d)$ ciphertexts $c_i \in \mathsf{LWE}_{\mathbf{s}}^{t/q}(m_i, \Delta)$ (for $i = 0, \ldots, \varphi(d) - 1$, all under the same secret key $\mathbf{s} \in \mathbb{Z}_q^n$) and a packing key consisting of Ring-LWE encryptions $K_{j,l} \in \mathsf{RingLWE}_{\tilde{z}}^{q/q}[s_l 2^j, \beta_P]$ (for $l = 0, \ldots, n-1$, $j = 0 \ldots, \lceil \log q \rceil - 1$ and key $\tilde{z} \in R_d/q$), outputs a Ring-LWE encryption $c \in \mathsf{RingLWE}_{\tilde{z}}^{t/q}[m(X), \beta]$ of $m(X) = \sum_i m_i X^i$ under $\tilde{z}$ with error at most $\beta \in O(\sqrt{d}\Delta + \sqrt{dn \log q}\beta_P)$.*

Proof. Let $c_i = (\mathbf{a}_i, b_i)$ be the input ciphertexts (with $\mathbf{a}_i \in \mathbb{Z}_q^n$ and $b_i \in Z_q$) and $e_i = b_i - \langle \mathbf{a}_i, \mathbf{s} \rangle - (q/t)m_i$ the corresponding error terms. Define the ring elements $\tilde{e} = \sum_i e_i X^i$, $\tilde{b} = \sum_i b_i X^i \in \mathcal{R}_d/q$ and vectors $\tilde{\mathbf{a}}_j \in (\sum_i \{0, 1\} \cdot X^i)^n$, for $j = 0, \ldots, \log q$, such that

$$\sum_{j < \log q} \tilde{\mathbf{a}}_j 2^j = \sum_i \mathbf{a}_i \cdot X^{i-1} \in (\mathcal{R}_d/q)^n.$$

Let $(\tilde{a}'_{j,l}, \tilde{b}'_{j,l}) = K_{j,l}$. Then taking

$$\begin{aligned}
\tilde{a}'' &= -\sum_{j,l} \tilde{a}_{j,l} \tilde{a}'_{j,l} \\
\tilde{b}'' &= \tilde{b} - \sum_{j,l} \tilde{a}_{j,l} \cdot \tilde{b}'_{j,l} \\
\tilde{e}'' &= \tilde{e} - \sum_{j,l} \tilde{a}_{j,l} \tilde{e}'_{j,l}
\end{aligned}$$

we have

$$\begin{aligned}
\tilde{b}'' - \tilde{a}'' \cdot \tilde{z} - \tilde{e}'' &= \tilde{b} - \sum_{j,l} (\tilde{a}_{j,l} \cdot \tilde{b}'_{j,l} - \tilde{a}_{j,l} \tilde{a}'_{j,l} \cdot \tilde{z} - \tilde{a}_{j,l} \tilde{e}'_{j,l}) - \tilde{e} \\
&= \tilde{b} - \sum_{j,l} \tilde{a}_{j,l} (\tilde{b}'_{j,l} - \tilde{a}'_{j,l} \cdot \tilde{z} - \tilde{e}_{j,l}) - \tilde{e} \\
&= \tilde{b} - \sum_{j,l} (\tilde{a}_{j,l} 2^j s_l) - \tilde{e} \\
&= \tilde{b} - \sum_i \sum_l (a_{i,l} s_l X^i) - \tilde{e} \\
&= \sum_i (b_i - \mathbf{a}_i \cdot \mathbf{s} - e_i) X^i = \sum_i (q/t) m_i X^i \text{ as desired .}
\end{aligned}$$

Since $|e_i| \leq \Delta$ for $i = 0, \ldots, \varphi(d) - 1$, we have $\|\tilde{e}\| \leq \sqrt{\varphi(d)}\Delta$. The ring elements $\tilde{a}_{j,l}$ are independent subgaussians of parameter $\sqrt{2\pi}$. It follows from Fact 1 that with high probability the row vector $A = [\tilde{a}_{j,l}]$ has spectral norm $s_1(A) = O(\sqrt{dn \log q})$. Using $\|\tilde{e}'_{j,l}\| \leq \beta_P$, we get that the error of the output is at most $\beta = O(\sqrt{d}\Delta + \sqrt{dn \log q}\beta_P)$.

The ring packing procedure given in Algorithm 3 performs $O(n \log q)$ operations in the ring $\mathcal{R}_d/q$. Each arithmetic operation in this ring can be performed in quasilinear time $\tilde{O}(d)$. So, the total running time $\tilde{O}(dn)$ is quasilinear in the input size $O(dn \log q)$. □

**Algorithm 3** $\mathsf{PackLWE}([(\mathbf{a}_i, b_i)]_{i<\varphi(d)}, \mathcal{K}^P)$

```
Input: $[(\mathbf{a}_i, b_i)]_{i<\varphi(d)}$                ▶ $(\mathbf{a}_i, b_i) \in \mathsf{LWE}_n^{t/q}[m_i]$
       $\mathcal{K}^P = K^P_{j,l}$                            ▶ $(\tilde{a}'_{j,l}, \tilde{b}'_{j,l}) = K_{j,l} \in \mathsf{RingLWE}_{\tilde{z}}^{q/q}[s_l 2^j]$
Output: $(\tilde{a}'', \tilde{b}'')$                           ▶ $(\tilde{a}'', \tilde{b}'') \in \mathsf{RingLWE}_{\tilde{z}}^{t/q}[m(X))]$
for $l \in \{0, ..., n-1\}$ do
  $\bar{a}_l \leftarrow \sum_i a_{i,l} X^i$
  for $j \in \{0, ..., \log q\}$ do
    $\tilde{a}_{j,l} \leftarrow l^{th}$ entry of the binary decomposition of $\bar{a}_l$ such that $\sum_{l<\log q} \tilde{a}_{j,l} 2n = \bar{a}_l$
$\tilde{a}'' \leftarrow -\sum_{j,l} \tilde{a}_{j,l} \cdot \tilde{a}'_{j,l}$
$\tilde{b}'' \leftarrow \tilde{b} - \sum_{j,l} \tilde{a}_{j,l} \cdot \tilde{b}'_{j,l}$
return $(\tilde{a}'', \tilde{b}'')$
```

The homomorphic decryption procedure produces LWE ciphertexts of dimension $n = q/2$ equal to the modulus of the packed ciphertext. This is problematic because on subsequent refreshing operations, the resulting error bound will become larger than $q \log q$, and the ciphertexts will no longer be decryptable. Before proceeding, we switch to a smaller modulus which we take to be $2n$ for simple composability. The details of the modulus switching procedure and its associated error bounds are given in Section 2.6.

