FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second

Léo Ducas[1](✉) and Daniele Micciancio[2]

[1] Centrum Wiskunde and Informatica, Amsterdam, Netherlands
leo.ducas@cwi.nl
[2] University of California, San Diego, California, USA
daniele@cse.ucsd.edu

Abstract. The main bottleneck affecting the efficiency of all known fully homomorphic encryption (FHE) schemes is Gentry's bootstrapping procedure, which is required to refresh noisy ciphertexts and keep computing on encrypted data. Bootstrapping in the latest implementation of FHE, the HElib library of Halevi and Shoup (Crypto 2014), requires about six minutes. We present a new method to homomorphically compute simple bit operations, and refresh (bootstrap) the resulting output, which runs on a personal computer in just about half a second. We present a detailed technical analysis of the scheme (based on the worst-case hardness of standard lattice problems) and report on the performance of our prototype implementation.

1 Introduction

Since Gentry's discovery of the first fully homomorphic encryption (FHE) scheme [15], much progress has been made both towards basing the security of FHE on more standard and well understood security assumptions, and improving the efficiency of Gentry's initial solution.

On the security front, a sequence of papers [2,5,8,9,16] has lead to (leveled) FHE schemes based on essentially the same intractability assumptions underlying standard (non homomorphic) lattice based encryption. To date, the main open theoretical problem still left to be solved is how to remove the "circular security" assumption made in [15] (and all subsequent works) to turn a leveled FHE scheme (i.e., a scheme where the homomorphic computation depth is chosen at key generation time) into a full fledged one which allows to perform arbitrarily large homomorphic computations on encrypted data, even after all key material has been fixed.

This research was supported in part by the DARPA PROCEED program and NSF grant CNS-1117936. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or NSF.

L. Ducas—This research was done while the author was employed by University of California, San Diego.

E. Oswald and M. Fischlin (Eds.): EUROCRYPT 2015, Part I, LNCS 9056, pp. 617–640, 2015.
DOI: 10.1007/978-3-662-46800-5_24

Improving the efficiency of Gentry's scheme has received even more attention [1,2,6,17–21,23], resulting in enhanced asymptotic performance, and some reference implementations and libraries [17,23] that are efficient enough to be run on a personal computer. Still, the cost of running FHE schemes is quite substantial. The main bottleneck is caused by the fact that all current FHE solutions are based on "noisy" encryption schemes (based on lattices or similar problems) where homomorphic operations increase the noise amount and lower the quality of ciphertexts. As more homomorphic operations are performed, the noise can easily grow to a level where the ciphertexts are no longer decryptable, and operating on them produces meaningless results. Gentry's breakthrough discovery [15] was an ingenious "bootstrapping" technique (used in all subsequent works) that refreshes the ciphertexts by homomorphically computing the decryption function on encrypted secret key, and bringing the noise of the ciphertexts back to acceptable levels. This bootstrapping method allows to homomorphically evaluate arbitrary circuits, but it is also the main bottleneck in any practical implementation due to the complexity of homomorphic decryption.

Going back to efficiency considerations, the current state of the art in terms of FHE implementation is represented by the recent HElib of Halevi and Shoup [23,24], which reported a bootstrapping/refreshing procedure with running times around 6 minutes. While this is much better than previous implementations, and a nontrivial amount of computation can be performed in-between refreshing operations, the fact that even the simplest computation requiring bootstrapping takes such a macroscopic amount of time makes FHE clearly unattractive.

Our Work. The goal of this paper is to investigate to what extent the running time of a useful FHE bootstrapping procedure can be reduced. We do so by analyzing bootstrapping *in vitro*, i.e., in the simplest possible setting: given two encrypted bits $E(b_1)$ and $E(b_2)$, we want to compute their logical NAND (or any other complete boolean operation) and obtain the encrypted result $E(b_1 \bar{\wedge} b_2)$ in a form which is similar to the input bits. As in the most recent FHE schemes, here $E(\cdot)$ is just a standard lattice (LWE [32]) encryption scheme. In particular, $E(b_i)$ are noisy encryptions, and the output ciphertext $E(b_1 \bar{\wedge} b_2)$ is homomorphically decrypted (i.e., bootstrapped) in order to reduce its noise level back to that of $E(b_1)$ and $E(b_2)$. Our main result is a new boostrapping method and associated implementation that allows to perform the entire computation (consisting of homomorphic NAND computation and homomorphic decryption/bootstrapping) in less than a second on a standard (consumer grade) personal computer as detailed in Section 6.4.

We remark that the problem solved here is definitely simpler than HElib [23], as we perform only a single bit operation before bootstrapping, while [23] allows to perform more complex operations. In fact, using complex ciphertexts packing and homomorphic SIMD techniques, [23] achieves an amortized cost (per homomorphic bit operation) which we estimate to be in the same order of magnitude as our solution. The main improvement with respect to previous work is in terms of granularity and simplicity: we effectively show that half hour delays are not a necessary requirement of bootstrapped FHE computations, and bootstrapping itself can be achieved at much higher speeds than previously thought possible. Another attractive feature of the scheme presented in this paper is simplicity: we implemented our fully bootstrapped NAND computation in just a few hundreds lines of code and just a few days of programming effort.

Finally, our methods are not necessarily limited to a single NAND computation. As a simple extension of our basic scheme we show how to compute (homomorphically, and at essentially the same level of efficiency) various other operations, like majority, threshold gates. This extension also offers *xor-for-almost-free* as previous homomorphic schemes. Combining our fast (subsecond) bootstrapping method with other techniques that allow to perform substantially more complex computations in-between bootstrappings, is left as an open problem.

Techniques. Our improvement is based on two main techniques. One is a new method to homomorphically compute the NAND of two LWE encryptions. We recall that LWE encryption satisfies certain approximate additive homomorphic properties. Specifically, given two encryptions $E(m_1)$ and $E(m_2)$ one can compute a noisier version of $E(m_1 + m_2)$. When working modulo 2, this allows to homomorphically compute the exclusive-or of two bits. The way we extend this operation to a logical NAND computation is by moving (during boostrapping) from arithmetic modulo 2 to arithmetic modulo 4. So, adding $E(m_1)$ and $E(m_2)$ results in the encryption $E(m)$ of $m = 2$ (if $m_1 \barwedge m_2 = 0$) or $m \in \{0, 1\}$ (if $m_1 \barwedge m_2 = 1$). Moving from this ciphertext to the encryption of $m_1 \barwedge m_2$ is then achieved by a simple affine transformation.

