Ring Packing and Amortized FHEW Bootstrapping

Daniele Micciancio
Department of Computer Science and Engineering, University of California - San Diego, CA, USA
daniele@cs.ucsd.edu

Jessica Sorrell
Department of Computer Science and Engineering, University of California - San Diego, CA, USA
jlsorrel@cs.ucsd.edu

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.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases homomorphic encryption, bootstrapping, lattice-based cryptography

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.100

Related Version A full version of the paper is available at `https://eprint.iacr.org/2018/532.pdf`.

Funding 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.

1 Introduction

Since Gentry's first construction of a Fully Homomorphic Encryption (FHE) scheme [13], much research has been done to improve both the security and efficiency of FHE. On the security front, a line of works [14, 8, 4, 6, 19] has led to a FHE scheme of Brakerski and Vaikuntanathan [9] based on learning with errors for polynomial approximation factors and therefore essentially *as secure as* regular (non-homomorphic) lattice-based *public-key encryption* [23].

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 [6] give a scheme for homomorphic evaluation of circuits of depth

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 100; pp. 100:1–100:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

L and security parameter λ requiring $\tilde{O}(\lambda \cdot L^3)$ per-gate computation. Gentry, Halevi, and Smart [18] used similar techniques to achieve homomorphic evaluation of width-$\Omega(\lambda)$ circuits with only polylog(λ) per-gate computation. However, these schemes place restrictions on circuit depth or size and additionally rely on the hardness of RingLWE for quasi-polynomial approximation factors, a stronger assumption than that used for public-key encryption.

Gentry's bootstrapping technique [13] 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 [5, 18, 17, 16, 15, 24, 20, 21], 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, and the amortized cost of bootstrapping over a large (and sufficiently wide) circuit.

A newer approach, explored in [1, 2, 12, 10], works towards reducing 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 [2] introduced a bootstrapping technique requiring $\tilde{O}(\lambda)$ homomorphic operations. Building upon this technique, Ducas and Micciancio [12] 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 [10]. However, it comes with the limitation that the bootstrapping procedure needs to be executed for essentially every gate of the circuit, without packing several messages into a single ciphertext. So bootstrapping is much faster than, say, in HElib [20, 21], 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 (where $n = \tilde{O}(\lambda)$), but at a cost comparable to that of [1, 2, 12, 10]. Our starting point is the FHEW bootstrapping method of [12]. We remark that FHEW has been improved in some follow-up works: [3] extended the FHEW scheme to larger gates, and [10] 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 [3] is limited to gates with at most $O(\log n)$ input wires, and the speed-up achieved in [10] is at most polylogarithmic. In fact, in this paper, we will make bootstrapping even slower than [12], 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 $\tilde{O}(3^{1/\epsilon} n^\epsilon)$ and for $\epsilon < 1/2$.

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] In [10], and in this work, 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.

1.1 The FHEW bootstrapping problem

The starting point of this work is the "FHEW" fully homomorphic encryption scheme of [12]. 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 δ. Similarly, we write $\mathsf{RingLWE}_d^{t/q}[m,\delta]$ for Ring LWE ciphertexts over the dth cyclotomic ring encrypting a polynomial $m(X)$ of degree $\varphi(d)$.[2] 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 $\mathsf{HomNAND}\colon \mathsf{LWE}_n^{4/q}[\delta] \times \mathsf{LWE}_n^{4/q}[\delta] \to \mathsf{LWE}_n^{2/q}[\Delta]$, 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", $b_0 \barwedge b_1 = \neg(b_0 \wedge b_1)$, but with somewhat larger noise Δ and encoded as an integer modulo 2. (We refer the reader to the full version of the paper for a formal definition and analysis of the $\mathsf{HomNAND}$ function.) Since these operations are not immediately composable, to evaluate arbitrary circuits on encrypted data, [12] also provides a refreshing procedure, $\mathsf{Refresh} : \mathsf{LWE}_n^{2/q}[\Delta] \to \mathsf{LWE}_n^{4/q}[\delta]$, that maps noisy ciphertexts modulo 2, back to ciphertexts modulo 4 with low noise δ. 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 [19], 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: approximately $O(n^{1+\epsilon})$ homomorphic modular multiplications on GSW ciphertexts. This reduces the amortized cost of refreshing to just $O(n^\epsilon)$ homomorphic (GSW) multiplications per ciphertext, rather than $O(n)$ as in the original FHEW cryptosystem.