## 4 Homomorphic Decryption

The homomorphic decryption procedure takes as input a single Ring-LWE ciphertext $(a, b) \in \mathsf{RingLWE}_{\tilde{z}}^{t/2n}[m, \beta] \subseteq (\mathcal{R}_d/2n)^2$ and the encryption of some function of the secret key $\tilde{z} \in \mathcal{R}_d/2n$. The decryption procedure needs to compute the ring element $b - a \cdot \tilde{z} \in \mathcal{R}_d/2n$, and then "round" the result to $\varphi(d)$ LWE ciphertexts. All this should be done homomorphically, given the encryption of (some function of) $\tilde{z}$. We can represent the computation of this ring element as an arithmetic circuit $C$ with inputs in $\mathbb{Z}_{2n}$, but to begin designing such a circuit, we must first consider the cryptographic registers on which we will be computing. We begin this section with a description of the registers used in our ciphertext refreshing procedure, to be followed by a description and analysis of the components of our homomorphic decryption procedure.

### 4.1 Homomorphic Registers

We use a symmetric/ring variant of the GSW cryptosystem to implement the cryptographic registers used by the homomorphic decryption procedure, similar to the accumulators of FHEW. Registers supporting arithmetic modulo $2n$ are implemented using the $\mathsf{GSW}_N^{2n/Q}$ cryptosystem based on the $N$th cyclotomic ring, for $N$ a power of 2 with $2n|N$.

We recall that in FHEW, a value $v \in \mathbb{Z}_{2n}$ is represented by $\mathsf{GSW}_N^{2n/Q}(Y^v)$, where $Y = X^i$ is a primitive $2n$th root of unity. In this scheme we take $N = 2n$, and therefore $X$ is our root of unity. To reduce redundancy given this choice of parameters, we omit the subscript $N$ when referring to $\mathsf{GSW}$ ciphertexts, writing $\mathsf{GSW}^{2n/Q}$. This choice of parameters is more thoroughly justified in Section 4.6, but as the homomorphic decryption procedure will produce $\mathsf{LWE}$ ciphertexts of dimension $n = N/2$, taking $N/2$ to be the original dimension of the $\mathsf{LWE}$ ciphertexts allows us to omit an additional step of key switching to change dimensions.

These $\mathsf{GSW}$ registers support the following operations:

- Initialization ($v \leftarrow w$): $uX^w\mathbf{G}$, with $u \in \mathbb{Z}_Q$, $u \approx Q/2t$, and invertible mod $Q$.
- Increment ($v \leftarrow v + c$): $\mathbf{C} \mapsto \mathbf{C} \cdot X^c$
- Addition: $\mathsf{GSW}^{2n/Q}(uX^v) \times \mathsf{GSW}^{2n/Q}(uX^w) = \mathsf{GSW}_z^{2n/Q}(uX^{v+w})$.
- Extraction: map the accumulator to an LWE ciphertext.

To support subtraction, we represent a value $v \in \mathbb{Z}_{2n}$ as a pair $(\mathsf{GSW}(uX^v), \mathsf{GSW}(uX^{-v}))$. Addition is computed componentwise: $(C_0, C_0') + (C_1, C_1') = (C_0 \times C_1, C_0' \times C_1')$. We implement negation simply by swapping the elements of a pair, and subtraction by combining the two operations. This gives us cryptographic registers supporting all operations required by our refreshing algorithm. To avoid explicitly writing these pairs, we define

$$\mathsf{REG}_z^{2n/Q}(v, \beta) = (\mathsf{GSW}^{2n/Q}(uX^v), \mathsf{GSW}^{2n/Q}(uX^{-v}))$$

and we will further simplify notation in the following section, for the purposes of detailing our algorithm.

## 4.2 Slow Multiplication

The use of the $\mathsf{REG}$ scheme restricts our arithmetic circuits to use only the operations described above. Given the asymmetric error growth of the underlying $\mathsf{GSW}$ operations, we must also be careful in how we design our circuit, as we don't want both inputs to any addition or subtraction gate to have already accumulated significant error. For the rest of this section, however, we omit explicit references to the $\mathsf{REG}$ scheme wherever possible, to simplify the presentation of the homomorphic decryption algorithm. We instead use the notation $[\![c]\!]$ to denote a register or registers encrypting value(s) $c$.

Our goal, then, is to specify an efficient circuit which is parameterized by the input ciphertext $(a, b)$, meets our restrictions, and outputs an encryption of the desired ring element $[\![b - a \cdot \tilde{z}]\!]$ that will then allow each coefficient to be homomorphically rounded to an $\mathsf{LWE}$ ciphertext. We discuss the rounding procedure in Section 4.5. In the next few sections, we specify an arithmetic circuit computing $[\![b - a \cdot \tilde{z}]\!]$, where this circuit takes as input a function of the secret key, $[\![f(\tilde{z})]\!]$, and outputs $C_{a,b}([\![f(\tilde{z})]\!]) = [\![b - a \cdot \tilde{z}]\!]$. We will start with a comparatively straightforward, but inefficient, such construction in which we compute $a \cdot \tilde{z}$ homomorphically with a slow multiplication algorithm.

Let $l = \log n + 1$ and $f(\tilde{z}) \in \mathbb{Z}_{2n}^{l \times \varphi(d)}$ be defined by $f(\tilde{z})_{j,k} = \tilde{z}_k 2^j$ Let $a_{i,j}$ be the $j$th bit of the binary decomposition of $a_i$ so that $a_i = \sum_{j=0}^{l-1} a_{i,j} 2^j$. We may express multiplication of $\tilde{z}_k$ by $a_i$ by computing $\tilde{z}_k \cdot a_i = \sum_{j=0}^{l-1} a_{i,j} \tilde{z}_k 2^j$. Then we define

$$C_{a_i}(\llbracket f(\tilde{z}) \rrbracket, k) = \sum_{j=0}^{l-1} a_{i,j} \llbracket f(\tilde{z})_{j,k} \rrbracket$$

to be a circuit computing this multiplication homomorphically using only additions, as the $a_{i,j}$ values are binary. We may then define a circuit $C_a$, computing a slow multiplication algorithm (mod $\Phi_d$) using these $C_{a_i}$ subcircuits, addition, and subtraction gates.