The main advantage of our new homomorphic NAND operation is that it introduces a much lower level of noise than previous techniques. So, the refreshing procedure (required for bootstrapping) is faced with a much simpler task. Our second technical contribution builds on a recent method from [2] to implement and speed up bootstrapping. Decryption of LWE ciphertexts requires essentially the computation of a scalar product (modulo q) and a rounding operation. So, homomorphic decryption needs to compute these operations on encrypted data. The scheme of [2] uses a homomorphic cryptosystem that encrypts integers modulo q, and allows the efficient computation of scalar products. This is achieved using a homomorphic encryption scheme for binary messages $x \in \{0, 1\}$, and encoding elements $v \in C$ of a cyclic group as vectors of ciphertexts $E(x_1), \ldots, E(x_{|C|})$, where $x_i = 1$ if and only if $i = v$. We introduce a ring variant of the bootstrapping method of [2] that also supports efficient homomorphic computation of scalar products modulo q. The use of ring lattices was first suggested[1] in [31] to reduce the asymptotic computation time of lattice cryptography from quadratic to quasi-linear (using FFT techniques), and have become a fundamental technique to bring theoretical lattice constructions to levels of performance that are attractive in practice. Our work uses the LWE instantiation of ring lattices [29,30] for the efficient implementation of encryption. But our bootstrapping method goes beyond the use of ring

[1] Similar lattices had previously been used in practice also by the NTRU cryptosystem [25] , but without employing quasi-linear FFT techniques, and no connection to the worst-case complexity of lattice problems.

lattices to speed up normal lattice operations. We also use the ring structure of these lattices to directly implement the encryption of cyclic groups by encoding the cyclic group $\mathbb{Z}_q$ into the group of roots of unity: $i \mapsto X^i$ where i is a primitive q-th root of unity. This allows to implement a bootstrapping method similar to [2], but where each cyclic group element is encoded by a single ciphertext, rather than a vector of ciphertexts.

As a last technique, in order to contain noise generation during key switching operations, we use LWE instances with binary secrets, which were recently proved as hard as standard LWE in [7].

Like all previously known schemes, our FHE construction requires (in addition to standard worst-case lattice intractability assumptions) a circular security assumption in order to release a compact homomorphic evaluation key, and allow to combine an arbitrarily large number of homomorphic bit operations.

Organization. The rest of the paper is organized as follows. In section 2 we give some background on lattices and related techniques as used in the paper. In Section 3 we present a detailed description of the LWE encryption scheme that we want to bootstrap. The high level structure of our bootstrapped homomorphic NAND computation is given in Section 4. Section 5 goes into the core of our new refreshing procedure based on ring lattices. Section 6 describes concrete parameters, implementation and performance details. Section 7 concludes the paper with extensions and open problems.

2 Preliminaries

We will use bold-face lower-case letters $\mathbf{a}, \mathbf{b} \ldots$ to denote column vectors over $\mathbb{Z}$ or any other ring $\mathcal{R}$, and boldface upper-case letters $\mathbf{A}, \mathbf{B} \ldots$ for matrices. The product symbol $\cdot$ will be used for both scalar products of two column vectors, and for matrix product, to be interpreted as the only applicable one. The norm $\|\cdot\|$, will denote the euclidean norm. When speaking of the norm of a vector $\mathbf{v}$ over the residue ring $\mathbb{Z}_Q$ of $\mathbb{Z}$ modulo Q, we mean the shortest norm among the equivalence class of $\mathbf{v} \in \mathbb{Z}_Q^n$ in $\mathbb{Z}^n$.

2.1 Distributions

A *randomized rounding* function $\chi \colon \mathbb{R} \to \mathbb{Z}$ is a function mapping each $x \in \mathbb{R}$ to a distribution over $\mathbb{Z}$ such that $\chi(x+n) = \chi(x) + n$ for all integers n. For any $x \in \mathbb{R}$, the random variable $\chi(x) - x$ is called the rounding error of $\chi(x)$. As a special case, when the domain of χ is restricted to $\mathbb{Z}$, we have $\chi(x) = x + \chi(0)$, i.e., the randomized rounding function simply adds a fixed "noise" distribution $\chi(0)$ to the input $x \in \mathbb{Z}$.

A random variable X over $\mathbb{R}$ is *subgaussian* with parameter $\alpha > 0$ if for all $t \in \mathbb{R}$, the (scaled) moment-generating function satisfies $E[\exp(2\pi t X)] \leq \exp(\pi \alpha^2 t^2)$. If X is subgaussian, then its tails are dominated by a Gaussian of parameter α, i.e., $\Pr\{|X| \geq t\} \leq 2\exp(-\pi t^2/\alpha^2)$ for all $t \geq 0$. Any B-bounded symmetric random variable X (i.e., $|X| \leq B$ always) is subgaussian with parameter $B\sqrt{2\pi}$. More generally, we say that a random vector $\mathbf{x}$ (respectively, a random matrix $\mathbf{X}$) is subgaussian (of parameter α) if all its one-dimensional marginals $\langle \mathbf{u}, \mathbf{x} \rangle$ (respectively, $\mathbf{u}^t \mathbf{X} \mathbf{v}$) for unit vectors $\mathbf{u}, \mathbf{v}$ are subgaussian (of parameter α). It follows immediately from the definition that the concatenation of independent subgaussian vectors with common parameter α, interpreted as either a vector or matrix, is subgaussian with parameter α.

2.2 The Cyclotomic Ring

Throughout the paper, we let N be a power of 2 defining the $(2N)$th cyclotomic polynomial $\Phi_{2N}(X) = X^N + 1$ and associated cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. We also write $\mathcal{R}_Q = \mathcal{R}/(Q\mathcal{R})$ for the residue ring of $\mathcal{R}$ modulo an integer Q. Elements in $\mathcal{R}$ have a natural representation as polynomials of degree $N-1$ with coefficients in $\mathbb{Z}$, and $\mathcal{R}$ can be identified (as an additive group) with the integer lattice $\mathbb{Z}^N$, where each ring element $a = a_0 + a_1 x + \ldots + a_{N-1} x^{N-1} \in \mathcal{R}$ is associated with the coefficient vector $\overrightarrow{a} = (a_0, \ldots, a_{N-1}) \in \mathbb{Z}^N$. We extend the notation $\overrightarrow{\cdot}$ to any vector (or matrix) over $\mathcal{R}$ component-wise. We use the identification $\mathcal{R} = \mathbb{Z}^N$ to define standard lattice quantities like the euclidean length of a ring element $\|a\| = \|\overrightarrow{a}\| = \sqrt{\sum_i |a_i|^2}$, or the spectral norm of a matrix $\mathbf{R} \in \mathcal{R}^{w \times k}$ of ring elements $s_1(\mathbf{R}) = \sup_{\mathbf{x} \in \mathcal{R}^k \setminus \{0\}} \|\mathbf{R} \cdot \mathbf{x}\| / \|\mathbf{x}\|$.