▶ **Theorem 1.** *For every $0 < \epsilon < 1/2$, there exists an algorithm* $\mathsf{Refresh}$ *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 $\mathsf{HomNAND}$ 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 [8] to evaluate a function

$$\mathsf{PackLWE} : \left[\mathsf{LWE}_n^{2/q}[m_i, \Delta]\right]_{i<\varphi(d)} \to \mathsf{RingLWE}_d^{2/q}[m(X), \Delta'] \tag{1}$$

[2] 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 $\varphi(d) = 2d/3$.

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.

Looking ahead, the final step of the homomorphic decryption procedure will produce LWE ciphertexts of dimension equal to half the modulus of its input ciphertext. Therefore, to simplify the composition of $\mathsf{HomNAND}$ and $\mathsf{Refresh}$ operations, we use modulus switching after packing to reduce the modulus of the packed ciphertext to $2n$. We use a ring version of the modulus switching technique of [8] to compute a function

$$\mathcal{R}\text{-}\mathsf{ModSwitch}\colon \mathsf{RingLWE}_d^{2/q}[m(x), \Delta'] \to \mathsf{RingLWE}_d^{2/2n}[m(x), \frac{n}{q}\Delta' + \omega(\sqrt{d \log d}) \cdot \|z\|] \tag{2}$$

mapping ciphertexts modulo q under key $z \in \mathcal{R}_d/q$ to ciphertexts modulo $2n$, with the stated error bound (with high probability over the randomized rounding). Details of the modulus switching operation are provided in the full version.

We can now move on to homomorphically decrypt this Ring LWE ciphertext. Following the general bootstrapping framework of [13], 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, that we denote abstractly as $\mathsf{REG}^{2n/Q}[\cdot]$. Note that the message modulus $2n$ matches the ciphertext modulus of $\mathsf{RingLWE}_d^{2/2n}$, which is required for entrywise encryption of the $\mathsf{RingLWE}^{2/2n}$ 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

$$\mathsf{RingDecrypt} : \mathsf{RingLWE}_d^{2/2n}[m(x), \Delta''] \to \left[\mathsf{REG}^{2n/Q}[\tilde{m}_i, \delta']\right]_{i<\varphi(d)} \tag{3}$$

which takes (as an implicit parameter) 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 $\mathsf{RingDecrypt}$ function is homomorphic in z, and it is computed simply by evaluating the linear component of the $\mathsf{RingLWE}$ decryption function homomorphically on $\mathsf{REG}^{2n/Q}[\mathsf{encode}(z)]$. This entails the multiplication of an encrypted polynomial by a known polynomial, which introduces some technical challenges.

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 [22].

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 and 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 ith level of recursion, we fix $Y = X^{n^\epsilon}$ for all steps. Therefore, for all ϵ, the algorithm admits at most $1/\epsilon$ recursive calls. 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 transforms inverted to give the encrypted product required for homomorphic ring decryption.

The second step of the refreshing procedure is the "rounding" of the REG ciphertexts to low-noise $\mathsf{LWE}_n^{4/Q}$ ciphertexts. 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, 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 $\mathsf{msbExtract}\colon \mathsf{REG}^{2n/Q}[\tilde{m}_i, \delta'] \to \mathsf{LWE}_n^{4/Q}[m_i, \delta']$ and the standard modulus switching procedure

$$\mathsf{ModSwitch}\colon \mathsf{LWE}_n^{4/Q}[\delta'] \to \mathsf{LWE}_n^{4/q}[\frac{q}{Q}\delta' + \sqrt{2\pi(1 + \|\mathbf{s}\|^2)}] \tag{4}$$

for a LWE ciphertext under key $\mathbf{s} \in \mathbb{Z}_Q^{2n}$, which increases the noise by a small additive term . Parameters will be set in such a way that the resulting noise is small enough to apply the $\mathsf{HomNAND}$ function and keep computing on encrypted data. This completes the high level description of our boootstrapping method.