**Lemma 4** *Let $B$ be the base of the geometric progression defining $\mathbf{g}$ in $\mathsf{GSW}$ encryption, and let $d_b = \lceil \log_B Q \rceil$. There is an algorithm*

$$\mathsf{SlowMult} : \mathcal{R}_d/2n \times \mathsf{REG}_z^{l \times \varphi(d)} \to \mathsf{REG}_z^{\varphi(d)}$$
$$(a, (\mathsf{REG}_z(f(\tilde{z})_{j,k}, \beta_R))_{j,k}) \mapsto (\mathsf{REG}_z((a \cdot \tilde{z})_i, \beta'))_i$$

*where $\beta' \in \tilde{O}(\beta_R B \sqrt{n d_B d})$ with overwhelming probability, and requiring $\tilde{O}(d^2)$ homomorphic additions and subtractions.*

Proof. Multiplication by a constant as described above requires at most $l$ homomorphic additions. We may compute each of the $\varphi(d)$ coefficients of $a \cdot \tilde{z}$ using at most $d$ constant multiplications for any coefficient, and therefore $\mathsf{SlowMult}$ requires $O(d^2 \cdot l)$ homomorphic operations.

To bound the error growth of the algorithm, we use an analysis similar to that of Lemma 11 in Ducas and Micciancio [DM15]. We refer to $\mathsf{GSW}$ ciphertexts for convenience in the analysis, rather than $\mathsf{REG}$, as the error growth will be identical for both components of the register. Let $\mathbf{C}^{(i)}$ denote the $\mathsf{GSW}$ ciphertext storing the result of the first $i$ additions, and set $\mathbf{D}^{(i)} = [D_1, ..., D_{d_B}]$, the base $B$ decomposition of $\mathbf{C}^i$.

The final sum $\mathbf{C}$ after $d \cdot l$ additions can then be written $\mathbf{C} = [\mathbf{A}, \mathbf{A} \cdot \tilde{z} + \mathbf{e} + uX^v\mathbf{G}]$ where

$$\mathbf{e} = \left[\mathbf{D}^{(0)}, \mathbf{D}^{(1)}, ..., \mathbf{D}^{(d \cdot l-1)}\right] \cdot \left(X^{(v_1)}\mathbf{e}^{(1)}, X^{(v_2)}\mathbf{e}^{(2)}, ..., X^{(v_{d \cdot l-1})}\mathbf{e}^{(d \cdot l-1)}\right),$$

where $\mathbf{e}^{(i)}$ is the error of a fresh $\mathsf{GSW}$ ciphertext (given as input) and $v_i$ is an integer. Fact 12 of [DM15] gives us that

$$s_1\left(\left[\mathbf{D}^{(0)}, \mathbf{D}^{(1)}, ..., \mathbf{D}^{(d \cdot l-1)}\right]\right) \in O(B\sqrt{n d_B d \cdot l}).$$

As the $\mathsf{GSW}$ ciphertexts to be summed are fresh encryptions with noise bound $\beta_R$, the final error of any register output by $\mathsf{SlowMult}$ is bounded by

$$\beta' = \tilde{O}(\beta_R B \sqrt{n d_B d}).$$

□

The original FHEW refreshing procedure requires only $\tilde{O}(n)$ homomorphic additions per ciphertext, so we already see this algorithm offers no improvement over sequential refreshing of $d$ ciphertexts using FHEW. We will instead use variants of existing fast multiplication algorithms for the homomorphic computation of $a \cdot \tilde{z}$, using $\mathsf{SlowMult}$ as a subroutine.

## 4.3 Homomorphic DFT

We briefly recall and introduce notation for the discrete Fourier transform. We denote the discrete Fourier transform of a length $m$ sequence of elements $x \in \mathcal{R}^m$, where ring $\mathcal{R}$ has $m$th principal root of unity $\omega_m$, by $\bar{x}_i = DFT(x)_i = \sum_{k=0}^{m-1} x_k \omega_m^{ik}$ and its inverse $x_k = DFT^{-1}(\bar{x})_k = \sum_{i=0}^{m-1} \bar{x}_i \omega_m^{-ik}$. The polynomial product $a \cdot \tilde{z}$ for $a, \tilde{z} \in \mathcal{R}_d/2n$ may then be computed as

$$(a * \tilde{z} \mod X^m - 1) \mod \Phi_d = DFT^{-1}(\frac{1}{m} DFT(a) \cdot DFT(\tilde{z})) \mod \Phi_d$$

provided an $m$th principal root of unity $\omega_m$ exists in $\mathcal{R}_d/2n$ (and $m > \frac{4}{3}d \geq deg(a \cdot \tilde{z})$).

But to compute the DFT homomorphically, we need to be able to homomorphically compute multiplication by $\omega_m^{-ik}$. If we take $\omega_m \in \mathbb{Z}_{2n}$, each multiplication by $\omega_m^{-ik}$ requires $l$ homomorphic operations per coefficient (as described in Section 4.2). Furthermore, reducing the quadratic complexity of the DFT requires FFT techniques, recursing to depth $\log m$. At each step of recursion, then, we perform homomorphic operations on registers produced from the last step, with some increase in error from the previous set of operations. We therefore cannot take advantage of the asymmetric error growth of the GSW scheme underlying our registers and the error growth of our algorithm will become quasi-polynomial in $n$, exceeding the desired polynomial bound on error overhead. This brings us to the last component of the algorithm, which avoids these scalar multiplications, and achieves efficient multiplication without sacrificing polynomial error growth.

## 4.4 Nussbaumer Transform

In order to efficiently compute the polynomial product $a \cdot \tilde{z} \in \mathcal{R}_d/2n$, we define a variation of the Nussbaumer polynomial transform for negacyclic convolution, suited for multiplication of polynomials modulo a power of 3 cyclotomic. Informally, the transform first maps an element $a \in \mathcal{R}_d/2n$ to a bivariate polynomial in such a way that it may be represented by a coefficient vector of length less than $\varphi(d)$, over $\mathcal{R}_{d^{1-\epsilon}}/2n$. Taking the DFT of this coefficient vector allows us to reduce the computation of $a \cdot \tilde{z}$ to the pointwise multiplication of two vectors with entries in $\mathcal{R}_{d^{1-\epsilon}}/2n$. The resulting vector of smaller polynomial products is then recombined via the inverse DFT and inverse map to yield $a \cdot \tilde{z} \in \mathcal{R}_d/2n$. We now give a more detailed description of the algorithm.