The ring $\mathcal{R}$ is also identified with the sub-ring of anti-circulant square matrices of dimension N by regarding each ring element $r \in \mathcal{R}$ as a linear transformation $x \mapsto r \cdot x$ over (the coefficient embedding) of $\mathcal{R}$. The corresponding matrix is denoted $\overset{\Rightarrow}{r} \in \mathbb{Z}^{N \times N}$, and its first column is $\overrightarrow{r}$. (The other columns are the cyclic rotations of $\overrightarrow{r}$ with the cycled entries negated.) We extend the notation $\overset{\Rightarrow}{\cdot}$ to vectors and matrices over $\mathcal{R}$: for $\mathbf{R} \in \mathcal{R}^{w \times k}$, $\overset{\Rightarrow}{\mathbf{R}} \in \mathbb{Z}^{Nw \times Nk}$ is a matrix with anti-circulant $N \times N$ blocks. Notice that the definition of spectral norm of a ring element (or a matrice of ring elements) is consistent with the definition of spectral norm of the corresponding anticirculant matrix (or blockwise anti-circulant matrix): $s_1(r) = s_1(\overset{\Rightarrow}{r})$ and $s_1(\mathbf{R}) = s_1(\overset{\Rightarrow}{\mathbf{R}})$.

We say that a random polynomial a is subgaussian if its associated vector $\overrightarrow{a}$ is subgaussian. The fact that a is subgaussian *does not* imply that its associated anticirculant matrix $\overset{\Rightarrow}{a}$ is also subgaussian, because its columns are not independent. Nevertheless, subgaussianity of a ring elements still allows a good bound on its singular norm. This bound is as small as its non-ring counterpart as soon as either w or k is larger than $\omega(\sqrt{\log N})$.

Fact 1 (Adapted from [12], Fact 6). *If $\mathcal{D}$ is a subgaussian distribution of parameter α over $\mathcal{R}$, and $\mathbf{R} \leftarrow \mathcal{D}^{w \times k}$ has independents coefficients drawn from $\mathcal{D}$, then, with overwhelming probability, we have $s_1(\mathbf{R}) \leq \alpha\sqrt{N} \cdot O(\sqrt{w} + \sqrt{k} + \omega(\sqrt{\log N}))$.*

Invertibility in $\mathcal{R}$. Invertibility in cyclotomic rings has to be handled with care. (E.g., see [12].) The main issue is that, for a power-of-two cyclotomic ring $\mathcal{R} =$

$\mathbb{Z}[X]/(X^N + 1)$, the residue ring $\mathcal{R}_Q$ is never a field whatever the choice of Q. Yet, for appropriate moduli Q, it is not so far from being a field. More concretely, for Q a power of 3 most elements in $\mathcal{R}$ will be invertible, and so will most of the square matrices over $\mathcal{R}$ as detailed by the following Lemma 4. The lemma uses the following two facts.

Fact 2 (Irreducible factors of $X^N + 1$ modulo 3). *For any $k \geq 3$ and $N = 2^k$ we have $X^N + 1 = (X^{N/2} + X^{N/4} - 1) \cdot (X^{N/2} - X^{N/4} - 1) \bmod 3$ and both factors are irreducible in $\mathbb{F}_3[X]$.*

Proof. This follows directly from [26, Theorem 2.47].

Lemma 3 (Hensel Lemma for powers of prime integers). *Let $\mathcal{R}$ be the ring $\mathbb{Z}[X]/(F(X))$ for some monic polynomial $F \in \mathbb{Z}[X]$. For any prime p, if $u \in \mathcal{R}_{p^e}$ is invertible* mod p *(*i.e. it is invertible in $\mathcal{R}_p$) then u is also invertible in $\mathcal{R}_{p^e}$.

Lemma 4 (Invertibility of random matrices). *For Q a power of 3, and any dimension k, if $\mathcal{D}$ is a distribution over $\mathcal{R}_Q$ such that $\mathcal{D} \bmod 3$ is (statistically close to) uniform over $\mathcal{R}_3$, then, with overwhelming probability $\mathbf{D} \leftarrow \mathcal{D}^{k\times k}$ is invertible.*

Proof. By Fact 2, the ring $\mathcal{R}_3$ factors as $\mathcal{R}_3 = \mathbb{F}_1 \times \mathbb{F}_2$, where $\mathbb{F}_1 = \mathcal{R}/(3, P_1(X) = X^{N/2} + X^{N/4} - 1)$ and $\mathbb{F}_2 = \mathcal{R}/(3, P_2(X) = X^{N/2} - X^{N/4} - 1)$ are fields of order $q = \#\mathbb{F}_i = 3^{N/2}$. Note that $\mathbf{D} \bmod (3, P_i(X))$ is (statistically close to) a uniform random variable over $\mathbb{F}_i^{k\times k}$. We recall that the number of invertible matrices over the field of size q is given by

$$\begin{aligned}\#\mathcal{GL}(k, q) &= q^{k^2} \prod_{i=0}^{k-1} (1 - q^{i-k}) \\ &\geq q^{k^2} (1 - \sum_{i=1}^{k} q^{-i}) \geq q^{k^2} (1 - \frac{1}{q} \sum_{i\geq 0} q^{-i}) = q^{k^2} (1 - \frac{1}{q-1}).\end{aligned}$$

In particular $\mathbf{D} \bmod (3, P_i(X))$ is invertible except with probability $1/(q-1)$. By a union bound, $\mathbf{D}$ is invertible modulo both $(3, P_1(X))$ and $(3, P_2(X))$, except with negligible probability $2/(q-1) = 2/(3^{N/2}-1)$. It follows that $\mathbf{D}$ is invertible modulo 3, and by Hensel lifting (Lemma 3), also modulo Q. Indeed, Hensel lemma extends to *matrices* over $\mathcal{R}$, considering that a matrix $\mathbf{M} \in \mathcal{R}_Q^{k\times k}$ is invertible if and only if its determinant over $\mathcal{R}_Q$ is invertible.

3 LWE Symmetric Encryption

We recall the definition of the most basic LWE symmetric encryption scheme (see [3,4,32]). LWE symmetric encryption is parametrized by a dimension n, a message-modulus $t \geq 2$, a ciphertext modulus $q = n^{O(1)}$ and a randomized rounding function $\chi\colon \mathbb{R} \to \mathbb{Z}$. The message space of the scheme is $\mathbb{Z}_t$. (Typically, 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 $\mathbf{s} \in \mathbb{Z}_q^n$, which may be chosen uniformly at random, or as a random short vector. The encryption of a message $m \in \mathbb{Z}_t$ under key $\mathbf{s} \in \mathbb{Z}_q^n$ is

$$\mathsf{LWE}_{\mathbf{s}}^{t/q}(m) = (\mathbf{a}, \chi(\mathbf{a} \cdot \mathbf{s} + mq/t) \bmod q) \in \mathbb{Z}_q^{n+1} \tag{1}$$

where $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ 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 $\chi(0)$. A ciphertext $(\mathbf{a}, b)$ is decrypted by computing

$$m' = \lfloor t(b - \mathbf{a} \cdot \mathbf{s})/q \rceil \bmod t \in \mathbb{Z}_t. \tag{2}$$