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 α if its tails are dominated by a Gaussian of parameter α, 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 $E[e^{2\pi tX}] \leq e^{\pi\alpha^2 t^2}$, 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 α 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 α. This extends to random matrices, where $\mathbf{X}^{m\times n}$ is subgaussian of parameter α 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 α. It follows immediately from these definitions that the concatenation of independent subgaussian vectors, all with parameter α, interpreted as either a vector or matrix, is also subgaussian with parameter α.

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 Nth cyclotomic polynomial, where $\omega_N = e^{2\pi \mathsf{i}/N} \in \mathbb{C}$ is the complex Nth 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 Φ_N is called the Nth 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 $\boldsymbol{(}a\boldsymbol{)} = (a_0, \ldots, a_{\varphi(N)-1}) \in \mathbb{Z}^{\varphi(N)}$. For any ring element $a \in \mathcal{R}_N$, $\|a\|$ is taken to mean the L^2 norm of the corresponding vector $\boldsymbol{(}a\boldsymbol{)} \in \mathbb{Z}^{\varphi(N)}$. Ring elements $a, b \in \mathcal{R}_N$ also admit a matrix representation $\mathbf{M}_a \in \mathbb{Z}^{\varphi(N)\times\varphi(N)} = [\boldsymbol{(}a \cdot X^0\boldsymbol{)}, \boldsymbol{(}a \cdot X^1\boldsymbol{)}, \ldots, \boldsymbol{(}a \cdot X^{\varphi(N)-1}\boldsymbol{)}]$ (used in Section 4.5,) such that $\mathbf{M}_a \cdot \boldsymbol{(}b\boldsymbol{)} = \boldsymbol{(}a \cdot b\boldsymbol{)}$. For any positive integer q, we write $\mathcal{R}_N/q$ for the quotient $\mathcal{R}_N/(q \cdot \mathcal{R}_N)$, 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 2** (Recall from [11], Fact 6)**.** *If $\mathcal{D}$ is a subgaussian distribution of parameter α 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 notation and working definitions for the basic LWE encryption scheme that our bootstrapping procedure operates on. For complete definitions, we refer the reader to the full version.

▶ **Definition 3** ((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 β is denoted $\mathcal{R}_N\text{-}\mathsf{LWE}_{\mathbf{s}}^{t/q}[m, \beta] = \{(\mathbf{a}, b) \mid \mathbf{a} \in \mathbb{R}_N^n/q, \|\mathbf{a} \cdot \mathbf{s} - mq/t\| \leq \beta\}$. 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 $\mathsf{LWE}^{t/q}_{\mathbf{s}}[m, \beta]$ (or $\mathsf{LWE}^{t/q}_{n}[m, \beta]$) 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 $\mathsf{RingLWE}^{t/q}_{N}$ as an abbreviation for $\mathcal{R}_N$-$\mathsf{LWE}^{t/q}_{1}$.

The (Ring) LWE decryption procedure plays a central role in our FHE bootstrapping process, specifically in the $\mathsf{RingDecrypt}$ procedure. The decryption of an $\mathcal{R}_N$-$\mathsf{LWE}^{t/q}_{\mathbf{s}}$ ciphertext $(\mathbf{a}, b) \in (\mathcal{R}^n_N, \mathcal{R}_N)/q$ is $\mathsf{Dec}(\mathbf{s}, (\mathbf{a}, b)) = \lfloor t(b - \mathbf{a} \cdot \mathbf{s})/q \rceil \bmod t \in \mathcal{R}_N/t \equiv \mathbb{Z}_t^{\varphi(N)}$. It is easy to check that for all $(\mathbf{a}, b) \in \mathsf{LWE}^{t/q}_{\mathbf{s}}[m, q/2t]$, the decryption procedure correctly recovers the encrypted message.