Let $d = 3^k$, $m = d^\epsilon$ with $d \geq 3m^2$, and $r = \varphi(d)/m = \varphi(d^{1-\epsilon})$. To multiply two polynomials, the transform maps each polynomial to a bivariate polynomial by the isomorphism

$$\psi : \mathbb{Z}_{2n}[X]/(\Phi_d) \to (\mathbb{Z}_{2n}[Y]/(\Phi_{d/m}(Y)))[X]/(X^m - Y)$$

$$a(X) = \sum_{i=0}^{\varphi(d)} a_i X^i \mapsto \sum_{j=0}^{m-1} \sum_{i=0}^{r-1} a_{mi+j} Y^i X^j \text{ where } Y = X^m.$$

Because $\psi(a)$ and $\psi(\tilde{z})$ have degree at most $m-1$ in $X$, computing $\psi(a) \cdot \psi(b)$ modulo any polynomial of degree greater than $2m-2$ in $X$ prior to reducing by $X^m - Y$ will not change the result. We also note that $Y$ is a principal $d/m$th root of unity in $\mathbb{Z}_{2n}[Y]/(\Phi_{d/m}(Y))$, and therefore $Y^{d/3m^2} = Y^{3^{k(1-2\epsilon)-1}}$ is a $3m$th root which can also be shown to be principal.

This allows us to efficiently compute $\psi(a) \cdot \psi(\tilde{z})$ first modulo $X^{3m} - 1$ by pointwise multiplication of the respective DFTs, followed by a reduction modulo $(X^m - Y)$. Since the "points" of the DFT pointwise multiplication step are elements of $\mathbb{Z}_{2n}[Y]/(\Phi_{d/m}(Y))$, these multiplications can be performed by recursive application of the transform or SlowMult.

Without recursion, this gives a key preprocessing function

$$f(\tilde{z}) = (1, 2, 4, ..., n) \otimes [\frac{1}{3m} DFT(\psi(\tilde{z}))_i]_{i<3m} \in (\mathcal{R}_{d/m}/2n)^{l \times 3m} \tag{11}$$

where the DFT evaluates $\psi(\tilde{z})$ at root of unity $\omega_{3m} = Y^{d/3m^2}$.

Let $\bar{a}_i = DFT(\psi(a))_i$ be the $i$th of the $3m$ degree $r$ polynomials produced by the Nussbaumer transform. Let $C_{\bar{a}_i}$ denote a circuit computing SlowMult of known polynomial $\bar{a}_i$ (of degree $< r$) with an encrypted polynomial given as input (along with encryptions of all power of 2 multiples of the polynomial's coefficients, as in Section 4.2). Let $C_{F^*}$ be a circuit homomorphically computing the inverse DFT for length $3m$ vectors of encrypted polynomials in $\mathcal{R}_{d/m}/2n$. Then we may specify a circuit computing $[\![a \cdot \tilde{z}]\!]$

$$C_a([\![f(\tilde{z})]\!] = C_{F^*}([C_{\bar{a}_i}([\![f(\tilde{z})_{(\cdot),i}]\!])]_i) \mod X^m - Y = [\![a \cdot \tilde{z}]\!].$$

$[\![b - a \cdot \tilde{z}]\!]$ is then computed by negating each of the registers $\mathsf{REG}((a \cdot \tilde{z})_i)$ and incrementing each one by the corresponding $b_i$, computing $C_{a,b}([\![f(\tilde{z})]\!]) = -C_a([\![f(\tilde{z})]\!]) + b$.

The map $\psi$ is purely representational, so requires no computation. The forward and inverse DFT steps require evaluating polynomials at the roots of unity $\omega_{3m}^i = Y^{id/3m^2}$, which is implemented by rotation of the coefficient vectors with negation, addition, and subtraction to implement the reduction in $\mathcal{R}_{d/m}/2n$. SlowMult was defined using only the operations of addition and subtraction, and the final reduction modulo $X^m - Y$ similarly only requires additions and subtractions. This circuit then satisfies our criteria, allowing for the homomorphic computation of $a \cdot \tilde{z}$ without use of multiplication gates. Algorithm 4 gives a summary of the RingDecrypt procedure.

The Nussbaumer transform admits a recursive algorithm that gives tradeoffs between runtime and error growth of the cryptographic registers. For an initial analysis, and to demonstrate the core idea of our refreshing algorithm, we state Lemma 5 considering the simplest form of the RingDecrypt procedure, using no recursion. We defer consideration of the recursive formulation to Section 4.7.

**Lemma 5** *Let $\mathcal{K}^R$ be a refreshing key*

$$\mathcal{K}^R = K^R_{i,j,k} = \mathsf{REG}_z^{2n/Q}([f(\tilde{z})_{i,j,k}]) = \mathsf{REG}_z^{2n/Q}\left[\frac{1}{3m} DFT(\psi(\tilde{z}))_{i,k} 2^j, \beta_R\right] \textit{(from Eq. 11)}$$

*where $DFT(\psi(\tilde{z}))_{i,k}$ indicates the $k$th coefficient of the $i$th polynomial output by the Nussbaumer transform (giving $i < 3m, j < l, k < r$). For every $0 < \epsilon < \frac{1}{2}$, there is an algorithm* RingDecrypt *that on input $\mathcal{K}^R$ and* RingLWE *ciphertext $(a, b) \in \mathsf{RingLWE}_{\tilde{z}}^{2/2n}[m, \beta]$ under $\tilde{z} \in \mathcal{R}_d/2n$, outputs $\varphi(d)$ ciphertexts $\left[\mathsf{REG}_z^{2n/Q}(\tilde{m}_i, \beta')\right]_{i<\varphi(d)}$ with $\beta' \in \tilde{O}\left(\beta_R B^4 (nd_B)^{3.5} d^{\epsilon+.5}\right)$, and requiring $\tilde{O}(d^{2-\epsilon})$ homomorphic operations.*

Proof. Let RingDecrypt be as specified in Algorithm 4. We are given $(a, b)$ in the clear, and so we omit the contribution to the complexity of the algorithm from the Nussbaumer transform of $a$, as this will be dominated by the inverse transform that must be computed homomorphically.

The SlowMult subroutine is used to multiply the $3m$ pairs of (degree less than $r$) polynomials produced by the Nussbaumer transform. From Lemma 4, each multiplication requires $O(r^2 \cdot l)$ homomorphic operations, so that SlowMult contributes $O(3mr^2 \cdot l)$ homomorphic operations to the algorithm and increases the error by a factor of $O\left(B\sqrt{nd_B d \cdot l}\right)$.