We write $\mathsf{LWE}_{\mathbf{s}}^{t/q}(m)$ to denote the set of all possible encryptions of m under $\mathbf{s}$. The error of a ciphertext $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t/q}(m)$ 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 $(\mathbf{a}, b) \in \mathsf{LWE}_{\mathbf{s}}^{t/q}(m)$ can be any vector in $\mathbb{Z}_q^{n+1}$. We write $\mathsf{LWE}_{\mathbf{s}}^{t/q}(m, E)$ to denote the set of all ciphertexts $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}^{t/q}(m)$ with error bounded by $|\mathsf{err}(\mathbf{c})| < E$. 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:

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

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

Modulus Switching. LWE ciphertexts can be converted from one modulus Q to another q using the (scaled) randomized rounding function $[\cdot]_{Q:q}\colon \mathbb{Z}_Q \to \mathbb{Z}_q$ defined as

$$[x]_{Q:q} = \lfloor qx/Q \rfloor + B$$

where $B \in \{0, 1\}$ is a Bernoulli random variable with $\Pr\{B = 1\} = (qx/Q) - \lfloor qx/Q \rfloor \in [0, 1)$. Notice that $\mathbb{E}[[x]_{Q:q}] = \lfloor qx/Q \rfloor + \mathbb{E}[B] = qx/Q$ and $|[x]_{Q:q} - (qx/Q)| < 1$ with probability 1. In particular, the rounding error $[x]_{Q:q} - (qx/Q)$ is subgaussian of parameter $\sqrt{2\pi}$. The randomized rounding function is applied to vectors (e.g., LWE ciphertexts) coordinatewise:

$$\mathsf{ModSwitch}(\mathbf{a}, b) = [(\mathbf{a}, b)]_{Q:q} = (([a_1]_{Q:q}, \ldots, [a_n]_{Q:q}), [b]_{Q:q}). \tag{3}$$

Lemma 5. *For any* $\mathbf{s} \in \mathbb{Z}_q^n$, $m \in \mathbb{Z}_t$ *and ciphertext* $\mathbf{c} \in \mathsf{LWE}_{\mathbf{s}}^{t/Q}(m)$ *with subgaussian error of parameter* σ, *the rounding* $\mathsf{ModSwitch}(\mathbf{c}) = [\mathbf{c}]_{Q:q}$ *is a* $\mathsf{LWE}_{\mathbf{s}}^{t/q}(m)$ *ciphertext with subgaussian error of parameter* $\sqrt{(q\sigma/Q)^2 + 2\pi(||\mathbf{s}||^2 + 1)}$.

Proof. Let $\mathbf{c} = (\mathbf{a}, b)$ and $[\mathbf{c}]_{Q:q} = (\mathbf{a}', b')$. We have $a'_i = \frac{q}{Q}a_i + r_i$ and $b' = \frac{q}{Q}b + r_0$ for independent subgaussian rounding errors $r_0 \dots r_n$ of parameter $\sqrt{2\pi}$. It follows that $\mathbf{c}'$ is an $\mathsf{LWE}^{t/q}_{\mathbf{s}}$ encryption of m with error $\mathsf{err}(\mathbf{c}') = b' - \mathbf{a}' \cdot \mathbf{s} - \frac{qm}{t} = (q\mathsf{err}(\mathbf{c})/Q) + r_0 - \sum_{i=1}^n s_i r_i$. Since $\mathsf{err}(\mathbf{c}), r_0, \dots, r_n$ are independent subgaussian variables, their sum is also subgaussian, with parameter $\sqrt{(q\sigma/Q)^2 + 2\pi(||\mathbf{s}||^2 + 1)}$.

In practice, one may use the non-randomized rounding function $\lfloor \cdot \rceil$. Then, the error of the ouput of $\mathsf{ModSwitch}$, according to the central limit heuristic is expected to be close to a gaussian of standard deviation $\sqrt{(q\sigma/Q)^2 + (||\mathbf{s}||^2 + 1)/12}$, based on the randomness of $\mathbf{a}$. The factor 1/12 comes from the standard deviation of a uniform distribution in $[-\frac{1}{2}, \frac{1}{2}]$.

Key Switching. Key switching allows to convert an LWE encryption under a key $\mathbf{z} \in \mathbb{Z}_q^N$ into an LWE encryption of the same message (and slightly larger error) under a different key $\mathbf{s} \in \mathbb{Z}_q^n$. The key switching procedure is parametrized by a base B_{ks}, and requires as an auxiliary input a suitable encryption of $\mathbf{z}$ under $\mathbf{s}$. Specifically, let $\mathbf{k}_{i,j,v} \in \mathsf{LWE}^{q/q}_{\mathbf{s}}(vz_i B^j_{\mathsf{ks}})$ be an encryption of $vz_i B^j_{\mathsf{ks}}$ under $\mathbf{z}$, for all $i = 1, \dots, N$, $v \in \{0, \dots, B_{\mathsf{ks}}\}$ and $j = 0, \dots, d_{\mathsf{ks}} - 1$, where $d_{\mathsf{ks}} = \lceil \log_{B_{\mathsf{ks}}} q \rceil$. (Notice that the message $vz_i B^j_{\mathsf{ks}}$ is interpreted as a value modulo $t = q$, and therefore the ciphertext $\mathbf{k}_{i,j,v}$ is not typically decryptable because it has error bigger than $q/2t = 1/2$.) Given the switching key $\mathfrak{K} = \{\mathbf{k}_{i,j,v}\}$ and a ciphertext $(\mathbf{a}, b) \in \mathsf{LWE}^{t/q}_{\mathbf{z}}(m)$, the key swtching procedure computes the base-B_{ks} expansion of each coefficient $a_i = \sum_j a_{i,j} B^j_{\mathsf{ks}}$, and outputs

$$\mathsf{KeySwitch}((\mathbf{a}, b), \mathfrak{K}) = (\mathbf{0}, b) - \sum_{i,j} \mathbf{k}_{i,j,a_{i,j}}. \tag{4}$$

Lemma 6 . *The key switching procedure, given a ciphertext* $\mathbf{c} \in \mathsf{LWE}^{t/q}_{\mathbf{z}}(m)$ *with subgaussian error of parameter* α*, and switching keys* $\mathbf{k}_{i,j,v} = \mathsf{LWE}^{q/q}_{\mathbf{s}}(vz_i B^j_{\mathsf{ks}})$ *and subgaussian error of parameter* σ*, outputs an encryption* $\mathsf{KeySwitch}(\mathbf{c}, \{\mathbf{k}_{i,j,v}\}) \in \mathsf{LWE}^{t/q}_{\mathbf{z}}(m)$ *with subgaussian error of parameter* $\sqrt{\alpha^2 + N d_{\mathsf{ks}} \sigma^2}$.