2.5 Ring-GSW encryption

The cryptographic accumulators of [12] (and the extended cryptographic registers defined in our work) make use of a ring variant of the GSW encryption scheme [19], which we now briefly describe. Let $\mathcal{R}_{N/Q}$ be the Nth 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 $\mathsf{GSW}^{N/Q}_z(m) = [\mathbf{a}, \mathbf{a} \cdot z + \mathbf{e}] + m \cdot \mathbf{g} \otimes \mathbf{I}_2$ where $\mathbf{g} = (B^0, B^1, B^2, \ldots, Q/B)$ for some base B. Similarly as for LWE, we write $\mathsf{GSW}^{N/Q}_z[m, \beta]$ for the set of ciphertexts encrypting m under z with error at most $\|\mathbf{e}\| \leq \beta$. 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. $\mathsf{GSW}^{N/Q}_z(m_0 \cdot m_1) = C_0 \times C_1$ 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 GSW ciphertexts (with similar initial error), then, the multiplications should be evaluated in a right-associative sequence.

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 $m(X) = \sum_i m_i X^{i-1}$.

▶ **Lemma 4.** *There exists a quasi-linear time algorithm that on input $\varphi(d)$ ciphertexts $c_i \in \mathsf{LWE}^{t/q}_{\mathbf{s}}(m_i, \Delta)$ (for $i = 0, \ldots, \varphi(d) - 1$, all under the same secret key $\mathbf{s} \in \mathbb{Z}^n_q$) and a packing key consisting of Ring-LWE encryptions $K_{j,l} \in \mathsf{RingLWE}^{q/q}_{\tilde{z}}[s_l 2^j, \beta_P]$ (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 $c \in \mathsf{RingLWE}^{t/q}_{\tilde{z}}[m(X), \beta]$ of $m(X) = \sum_i m_i X^i$ under $\tilde{z}$ with error at most $\beta = O(\sqrt{d}\Delta + \sqrt{dn \log q}\beta_P)$.*

The proof of Lemma 4 and accompanying pseudocode may be found in the full version, but we summarize its conclusion here. Let $K_{j,l} = (\tilde{a}'_{j,l}, \tilde{b}'_{j,l})$ be the entries of the packing key and $c_i = (\mathbf{a}_i, b_i)$ be the input ciphertexts. Defining $\sum_{j < \log q} \tilde{\mathbf{a}}_j 2^j = \sum_i \mathbf{a}_i \cdot X^{i-1} \in \mathcal{R}^n_{d,q}$ and taking $\tilde{a}'' = -\sum_{j,l} \tilde{a}_{j,l} \tilde{a}'_{j,l}$ and $\tilde{b}'' = \tilde{b} - \sum_{j,l} \tilde{a}_{j,l} \cdot \tilde{b}'_{j,l}$ gives the desired packed ciphertext $(\tilde{a}'', \tilde{b}'')$ encrypting $m(X)$.

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 omitted here, but provided in the full version of the paper.

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 Nth 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 GSW ciphertexts, writing $\mathsf{GSW}^{2n/Q}$. This choice of parameters is more thoroughly justified in Section 4.5, but as the homomorphic decryption procedure will produce LWE ciphertexts of dimension $N/2$, taking $N/2 = n$, where n is the original dimension of the LWE ciphertexts, allows us to omit an additional step of key switching back to dimenion n.