**Algorithm 4** $\mathsf{RingDecrypt}((a,b),\mathcal{K}^R)$

```
Input: $(a,b)$                                              ▶ $(a,b) \in \mathsf{RingLWE}_{\tilde{z}}^{2/2n}[m(X),\beta]$
       $\mathcal{K}^R = K^R_{i,j,k}$                          ▶ $K^R_{i,j,k} \in \mathsf{REG}_z^{2n/Q}[\frac{1}{3m}DFT(\psi(\tilde{z}))_{i,k}2^j, \beta_R]$
Output: $[[\![c_i]\!]]_{i<\varphi(d)}$                          ▶ $[\![c_i]\!] \in \mathsf{REG}_z^{2n/Q}[(b - a\cdot\tilde{z})_i, \beta']$
$\hat{a}_i \leftarrow [\psi(a)]_i$                            ▶ $\hat{a}_i \in \mathcal{R}_{d/m}/2n$
$\bar{a}_i \leftarrow DFT(\psi(a))_i$                         ▶ $\bar{a}_i \in \mathcal{R}_{d/m}2n$
for $i \in \{0,...,3m\}$ do
    $\bar{\mathbf{C}}_i \leftarrow \mathsf{SlowMult}(\bar{a}_i, \mathcal{K}^R_i)$          ▶ $\bar{\mathbf{C}}_i \in \left[\mathsf{REG}_z^{2n/Q}[DFT(\psi(a\cdot\tilde{z}))_i]_j\right]_{j<r}$
$[\hat{\mathbf{C}}_k]_{k<3m} \leftarrow DFT^{-1}([\bar{\mathbf{C}}_i]_{i<3m})$       ▶ $\hat{\mathbf{C}}_k \in \left[\mathsf{REG}_z^{2n/Q}[\psi(a\cdot\tilde{z})_k]_j\right]_{j<r}$
for $k \in \{m,...,2m-1\}$ do {reduction modulo $X^m - Y$}
    $\hat{\mathbf{C}}_k \leftarrow \hat{\mathbf{C}}_k Y \mod \Phi_{d/m}$
for $k \in \{2m,...,3m-1\}$ do
    $\hat{\mathbf{C}}_k \leftarrow \hat{\mathbf{C}}_k Y^2 \mod \Phi_{d/m}$
for $i \in \{0,...,m-1\}$ do
    for $j \in \{0,...,r-1\}$ do
        $[\mathbf{C}_{mj+i}] \leftarrow (\mathbf{C}_{i,j} + \mathbf{C}_{m+i,j} + \mathbf{C}_{2m+i,j})$
                                                             ▶ $\mathbf{C}_{mj+i} \in \mathsf{REG}_z^{2n/Q}[(a\cdot\tilde{z})_{mj+i}]$
$\mathbf{C}_i \leftarrow -\mathbf{C}_i \cdot X^{b_i}$                          ▶ $\mathbf{C}_i \in \mathsf{REG}_z^{2n/Q}[\tilde{m}_i, \beta' \in \tilde{O}\left(\beta_R B^4 (nd_B)^{3.5} d^{.5+\epsilon}\right)]$
return $[\mathbf{C}_i]_{i<\varphi(d)}$
```

The inverse DFT step of the inverse Nussbaumer transform requires the evaluation of $3m$ degree $r$ polynomials at powers of the root of unity $Y^{d/3m^2}$. These evaluations are taken modulo $\Phi_{d/m}(Y) = Y^r + Y^{r/2} + 1$, and therefore multiplications by $Y^i$ amount to rotations

$$aY^v \mapsto aY^{v+i \mod d/m}, \text{ for } v+i<r \text{ and } v+i>d/m$$

with negation and subtraction of the overflow terms

$$aY^v \mapsto -aY^{v+i \mod r} - aY^{v+i+\frac{r}{2} \mod r} \text{ when } r \le v+i < d/m.$$

Then we see that for a polynomial $p(Y) \in \mathcal{R}_{d/m}/2n$, we will only need to compute $p_v - p_{v+\frac{r}{2}}$ for $0 \le v < r/2$ to produce all values required for the coefficients of $p(Y)\cdot Y^i$ for any $i$, and therefore these evaluations require $O(mr)$ homomorphic operations.

The noise growth for this step may be analyzed as in Lemma 4, but without the independence assumption. Let

$$A = \mathsf{GSW}_z^{2n/Q}[a,\beta] = [\mathbf{A}, \mathbf{A}\cdot z + \mathbf{e} + uX^a\mathbf{G}]$$
$$\text{and } B = \mathsf{GSW}_z^{2n/Q}[b,\beta] = [\mathbf{B}, \mathbf{B}\cdot z + \mathbf{e}' + uX^b\mathbf{G}]$$

be the relevant register entries. Then $A + B = [\mathbf{C}, \mathbf{C}\cdot z + \mathbf{e}'' + uX^{a+b}\mathbf{G}]$ will have error

$$\mathbf{e}'' = [\mathbf{A}_1, ..., \mathbf{A}_{d_B}] \cdot X^b\mathbf{e}', \text{ where } s_1([\mathbf{A}_1,...,\mathbf{A}_{d_B}]) \in O(d_B B n).$$

Each of the $3m$ components of the inverse DFT are then computed as described in Section 4.3, with $3m$ additions of polynomials with degree less than $r$. These summations require a total of $O((3m)^2 r)$ homomorphic operations and result in registers with error bounded by $O(\beta_R d_B m B n)$.

The final reduction modulo $X^m - Y$ as described in Algorithm 4 requires a number of additions and subtractions linear in $\varphi(d)$. Each register, however, will only be involved in constant number of homomorphic operations. Following the analysis above, this gives an additional error growth factor of $O(d_B B n)$.

The final complexity of the RingDecrypt algorithm then, in terms of homomorphic operations, is

$$O((3m)^2 r + 3mr^2 l) = \tilde{O}(d^{1+\epsilon} + d^{2-\epsilon})$$

and the algorithm outputs registers with error $\tilde{O}\left(\beta_R B^4 (n d_B)^{3.5} d^{\epsilon+.5}\right)$. Recall that in order to have the necessary $3m$th root of unity, we need $d \geq 3m^2 = 3^{2k\epsilon+1}$. Therefore $\epsilon \leq \frac{1}{2} - \frac{1}{2k}$, forcing the multiplication step to dominate the DFT step with complexity $\tilde{O}(d^{2-\epsilon})$.

□

Once we have performed the RingDecrypt procedure, it remains to round the resulting vector of ciphertexts to yield $\varphi(d)$ refreshed LWE ciphertexts.

## 4.5 msbExtract

Once we have computed the sequence of registers $[\mathsf{REG}_z^{2n/Q}(b_i - (a \cdot \tilde{z})_i)]_{i<\varphi(d)}$, we must homomorphically perform the rounding step of decryption and recover a sequence of reduced-error LWE ciphertexts encrypting each $m_i$. This can be accomplished as in FHEW, by applying the msbExtract procedure described in Algorithm 5 to each of the $\varphi(d)$ registers produced by RingDecrypt.