Proof. Let $e_{i,j,v} = \mathsf{err}(\mathbf{k}_{i,j,v})$, so that $\mathbf{k}_{i,j,v} = (\mathbf{a}'_{i,j,v}, \mathbf{a}'_{i,j,v} \cdot \mathbf{s} + vz_i B^j_{\mathsf{ks}} + e_{i,j,v})$ for some $\mathbf{a}'_{i,j,v} \in \mathbb{Z}_q^n$. The output of the key switching procedure is $\mathsf{KeySwitch}(\mathbf{a}, b) = (\mathbf{a}', b')$ where $\mathbf{a}' = -\sum_{i,j} \mathbf{a}'_{i,j,a_{i,j}}$ and

$$b' = b - \sum_{i,j} (\mathbf{a}'_{i,j,a_{i,j}} \cdot \mathbf{s} + a_{i,j} z_i B^j_{\mathsf{ks}} + e_{i,j,a_{i,j}}) = b - \mathbf{a} \cdot \mathbf{z} + \mathbf{a}' \cdot \mathbf{s} - E,$$

where $E = \sum_{i,j} e_{i,j,a_{i,j}}$ is subgaussian with parameter $\sigma\sqrt{N d_{\mathsf{ks}}}$. It follows that $(\mathbf{a}', b')$ has error

$$\mathsf{err}(\mathbf{a}', b') = b' - \mathbf{a}' \cdot \mathbf{s} - \frac{qm}{t} = b - \mathbf{a} \cdot \mathbf{z} - E - \frac{qm}{t} = \mathsf{err}(\mathbf{a}, b) - E.$$

Since $\mathsf{err}(\mathbf{a}, b)$ and E are both subgaussian, their difference is also subgaussian with parameter $\sqrt{\alpha^2 + N d_{\mathsf{ks}} \sigma^2}$.

4 Our FHE: High Level Structure

In this section we describe the high level structure/design of our fully homomorphic (symmetric) encryption scheme. (This private-key FHE scheme can be transformed into a public-key one using standard techniques.) The encryption scheme itself is just the standard LWE symmetric encryption described in Section 3. For now we focus on encrypting single bits, and evaluating boolean NAND circuits. In summary, we need to solve the following problem: given two ciphertexts $c_i \in \mathsf{LWE}^{2/q}_{\mathbf{s}}(m_i)$ (for $i = 0, 1$), compute a ciphertext $c \in \mathsf{LWE}^{2/q}_{\mathbf{s}}(m)$ where $m = 1 - m_0 \cdot m_1 = m_0 \bar{\wedge}\, m_1$ is the logical NAND of m_0 and m_1.

4.1 A New Homomorphic NAND Gate

The main idea to perform this encrypted NAND computation is to assume that the input ciphertexts are available in a slightly different form. (We will see later how to perform the required transformation.) Namely, assume that the input bits $m_0, m_1 \in \{0, 1\}$ are encrypted as ciphertexts $c_i \in \mathsf{LWE}^{4/q}_{\mathbf{s}}(m_i, q/16)$ using a slighly different message modulus $t = 4$ and error bound $E = q/16$. (Compare to the standard binary LWE encryption parameters $t = 2$ and $E = q/4$.)

Lemma 7. *There is a simple algorithm*

$$\mathsf{HomNAND}\colon \mathsf{LWE}^{4/q}_{\mathbf{s}}(m_0, q/16) \times \mathsf{LWE}^{4/q}_{\mathbf{s}}(m_1, q/16) \to \mathsf{LWE}^{2/q}_{\mathbf{s}}(m_0 \bar{\wedge}\, m_1, q/4)$$

that on input two ciphertexts $\mathbf{c}_i \in \mathsf{LWE}^{4/q}(m_i, q/16)$ (for $i = 0, 1$) encrypting binary messages $m_0, m_1 \in \{0, 1\}$, outputs an encryption $\mathsf{HomNAND}(\mathbf{c}_0, \mathbf{c}_1) \in \mathsf{LWE}^{2/q}(m, q/4)$ of their logical NAND $m = 1 - m_0 m_1 = m_0 \bar{\wedge}\, m_1$ with error less than $q/4$.

Proof. The NAND of the two ciphertexts $c_i = (\mathbf{a}_i, b_i)$ can be easily computed as

$$(\mathbf{a}, b) = \mathsf{HomNAND}((\mathbf{a}_0, b_0), (\mathbf{a}_1, b_1)) = \left(-\mathbf{a}_0 - \mathbf{a}_1, \frac{5q}{8} - b_0 - b_1\right).$$

(Remember that we assumed for simplicity that $8 = 2t$ divides q, and therefore $5q/8$ is an integer.) The resulting ciphertext satisfies

$$b - \mathbf{a}\mathbf{s} - (1 - m_0 m_1)\frac{q}{2} = \frac{q}{4}(\frac{1}{2} - (m_0 - m_1)^2) - (e_0 + e_1) = \pm\frac{q}{8} - (e_0 + e_1).$$

So, $(\mathbf{a}, b) = \mathsf{HomNAND}(c_0, c_1)$ is a regular $\mathsf{LWE}^{2/q}_{\mathbf{s}}$ encryption of $1 - m_0 m_1 = m_0 \bar{\wedge}\, m_1$ with error at most

$$\left|\pm\frac{q}{8} - (e_0 + e_1)\right| < \frac{q}{8} + \frac{q}{16} + \frac{q}{16} = \frac{q}{4}.$$

Notice that the $\mathsf{HomNAND}$ function can be computed from the input ciphertexts without using any key material, and it requires just a handfull of additions modulo q. This shows that in order to compute the NAND of two ciphertexts (and therefore homomorphically evaluate any boolean circuit on LWE encryptions), it is enough to be able to compute a refreshing function

$$\mathsf{Refresh}\colon \mathsf{LWE}^2_{\mathbf{s}}(m, q/4) \to \mathsf{LWE}^4_{\mathbf{s}}(m, q/16).$$

The refreshing function will require some key material, and it will be substantially more expensive of $\mathsf{HomNAND}$, accounting essentially for the whole cost of homomorphic circuit evaluation. Notice that the number of refresh computations required to evaluate a circuit with g gates is $n + g$ (one for each circuit input and gate-output wire). Assuming that the encrypted input bits are already provided in refreshed form (e.g., by using the modified $\mathsf{LWE}^{4/q}(m, q/16)$ encryption scheme), one needs just 1 refresh evaluation per gate, applied to the output of the gate, rather than 2 evaluation (one for each input into the gate). So, the computational cost of homomorphically evaluating a NAND gate is essentially that of a single refresh function computation.

Improvement. Previous methods to compute homomorphic AND gates on LWE ciphertexts require errors of input to be at most $O(\sqrt{q})$, against $O(q)$ in our case. Our technique therefore relaxes the requirement on the $\mathsf{Refresh}$ procedure, potentially making the overall scheme faster.