These 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

$$\mathsf{REG}_z^{q/Q}(v, \beta) = (\mathsf{GSW}_N^{q/Q}(uX^v), \mathsf{GSW}_N^{q/Q}(uX^{-v})).$$

4.2 Slow Multiplication

The use of the REG scheme restricts our arithmetic circuits to use only the operations described above. Given the asymmetric error growth of the underlying 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 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 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 computes the linear decryption step $C_{a,b}([\![f(\tilde{z})]\!]) = [\![b - a \cdot \tilde{z}]\!]$. 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 jth 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 $C_{a_i}([\![f(\tilde{z})]\!], k) = \sum_{j=0}^{l-1} a_{i,j} [\![f(\tilde{z})_{j,k}]\!]$ 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 Φ_d) using these C_{a_i} subcircuits, addition, and subtraction gates.

▶ **Lemma 5.** *Let B be the base of the geometric progression defining* $\mathbf{g}$ *in* GSW *encryption, and let $d_b = \lceil \log_B Q \rceil$. There is an algorithm*

$\mathsf{SlowMult} : a \in \mathcal{R}_d/2n \times (\mathsf{REG}_s(f(\tilde{z})_{j,k}, \beta))_{j,k} \to (\mathsf{REG}_s((a \cdot \tilde{z})_i, \beta'))_i$

requiring $\tilde{O}(d^2)$ homomorphic operations, and where $\beta' \leq \tilde{O}(\beta B \sqrt{n d_B d})$ with high probability.

The proof of Lemma 5 appears in the full version, and the analysis of error growth is similar to that of [11]. To briefly justify its stated complexity, we observe that SlowMult is just naive polynomial multiplication algorithm using only additions, and therefore taking time $O(l)$ per scalar multiplication. Therefore its complexity is $\tilde{O}(d^2)$. 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 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 mth principal root of unity ω_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 mth principal root of unity ω_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 ω_m^{-ik}. If we take $\omega_m \in \mathbb{Z}_{2n}$, each multiplication by ω_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 transform, 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 dimension smaller 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 inverse map on the inverse DFT of the smaller polynomial products yields $a \cdot \tilde{z} \in \mathcal{R}_d/2n$. 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/mth 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

$$f(\tilde{z}) = (1, 2, 4, ..., n) \otimes [\frac{1}{3m} DFT(\psi(\tilde{z}))_i]_{i<3m} \in (\mathcal{R}_{d/m}/2n)^{l \times 3m} \tag{5}$$

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 ith 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, for $2^l \leq n$, 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 ψ 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.

The Nussbaumer transform admits a recursive algorithm that gives tradeoffs between runtime and error growth of the cryptographic registers, but we defer consideration of the recursive formulation to Section 4.7.

▶ **Lemma 6.** *Let $\mathcal{K}^R$ be a refreshing key*

$$\mathcal{K}^R = K^R_{i,j,k} = \mathsf{REG}_z^{N/Q}([f(\tilde{z})_{i,j,k}]) = \mathsf{REG}_z^{N/Q}\left[\frac{1}{3m} DFT(\psi(\tilde{z}))_{i,k} 2^j, \beta_R\right] \text{ (from Eq. 5)}$$

where $DFT(\psi(\tilde{z}))_{i,k}$ indicates the kth coefficient of the ith 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 $\left[\mathsf{REG}_z^{N/Q}[\tilde{m}_i, \beta']\right]_{i<\varphi(d)}$ with $\beta' < \tilde{O}\left(\beta B^4 (n d_B)^{3.5} d^{\epsilon+.5}\right)$, and requiring $\tilde{O}(d^{2-\epsilon})$ homomorphic operations.*

The proof of Lemma 6 may be found in the full version of the paper, but the stated complexity is justified here informally. Lemma 5 states that SlowMult requires $\tilde{O}(d^2)$ operations on input polynomials of degree $O(d)$. The RingDecrypt algorithm multiplies $3m$ polynomials of degree $O(d/m)$, so these multiplications contribute $\tilde{O}(d^{2-\epsilon})$ homomorphic operations. The complexity of the DFT step, which performs $3m$ summations of $3m$ polynomials with degree $O(d/m)$, will then be dominated by that of SlowMult for all permissible ϵ, and therefore RingDecrypt will require $\tilde{O}(d^{2-\epsilon})$ many homomorphic operations.