We note that our msbExtract procedure differs somewhat from that of FHEW, in that we omit the step of key switching. In FHEW, the procedure takes as additional input a switching key. This key enables the LWE encryptions under key $\boldsymbol{(}z\boldsymbol{)}$ that are recovered at an intermediate stage of the rounding algorithm to be converted to LWE encryptions under the initial key $\mathbf{s}$. Choosing our REG key $z$ such that $\boldsymbol{(}z\boldsymbol{)} = \mathbf{s}$ obviates the need for key switching, as the intermediate LWE ciphertexts will already be encrypted under the proper key. This choice of key requires assuming the security of RingLWE with binary secrets, as assumed in [CGGI16], but this assumption may be removed at the cost of the additional key switching step.

**Lemma 6** *Let $\tilde{m} = m_i n + \tilde{e} = b_i - (a \cdot \tilde{z})_i$ be the ith coefficient of a noisy* RingLWE *message. There is an algorithm* msbExtract *that, given a cryptographic register of the form* $\mathsf{REG}_z^{2n/Q}(\tilde{m}, \beta)$ *as input, with $\boldsymbol{(}z\boldsymbol{)} = \mathbf{s}$, outputs a LWE ciphertext* $\mathsf{LWE}_{\mathbf{s}}^{4/Q}(m_i, \sqrt{n}\beta)$.

Proof. Let $\mathbf{C} = \mathsf{GSW}_z^{2n/Q}(\tilde{m}, \beta)$ be the positive component of the REG input. Then the second row of $\mathbf{C} \cdot X^{n/2}$ is a $\mathsf{RingLWE}_z^{2n/Q}$ encryption

$$[a_1, b_1] = [a_1, a_1 \cdot z + e_1 + uX^{\tilde{m}+n/2}].$$

Noting that

$$[\boldsymbol{(}a_1\boldsymbol{)}, \boldsymbol{(}b_1\boldsymbol{)}] = [\boldsymbol{(}a_1\boldsymbol{)}, \mathbf{M}_{a_1} \cdot \boldsymbol{(}z\boldsymbol{)} + \boldsymbol{(}e_1\boldsymbol{)} + u\boldsymbol{(}X^{\tilde{m}+n/2}\boldsymbol{)}]$$

and our target is a ciphertext of the form $[\mathbf{a}, \mathbf{a}\cdot\mathbf{s} + e + \frac{Q}{t}m_i]$, we let $\mathbf{t} = [-1, -1, ..., -1] \in \mathbb{Z}_Q^{2n}$ and compute

$$[\mathbf{a}^t, b'] = \mathbf{t}\cdot[\mathbf{M}_{\tilde{a}}, (b_1)] = \mathbf{t}\cdot[\mathbf{M}_{\tilde{a}}, \mathbf{M}_{\tilde{a}}\cdot(z) + (e_1) + u(X)^{\tilde{m}+n/2}].$$

The term $\mathbf{t}\cdot u(X)^{\tilde{m}+n/2} = -u$ for $-n/2 < m_i n + e < n/2$ and $u$ otherwise, so

$$[\mathbf{a}, b] = [\mathbf{0}, u] + [\mathbf{a}, b'] = [\mathbf{a}, \mathbf{a}\cdot\mathbf{s} + \mathbf{t}\cdot(e_1) + 2um_i]$$

when the encryption error $|\tilde{e}| < \frac{n}{2}$. We took $u \approx Q/2t$, and so $b = \mathbf{a}\cdot\mathbf{s} + e' + \frac{Q}{t}m_i$, for $e' \le \beta\|\mathbf{t}\| = \beta\sqrt{n}$, as desired. □

**Algorithm 5** msbExtract($\mathbf{R}$)

```
Input: $\mathbf{R}$                                           ▶ $\mathbf{R} \in \mathsf{REG}_z^{2n/Q}[\tilde{m}]$
Output: $(\mathbf{a}, b)$                                     ▶ $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{4/Q}[m_i]$
$\mathbf{C} \leftarrow \mathbf{R}_0$                          ▶ $\mathbf{C} \in \mathsf{GSW}_z^{2n/Q}[\tilde{m}]$
$[a_1, b_1] \leftarrow [0, 1, 0, ..., 0]\cdot\mathbf{C}X^{n/2}$   ▶ $(a_1, b_1) \in \mathsf{RingLWE}_z^{2n/Q}[uX^{\tilde{m}+n/2}]$
$[\mathbf{a}^t, b] \leftarrow [\mathbf{0}^t, u] + [-1, -1, ..., -1]\cdot[\mathbf{M}_{\tilde{a}}, (b_1)]$
                                                              ▶ $(\mathbf{a}, b) \in \mathsf{LWE}_{(z)}^{4/Q}[m_i, \sqrt{n}\beta]$
return $(\mathbf{a}, b)$
```

## 4.6 Refreshing Algorithm

We now present the amortized bootstrapping algorithm from start to finish and give an analysis of its runtime and error growth.

The algorithm takes as input $\varphi(d)$ ciphertexts for $d = 3^k$, under the same key $s \in \mathbb{Z}_q^n$ and with error at most $\Delta$, to be simultaneously refreshed. It also requires key material for the PackLWE and RingDecrypt procedures, described in their respective sections, but recalled here for reference. The packing key $\mathcal{K}^P = K_{j,l}^P$ is required to pack the LWE ciphertexts into a single RingLWE ciphertext under a new key $\tilde{z} \in \mathcal{R}_d/q$, and is given by

$$K_{j,l}^P = \mathsf{RingLWE}_{\tilde{z}}^{q/q}(s_l 2^j, \beta_P).$$

The refreshing key $\mathcal{K}^R = K_{i,j,k}^R$ encrypts a function of the RingLWE secret $f(\tilde{z})$ under a REG key $z \in \mathcal{R}_{2n}/Q$, and is required for the homomorphic decryption of the resulting RLWE ciphertext. Its entries are given by $K_{i,j,k}^R = \mathsf{REG}_z^{2n/Q}\left(\frac{1}{3m}DFT(\psi(\tilde{z}))_{i,k})2^j, \beta_R\right)$.

**Theorem 2** *For every $0 < \epsilon < \frac{1}{2}$, there exists an algorithm* Refresh *that, on the input described above, produces $\varphi(d)$* LWE *ciphertexts with error $\tilde{O}\left(\|s\| + \frac{q}{Q}\cdot\beta_R(Bn)^4 d_B^{3.5} d^{\epsilon+.5}\right)$, and requires $\tilde{O}(d^{2-\epsilon})$ homomorphic operations.*

Proof. From Lemma 5, we already have that the homomorphic complexity of RingDecrypt is $\tilde{O}(d^{2-\epsilon})$, and this dominates the complexity of RingDecrypt.

The correctness of this refreshing scheme will rely on the error bounds at two stages of the algorithm. For msbExtract to recover the correct $m_i$'s from the output of RingDecrypt, the error of each $\tilde{m}_i = \frac{2n}{t} m_i + e$ must not exceed $\frac{n}{t}$ (for $t = 2$, the message space $m_i$).