4.2 Refreshing via Homomorphic Accumulator

We now move to the description of the refreshing function. As in all previous works on FHE, our ciphertext refreshing is based on Gentry's bootstrapping technique of homomorphically evaluating the decryption function. More specifically, in our setting, given an LWE ciphertext $(\mathbf{a}, b) \in \mathsf{LWE}^{2/q}_{\mathbf{s}}(m)$, we compute an encryption $E(m)$ of the same message under a different encryption scheme E by homomorphically evaluating the LWE decryption procedure (2) on the encrypted key $E(\mathbf{s})$ to yield

$$\lfloor 2(b - \mathbf{a} \cdot E(\mathbf{s}))/q \rceil \bmod 2 \simeq E(m).$$

We recall that the final goal of the refreshing function is to obtain an encryption in $\mathsf{LWE}^{4/q}(m, q/16)$. However, this target encryption scheme is not versatile enough to perform the required homomorphic computation. Instead, following [2], we use an intermediate encryption scheme E with message space $\mathbb{Z}_q$, which allows to encrypt the secret $\mathbf{s} \in \mathbb{Z}_q^n$ componentwise $E(\mathbf{s}) = (E(s_1), \ldots, E(s_n))$ and supports the efficient computation of affine transformations $b - \mathbf{a} \cdot E(\mathbf{s}) = E(b - \mathbf{a} \cdot \mathbf{s})$. Once this computation is done, it remains to homomorphically extract the most significant bit of $b - \mathbf{a} \cdot \mathbf{s}$ as an LWE ciphertext. We summarize our requirements under the following definition. Notice that the definition makes use of two (typically different) encryption schemes:

– a scheme E, which is used internally by the accumulator, and it is left unspecified to allow a wider range of possible implementations, and
– a target encryption scheme, which for simplicity we fix to $\mathsf{LWE}^{t/q}$ as required by our application.

The definition is easily generalized to make it parametric also with respect to the target encryption scheme.

Definition 1 (Homomorphic Accumulator) . *A Homomorphic Accumulator Scheme is a quadruple of algorithms* $(E, \mathsf{Init}, \mathsf{Incr}, \mathsf{msbExtract})$ *together with moduli* t, q*, where* E *and* $\mathsf{msbExtract}$ *may require key material related to an* LWE *key* $\mathbf{s}$*. For brevity, we write* $\mathsf{ACC} \leftarrow v$ *for* $\mathsf{ACC} \leftarrow \mathsf{Init}(v)$*, and* $\mathsf{ACC} \overset{+}{\leftarrow} E(v)$ *for* $\mathsf{ACC} \leftarrow \mathsf{Incr}(\mathsf{ACC}, E(v))$*. For any* $v_0, v_1 \dots v_\ell \in \mathbb{Z}_q$*, after the sequence of operations*

$$\mathsf{ACC} \leftarrow v_0; \quad \textbf{\textit{for}}\ i = 1\ \textbf{\textit{to}}\ \ell\ \textbf{\textit{do}}\ \mathsf{ACC} \overset{+}{\leftarrow} E(v_i)$$

we say that we say that ACC *is an* ℓ*-encryption of* v*, where* $v = \sum v_i \bmod q$*.*

A Homomorphic Accumulator Scheme is said $\mathcal{E}$*-correct for some function* $\mathcal{E}$ *if, for any* ℓ*-encryption* ACC *of* v*, computing* $\mathbf{c} \leftarrow \mathsf{msbExtract}(\mathsf{ACC})$ *ensures* $\mathbf{c} \in \mathsf{LWE}^{t/q}_{\mathbf{s}}(v, \mathcal{E}(\ell))$ *with overwelming probability.*

In order to use the accumulator in our refreshing function, we set $t = 4$ and we will need $\mathcal{E}(\ell) \le q/16$. Note that the correctness requirement assumes that all ciphertexts added to the accumulator are freshly generated and independent. (In particular, although the internal encryption scheme E may enjoy useful homomorphic properties, the ciphertexts $E(v_i)$ are generated by a direct application of the encryption function E on v_i, rather than performing homomorphic operations on ciphertexts.)

Using this accumulator data structure, we describe a family of refreshing procedures (exhibiting different space/time trade-offs) parametrized by an integer B_{r}. (The subscript in B_{r} stands for $\mathsf{Refresh}$, and it is used to distinguish B_{r} from similar basis parameters used elsewhere in the paper.) The refreshing procedure takes as input a ciphertext $(\mathbf{a}, b) \in \mathsf{LWE}^{2/q}_{\mathbf{s}}(m, q/4)$ and a refreshing key $\mathcal{K}$ consisting of the encryptions $K_{i,c,j} = E(cs_i B_{\mathsf{r}}^j \bmod q)$ for $c \in \{0, \dots, B_{\mathsf{r}} - 1\}$, $j = 0, \dots, d_{\mathsf{r}} - 1$ (where $d_{\mathsf{r}} = \lceil \log_{B_{\mathsf{r}}} q \rceil$) and $i = 1, \dots, n$. (In total, $nB_{\mathsf{r}}d_{\mathsf{r}} \approx n(B_{\mathsf{r}}/\log B_{\mathsf{r}}) \log q$ ciphertexts). It then proceeds as described in Algorithm 1.

Algorithm 1. $\mathsf{Refresh}_{\mathcal{K}}(\mathbf{a}, b)$, for $\mathcal{K} = \{K_{i,c,j}\}_{i \le n, c \le B_{\mathsf{r}}, j \le d_{\mathsf{r}}}$

```
ACC ← $b + (q/4)$
for $i = 1, \dots, n$ do
    Compute the base-$B_{\mathsf{r}}$ representation of $-a_i = \sum_j B_{\mathsf{r}}^j \cdot a_{i,j} \pmod q$
    for $j = 0, \dots, d_{\mathsf{r}} - 1$ do ACC $\overset{+}{\leftarrow} K_{i,a_{i,j},j}$
end for
Output msbExtract(ACC).
```

Theorem 8. *If* $(E, \mathsf{Init}, \mathsf{Incr}, \mathsf{msbExtract})$ *is a correct Homomorphic Accumulator Scheme, then the* Refresh *procedure, on input any ciphertext* $\mathbf{c} \in \mathsf{LWE}^{2/q}_{\mathbf{s}}(m, q/4)$*, and a valid refreshing key* $\mathcal{K} = \{K_{i,c,j} = E(cs_iB^j_{\mathsf{r}})\}_{i,c,j}$*, outputs a ciphertext* $\mathsf{Refresh}_{\mathcal{K}}(\mathbf{c}) \in \mathsf{LWE}^{t/q}_{\mathbf{s}}(m, \mathcal{E}(nd))$*.*

Proof. The refreshing procedure initializes the accumulator to $b + q/4$, and then adds nd (distinct, freshly generated) ciphertexts $K_{i,a_{i,j},j} = E(a_{i,j}s_iB^j_{\mathsf{r}})$ to it. So, the final output is (with overwhelming probability) an LWE encryption with error at most $\mathcal{E}(nd)$. The implicit value v of the accumulator at the end of the main loop is

$$v - \frac{q}{4} = b + \sum_{i,j} a_{i,j}s_iB^j_{\mathsf{r}} = b + \sum_i s_i \sum_j B^j_{\mathsf{r}} a_{i,j} = b + -\sum_i a_is_i = \frac{q}{2}m + e$$

where e is the error of the input ciphertext $(\mathbf{a}, b)$. Since $|e| < q/4$ by assumption, we have $0 < e + q/4 < q/2$. If follows that $0 < v < q/2$ if $m = 0$, and $q/2 < v < q$ if $m = 1$. Therefore, $\mathsf{msbExtract}(\mathsf{ACC})$ produces a $\mathsf{LWE}^{q/t}_{\mathbf{s}}(m, \mathcal{E}(nd))$ encryption as claimed.