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 of [12] to each of the $\varphi(d)$ registers produced by RingDecrypt. As in FHEW, this procedure produces refreshed ciphertexts with the GSW modulus Q, so must be followed by ModSwitch to convert them to the smaller intial modulus q.

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 (z) 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 $(z) = \mathbf{s}$ 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 [10], but this assumption may be removed at the cost of the additional key switching step. The proof of Lemma 7 may be found in the full version, and is similar to that of [12].

▶ **Lemma 7.** *There is an algorithm* msbExtract *that, given a cryptographic register of the form $\mathsf{REG}_z^{2n/Q}(b_i - (a \cdot \tilde{z})_i, \beta)$ as input, with $(z) = \mathbf{s}$, outputs a LWE ciphertext $\mathsf{LWE}_{\mathbf{s}}^{4/Q}(m_i, \sqrt{n} \cdot \beta)$.*

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 Δ, 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 $\mathcal{K}^P = K^P_{j,l}$ 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 $K^P_{j,l} = \mathsf{RingLWE}^{q/q}_{\tilde{z}}(s_l 2^j, \beta_P)$. The refreshing key $\mathcal{K}^R = K^R_{i,j,k}$ 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 $K^R_{i,j,k} = \mathsf{REG}^{2n,Q}_z\left(\frac{1}{3m} DFT(\psi(\tilde{z}))_{i,k}) 2^j, \beta_R\right)$.

▶ **Theorem 8.** *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 6, 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 of m_i).

From the error bounds of Theorem 4 and Equation 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 Δ 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 bounds of Lemma 6, Lemma 7, and Equation 4, 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 [7]), 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. ◀

4.7 Recursive optimization

In this section we summarize a recursive formulation of the Nussbaumer transform, and how it can improve the complexity of the RingDecrypt algorithm, at the cost of an increase in the error. The proof of Theorem 9 may be found in the full version.

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 ρ 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 $d/m^\rho \geq 3m$. ϵ is fixed and so this bounds the recursive depth of the algorithm by $\rho < \frac{1}{\epsilon} - 1$.

▶ **Theorem 9.** *The recursive $\mathsf{RingDecrypt}_\rho$ algorithm, with constant parameter ϵ 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 $\tilde{O}(B^{3\rho+1}(nd_B)^{3\rho}\sqrt{nd_B d^{1+\rho\epsilon}})$.*

References

1 Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40041-4_1.

2 Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_17.

3 Jean-François Biasse and Luis Ruiz. FHEW with efficient multibit bootstrapping. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America*, volume 9230 of *Lecture Notes in Computer Science*, pages 119–135. Springer, Heidelberg, August 2015. doi:10.1007/978-3-319-22174-8_7.

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

5 Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 1–13. Springer, Heidelberg, February / March 2013. doi:10.1007/978-3-642-36362-7_1.

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

7 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 Annual ACM Symposium on Theory of Computing*, pages 575–584. ACM Press, June 2013.

8 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, 43(2):831–871, 2014. URL: https://doi.org/10.1137/120868669, doi:10.1137/120868669.

9 Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *ITCS 2014: 5th Innovations in Theoretical Computer Science*, pages 1–12. Association for Computing Machinery, January 2014.

10 Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53887-6_1.

11 Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 335–352. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_19.

12 Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_24.

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

14 Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 107–109. IEEE Computer Society Press, October 2011.

15 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: 8th International Conference on Security in Communication Networks*, volume 7485 of *Lecture Notes in Computer Science*, pages 19–37. Springer, Heidelberg, September 2012.

16 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: 15th International Conference on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16. Springer, Heidelberg, May 2012.

17 Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, Heidelberg, April 2012.

18 Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, Heidelberg, August 2012.

19 Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40041-4_5.

20 Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_31.

21 Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_25.

22 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. URL: https://doi.org/10.1109/ICASSP.1980.1170902, doi:10.1109/ICASSP.1980.1170902.

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

24 Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014. URL: https://doi.org/10.1007/s10623-012-9720-4, doi:10.1007/s10623-012-9720-4.