From the error bounds of Theorem 3 and Lemma 2, we have that the ciphertext output by $\mathsf{ModSwitch}_{\mathcal{R}}$ will have error bounded by $O\left(\frac{2n}{q}(\sqrt{d}\Delta + \sqrt{dn \log q}\beta_P) + \omega(\sqrt{d \log d}) \cdot \|\tilde{z}\|\right)$. To bound this by $n/2$, we restrict $\tilde{z}$ to $\|\tilde{z}\| = O(\sqrt{d})$ . Then so long as the LWE ciphertext error $\Delta$ satisfies $\Delta < O(\frac{q}{\sqrt{d}})$, $d\sqrt{\log d} < O(n)$, and $d^2 n < O(\frac{q^2}{\log q})$, the packed ciphertexts will be decryptable with high probability.

We also need to guarantee that the ciphertexts output by Refresh will have error small enough that this scheme is composable. From the bound above, this requires us to bound the error of the ciphertexts output by Refresh by $\beta' = \tilde{O}\left(\|s\| + \frac{q}{Q} \cdot \beta_R (nB)^4 d_B^{3.5} d^{\epsilon+.5}\right) < \frac{q}{\sqrt{d}}$. Letting $s$ be a binary secret, (which does not reduce hardness of the associated LWE instance, as shown in [BLP+13]), this gives us that $\frac{Q}{(d_B)^{3.5}} > \beta_R n^4 \sqrt{\log n} B^4 d^{1+\epsilon}$, so taking $B = \Theta(1)$ and $Q > \tilde{O}(\beta_R n^4 d^{1+\epsilon} (\log d)^{3.5})$ will guarantee correct decryption with high probability.

□

**Algorithm 6** $\mathsf{Refresh}([(a_i, b_i)]_{i<\varphi(d)})$

```
**Input:** $[(a_i, b_i)]_{i<\varphi(d)}$                                  ▶ $(a_i, b_i) \in \mathsf{LWE}_{\mathbf{s}}^{2/q}[m_i]$
**Output:** $[(a_i, b_i)]_{i<\varphi(d)}$                                 ▶ $(a_i, b_i) \in \mathsf{LWE}_{\mathbf{s}}^{4/q}[m_i]$
$(a^{(1)}, b^{(1)}) \leftarrow \mathsf{PackLWE}([(a_i, b_i)]_{i<\varphi(d)})$              ▶ $(a^{(1)}, b^{(1)}) \in \mathsf{RingLWE}_{\tilde{z}}^{2/q}[m(X)]$
$(a^{(2)}, b^{(2)}) \leftarrow \mathcal{R}\text{-}\mathsf{ModSwitch}_R(a^{(1)}, b^{(1)})$                ▶ $(a^{(2)}, b^{(2)}) \in \mathsf{RingLWE}_{\tilde{z}}^{2/2n}[m(X)]$
$[\mathbf{C}_i]_{i<\varphi(d)} \leftarrow \mathsf{RingDecrypt}(a^{(2)}, b^{(2)})$                  ▶ $\mathbf{C}_i \in \mathsf{REG}_z^{2n/Q}[\tilde{m}_i]$
$[(a_i^{(3)}, b_i^{(3)})]_{i<\varphi(d)} \leftarrow [\mathsf{msbExtract}(\mathbf{C}_i)]_{i<\varphi(d)}$      ▶ $(a_i^{(3)}, b_i^{(3)}) \in \mathsf{LWE}_{\mathbf{s}}^{4/Q}[m_i]$
$[(a_i, b_i)]_{i<\varphi(d)} \leftarrow [\mathsf{ModSwitch}(a_i^{(3)}, b_i^{(3)})]_{i<\varphi(d)}$      ▶ $(a_i, b_i) \in \mathsf{LWE}_{\mathbf{s}}^{4/q}[m_i]$
**return** $[(a_i, b_i)]_{i<\varphi(d)}$
```

## 4.7 Recursive optimization

In this section we show that a recursive formulation of the Nussbaumer transform can improve the complexity of the RingDecrypt algorithm, at the cost of an increase in the error.

The Nussbaumer transform as described in Section 4.4 can be thought of as a reduction from a single multiplication of two polynomials in $\mathbb{Z}_{2n}[X]/(\Phi_d(X))$ to $3m$ multiplications of pairs of polynomials in $\mathbb{Z}_{2n}[Y]/(\Phi_{d/m}(Y))$. We may recursively apply this transformation $\rho$ times, provided we have a $3m$th root of unity in the ring $\mathbb{Z}_{2n}[Y]/(\Phi_{d/m^\rho}(Y))$, which will be the case as long as $d/m^\rho \geq 3m$. $\epsilon$ is fixed and so this bounds the recursive depth of the algorithm by $\rho < \frac{1}{\epsilon} - 1$.

**Theorem 3** *The recursive* $\mathsf{RingDecrypt}_\rho$ *algorithm, with constant parameter* $\epsilon$ *and recursive depth* $\rho < \frac{1}{\epsilon} - 2$*, requires* $\tilde{O}(3^\rho d^{2-\rho\epsilon} + 3^\rho d^{1+\epsilon})$ *homomorphic operations and yields an error growth of* $\tilde{O}(B^{3\rho+1}(nd_B)^{3\rho}\sqrt{nd_B d^{1+\rho\epsilon}})$.

Proof. Because the transform reduces a single multiplication of degree (at most) $\varphi(d)$ polynomials to $3m$ multiplications of degree (at most) $\varphi(d)/m$ polynomials, after $\rho$ recursive calls we will still need to compute $(3m)^\rho$ multiplications of degree $\varphi(d)/m^\rho$. We do this with the SlowMult algorithm as before. From Lemma 4, the complexity of one of these multiplications, in terms of register operations, will be $\tilde{O}((\frac{d}{m^\rho})^2)$. Therefore the contribution from the multiplications to the complexity of the recursive algorithm will be $\tilde{O}((3m)^\rho(\frac{d}{m^\rho})^2) = \tilde{O}(3^\rho d^{2-\rho\epsilon})$. When we take $\rho = \frac{1}{\epsilon} - 2$, the complexity will be $\tilde{O}(3^{1/\epsilon}d^{1+2\epsilon})$.