5 Homomorphic Accumulator from Ring-GSW

In this section we show how to implement the homomorphic accumulator scheme needed by our refreshing procedure. As a reminder, the homomorphic accumulator is parametrized by a modulus $q = 2^k$ (which we assume to be a power of 2), an integer t (in our main application, $t = 4$), and an encryption scheme E with message space $\mathbb{Z}_q$.

Our construction follows the suggestion of Alperin-Sheriff and Peikert [2] to generalize their scheme. Essentially, we avoid the costly construction of the additive group $\mathbb{Z}_q$ as a subgroup of some symmetric group $\mathfrak{S}_\ell$ (represented as permutation matrices). Instead, we directly implement $\mathbb{Z}_q$ as the multiplicative (sub)group of the roots of unity of the ring $\mathcal{R}$.

5.1 Description

The scheme is parametrized by a modulus Q, a dimension $N = 2^K$ such that q divides $2N$, and a base B_{g}. (Here the subscript in B_{g} stands for gadget.) For simplicity of the analysis, we will assume that $Q = B_{\mathsf{g}}^{d_{\mathsf{g}}}$ for some integer d_{g}, and that B_{g} is a power of 3. We use the rings $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_Q = (\mathcal{R}/Q\mathcal{R})$ (see Section 2), and an additional parameter u, which should be an invertible element of $\mathbb{Z}_Q$ close to $Q/2t$. Since Q is a power of 3, either $\lfloor Q/2t \rfloor$ or $\lceil Q/2t \rceil$ is invertible, and we can let u be one of these two numbers, so that the distance $\delta = u - Q/2t$ is at most $|\delta| < 1$.

Messages $m \in \mathbb{Z}_q$ are encoded as roots of unity $Y^m \in \mathcal{R}$ where $Y = X^{2N/q}$. Notice that the roots of unity $\mathcal{G} = \langle X \rangle = \{1, X \ldots, X^{N-1}, -1, -X \ldots, -X^{N-1}\}$ form a cyclic group, and the message space $\mathbb{Z}_q \simeq \langle Y \rangle$ is a subgroup of $\mathcal{G} \simeq \mathbb{Z}_{2N}$. Our Homomorphic Accumulator Scheme is based on a variant of the GSW cryptosystem and works as follows:

– $E_z(m)$, on input a message m and a key $z \in \mathcal{R}$, picks $\mathbf{a} \in \mathcal{R}_Q^{2d_g}$ uniformly at random, and $\mathbf{e} \in \mathcal{R}^{2d_g} \simeq \mathbb{Z}^{2d_g N}$ with a subgaussian distribution χ of parameter ς, and outputs

$$E_z(m) = [\mathbf{a}, \mathbf{a} \cdot z + \mathbf{e}] + uY^m\mathbf{G} \qquad \in \mathcal{R}_Q^{2d_g \times 2}$$

where $\mathbf{G} = (\mathbf{I}, B_g\mathbf{I}, \ldots, B_g^{d_g-1}\mathbf{I}) \in \mathcal{R}_Q^{2d_g \times 2}$.

– Init $(\mathsf{ACC} \leftarrow v)$, on input $v \in \mathbb{Z}_q$, simply sets $\mathsf{ACC} := uY^v \cdot \mathbf{G} \in \mathcal{R}_Q^{2d_g \times 2}$.

– Incr $(\mathsf{ACC} \xleftarrow{+} \mathbf{C})$, on input the current accumulator content $\mathsf{ACC} \in \mathcal{R}_Q^{2d_g \times 2}$ and a ciphertext $\mathbf{C} \in \mathcal{R}_Q^{2d_g \times 2}$, first computes the base-B_g decomposition of $u^{-1}\mathsf{ACC} = \sum_{i=1}^{d_g} B_g^{i-1}\mathbf{D}_i$ (where each $\mathbf{D}_i \in \mathcal{R}^{2d_g \times 2}$ has entries with coefficients in $\{\frac{1-B_g}{2}, \ldots, \frac{B_g-1}{2}\}$), and then updates the accumulator to

$$\mathsf{ACC} := [\mathbf{D}_1, \ldots, \mathbf{D}_{d_g}] \cdot \mathbf{C}.$$

An efficient algorithm for Incr using FFT/NTT will be detailed in Section 5.3.

– $\mathsf{msbExtract}$ (defined by Algorithm 2) uses a key-switching auxiliary input $\mathfrak{K}$ (as defined in Section 2) and a testing vector $\mathbf{t} = -\sum_{i=0}^{q/2-1} \overrightarrow{Y^i}$. (On a first reading, the reader may want to focus on the special case where $q = 2N$, where the testing vector is just $\mathbf{t} = -(1, 1, \ldots, 1)$.) The algorithm follows. The crux of matter for the extraction of the msb is that $\mathbf{t} \cdot \overrightarrow{Y^v} = -1$ if $0 \leq i < N$, and $+1$ if $N \leq i < 2N$.

Before providing a detailed analysis (Theorem 10) we explain the ideas behind the definitions of our homomorphic accumulator. As already mentioned, the scheme is based on a variant of the (private-key, ring-based) GSW encryption scheme. There are two main differences between our scheme and the original GSW scheme: the use of the extra parameter u (which plays an important role in our msb extraction algorithm), and the fact that the messages are encrypted in the exponent (of Y). At any point in time, the accumulator data structure holds the encryption $E_z(v)$ of some value $v \in \mathbb{Z}_q$ under a fixed key z. The initialization step Init simply sets ACC to a trivial (noiseless) encryption of v. The increment procedure is similar to the *multiplicative* homomorphism of the GSW scheme [22]. Since our messages are *in the exponent*, this provides homomorphic additions of ciphertexts.