Each recursive application also requires a DFT of the polynomials produced by the map $\psi$. Keeping $\epsilon$ constant at each step fixes the dimension of each DFT to $3m = 3d^\epsilon$. After $\rho$ applications of the transform we have reduced our polynomials to elements of $\mathbb{Z}_{2n}[Y]/(\Phi_{d/m^\rho}(Y))$. From the analysis of Lemma 5, we then have that a DFT at depth $\rho$ will require $O((3d^\epsilon)^2 d^{1-\rho\epsilon})$ operations. There will be $(3d^\epsilon)^{\rho-1}$ such DFTs at depth $\rho$, so the contribution of the DFTs to the complexity will be

$$O\left(\sum_{i=1}^{\rho}(3d^\epsilon)^{i-1}(3d^\epsilon)^2 d^{1-i\epsilon}\right) = O\left(d^{1+\epsilon}\sum_{i=1}^{\rho}3^i\right) = O(d^{1+\epsilon}3^\rho).$$

The total homomorphic operation complexity of the recursive formulation, for fixed $\epsilon$ and depth $\rho$ is then $\tilde{O}\left(3^\rho d^{2-\rho\epsilon} + d^{1+\epsilon}3^\rho\right)$. Considering again the case where we recurse to the maximum depth, fixing $\rho = \frac{1}{\epsilon} - 2$, we have a complexity of $\tilde{O}(3^{1/\epsilon}d^{1+2\epsilon} + d^{1+\epsilon}3^{1/\epsilon}) = \tilde{O}(3^{1/\epsilon}d^{1+2\epsilon})$.

We now consider the effect of recursion on the error growth in the RingDecrypt algorithm. From Lemma 4, we have that the error grows by a factor $\tilde{O}(B\sqrt{nd_B d})$ for degree $\varphi(d)$ polynomial multiplication. As the degree of the polynomials requiring SlowMult multiplication is reduced to $\varphi(d^{1-\rho\epsilon})$ in the recursive version of the algorithm, the error growth will similarly be reduced to $\tilde{O}(B\sqrt{nd_b d^{1-\rho\epsilon}})$.

Lemma 5 gave a bound on error following the DFT step of $O(\beta(Bd_b n)^3 d^\epsilon)$, when the inputs are registers of error $\beta$. The error growth compounds with each inverse DFT so that the total error contribution from all DFTs can be bounded by $O\left(\beta_R((Bd_b n)^3 d^\epsilon)^\rho\right)$, increasing exponentially in the recursive depth.

Combining the error bounds for the multiplication and DFT steps of $\mathsf{RingDecrypt}_\rho$ gives a final error bound of $\tilde{O}\left(B^{3\rho+1}(nd_B)^{3\rho}\sqrt{nd_B d^{1+\rho\epsilon}}\right)$.

□

**Acknowledgements**

The first author would like to thank Paul Kirchner for useful conversations during his 2015 visit. Both authors are grateful to the anonymous reviewers for helpful comments.

## References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.

[AP13] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of

*LNCS*, pages 1–20, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[BDF18] Guillaume Bonnoron, Léo Ducas, and Max Fillinger. Large FHE gates from tensored homomorphic accumulator. In *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*, volume 10831 of *Lecture Notes in Computer Science*, pages 217–251. Springer, 2018.

[BGH13] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 1–13, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. ACM.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BR15] Jean-François Biasse and Luis Ruiz. FHEW with efficient multibit bootstrapping. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 119–135, Guadalajara, Mexico, August 23–26, 2015. Springer, Heidelberg, Germany.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[BV14a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, 43(2):831–871, 2014.

[BV14b] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014*, pages 1–12, Princeton, NJ, USA, January 12–14, 2014. ACM.

[CGGI16] 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, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[CZ17] Long Chen and Zhenfeng Zhang. Bootstrapping fully homomorphic encryption with ring plaintexts within polynomial noise. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *Provable Security*, pages 285–304, Cham, 2017. Springer International Publishing.

[DM14] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 335–352, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[GH11] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 107–109, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[GHPS12] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Ring switching in BGV-style homomorphic encryption. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 19–37, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg, Germany.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 1–16, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[GHS12c] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[GINX16] Nicolas Gama, Malika Izabachène, Phong Q. Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 528–558, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[GSW13] 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, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[HS15] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 641–670, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[Nus80] Henri J. Nussbaumer. Fast polynomial transform methods for multidimensional dfts. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '80, Denver, Colorado, April 9-11, 1980*, pages 235–237. IEEE, 1980.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[SV14] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014.

## A FHEW NAND computation

The refreshing scheme of [DM15], as well as the algorithm $\mathsf{Refresh}$ of this work, are tailored to the refreshing of ciphertexts output by the FHEW NAND computation, and we paraphrase the NAND computation here for reference. FHEW gives an algorithm for the homomorphic computation of the NAND of two messages $m_0, m_1 \in \mathbb{Z}_2$, encrypted in $\mathsf{LWE}$ ciphertexts. The procedure, $\mathsf{HomNAND}$, takes as input two ciphertexts $(\mathsf{LWE}_{\mathbf{s}}^{4/q}(m_0, q/16), \mathsf{LWE}_{\mathbf{s}}^{4/q}(m_1, q/16))$, and outputs a ciphertext $\mathsf{LWE}_{\mathbf{s}}^{2/q}(m_0 \bar{\wedge} m_1, q/4)$. Note that even though the messages $m_0$ and $m_1$ are bits, the message space of the $\mathsf{LWE}$ ciphertext is 4 rather than 2.

Let $\mathsf{LWE}_{\mathbf{s}}^{4/q}(m_0, q/16) = (\mathbf{a}_0, b_0)$ and $\mathsf{LWE}_{\mathbf{s}}^{4/q}(m_1, q/16) = (\mathbf{a}_1, b_1)$ $\mathsf{HomNAND}$ computes

$$(\mathbf{a}, b) = (-\mathbf{a}_0 - \mathbf{a}_1, \frac{5q}{8} - b_0 - b_1) = (-\mathbf{a}_0 - \mathbf{a}_1, \mathbf{s} \cdot (-\mathbf{a}_0 - \mathbf{a}_1) - \frac{q}{4}(m_0 + m_1) - e_0 - e_1 + \frac{5q}{8})$$

so that

$$\begin{aligned} b - \mathbf{a}\cdot\mathbf{s} &= \frac{q}{4}(-m_0 - m_1) - e_0 - e_1 + \frac{5q}{8} \\ &= \frac{q}{4}(2 - 2m_0m_1 + \frac{1}{2} - (m_0 - m_1)^2) - e_0 - e_1 \\ &= \pm\frac{q}{8} - e_0 - e_1 + \frac{q}{2}(1 - m_0m_1) \end{aligned}$$

and therefore $(\mathbf{a}, b)$ is a $\mathsf{LWE}$ encryption of $(1 - m_0m_1) = m_0 \bar{\wedge}\, m_1$ with error $< \frac{q}{4}$.