Algorithm 2. $\mathsf{msbExtract}_{\mathfrak{K}}(\mathsf{ACC})$, for $\mathfrak{K} = \{\mathbf{k}_{i,j,w}\}_{i \leq N, j \leq B_{\mathsf{ks}}, w \leq d_{\mathsf{ks}}}$

```
Require: A switching key $\mathfrak{K} = \{\mathbf{k}_{i,j,w}\}_{i,j,w}$ from $\mathbf{z}$ to $\mathbf{s}$: $\mathbf{k}_{i,j,w} \leftarrow \mathsf{LWE}_{\mathbf{s}}^{q/q}(w \cdot z_i \cdot d_{\mathsf{ks}}^j)$.
    An accumulator $\mathsf{ACC}$ that is an $\ell$-encryption of $v$.
1: $[\mathbf{a}^t, \mathbf{b}^t] \leftarrow ([\overrightarrow{0}^t, \mathbf{t}^t, \overrightarrow{0}^t, \ldots, \overrightarrow{0}^t] \cdot \overset{\Rightarrow}{\mathsf{ACC}}) \in \mathbb{Z}_Q^{2N}$    // $\overset{\Rightarrow}{\mathsf{ACC}} \in \mathbb{Z}^{2Nd_g \times 2N}$
2: $\mathbf{c} \leftarrow (\mathbf{a}, b_0 + u)$    $\in \mathsf{LWE}_{\overrightarrow{z}}^{t/Q}(\mathrm{msb}(v))$
3: $\mathbf{c}' \leftarrow \mathsf{KeySwitch}(\mathbf{c}, \mathfrak{K})$    $\in \mathsf{LWE}_{\mathbf{s}}^{t/Q}(\mathrm{msb}(v))$
4: $\mathbf{c}'' \leftarrow \mathsf{ModSwitch}(\mathbf{c}')$    $\in \mathsf{LWE}_{\mathbf{s}}^{t/q}(\mathrm{msb}(v))$
5: Return $\mathbf{c}'$.
```

5.2 Correctness

In this subsection we prove that ACC is a correct Homomorphic Accumulator Scheme for an appropriate error function $\mathcal{E}$. The main result is given in Theorem 10. But first, let us detail the behaviour of individual operations.

The Init operation $\mathsf{ACC} \leftarrow v$ sets up ACC to a noiseless encryption of v under E_z for any secret key z. The homomorphic property of Incr follows from the following claim.

Fact 9. *For any messages $m, m' \in \mathbb{Z}_q$, if $\mathsf{ACC} = [\mathbf{a}, \mathbf{a} \cdot z + \mathbf{e}] + uY^m\mathbf{G}$ and $\mathbf{C} = [\mathbf{a}', \mathbf{a}' \cdot z + \mathbf{e}'] + uY^{m'}\mathbf{G}$, then $\mathsf{ACC} \xleftarrow{+} \mathbf{C}$ has the form $[\mathbf{a}'', \mathbf{a}'' \cdot z + \mathbf{e}''] + uY^{m+m'}\mathbf{G}$ for $\mathbf{e}'' = \mathbf{e} + [\mathbf{C}_1, \ldots, \mathbf{C}_{d_\mathsf{g}}] \cdot \mathbf{e}'$.*

The last operation $\mathsf{msbExtract}$ is slightly more intricate. Let us put aside the key and modulus switching steps, and consider, as in the algorithmic definition of $\mathsf{msbExtract}$, the vector

$$\left[\mathbf{a}^t, b_0\right] \leftarrow \mathbf{t}^t \cdot \left[\overset{\Rightarrow}{a}, \overrightarrow{b'}\right]$$

where $[a, b'] \in \mathcal{R}^{1\times 2}$ is the second row of the accumulator $\mathsf{ACC} \in \mathcal{R}^{2d_\mathsf{g}\times 2}$. If ACC is a GSW encryption of a value v, $[a, b']$ verifies $\overrightarrow{b'} = \overset{\Rightarrow}{a} \cdot \overrightarrow{z} + u \cdot \overrightarrow{Y^v} + \mathbf{e}$ for some small error $\mathbf{e}$. Let's write $\overrightarrow{Y^v}$ as the vector $\mathbf{x}_{v\cdot 2N/q} \in \mathbb{Z}_Q^N$ defined as follows:

$$\mathbf{x}_i = \overrightarrow{X^i} = (\underbrace{0, \ldots, 0}_{i-1}, 1, 0 \ldots, 0) \text{ if } i \in \{0 \ldots N-1\}, \quad \mathbf{x}_i = -\mathbf{x}_{i-N} \text{ otherwise.}$$

For $i \in \mathbb{Z}_{2N}$, summing all coordinates of $\mathbf{x}_i$ results in $(-1)^{\mathrm{msb}(i)}$, and $\mathbf{t}^t \cdot \overrightarrow{Y^v} = -(-1)^{\mathrm{msb}(v)}$ for any $v \in \mathbb{Z}_q$. It remains to recall the identity $1 - (-1)^x = 2x$ for any bit $x \in \{0, 1\}$ to rewrite

$$\mathbf{c} = (\mathbf{a}, b_0 + u) = (\mathbf{a}, \mathbf{a} \cdot \overrightarrow{z} + \mathbf{t} \cdot \mathbf{e} + 2u\,\mathrm{msb}(v)) \qquad \text{where } \mathbf{a} = \mathbf{t}^t \cdot \overset{\Rightarrow}{a},$$

which is an $\mathsf{LWE}^{t/Q}_{\overrightarrow{z}}$ encryption of $\mathrm{msb}(v)$ since $u \approx Q/2t$. We may now move to the formal correctness statement, including bound on error size.

Theorem 10. *Assuming the hardness Ring-$\mathsf{LWE}_{\mathcal{R},Q,\chi}$ the above Homomorphic Accumulator Scheme is $\mathcal{E}$-correct with error function*

$$\mathcal{E}(\ell) = \sqrt{\frac{q^2}{Q^2}\left(\varsigma^2 B_\mathsf{g}^2 \cdot \ell \cdot q \cdot N d_\mathsf{g} + \sigma^2 N d_\mathsf{ks}\right) + \|\mathbf{s}\|^2} \cdot \omega(\sqrt{\log n}).$$

We obtain Theorem 10 by combining the following Lemma 11 with the correctness of Key Switching and Modulus Switching, Lemmata 6 and 5. The hardness assumption is not strictly necessary for correctness, but does simplify the proof by allowing one to assume that fresh ciphertexts $\mathbf{C} \leftarrow E_z(\cdot)$ behave as independent uniform random matrices.

Lemma 11 (Intermediate error). *Assume the hardness of Ring-*$\mathsf{LWE}_{\mathcal{R},Q,\chi}$*, and let* ACC *is an* ℓ*-encryption of* v *where* $\ell \geq \omega(\sqrt{\log N})$*. Then the ciphertext* $\mathbf{c} \in \mathsf{LWE}^{t/Q}_{\mathbf{z}}(\mathrm{msb}(v))$ *as define in line 2 of algorithm 2 while computing* $\mathsf{msbExtract}(\mathsf{ACC})$ *has an error* $\mathsf{err}(\mathbf{c})$ *which is a subgaussian with variable parameter* β *and mean* 2δ *under the randomness used in the calls to* $E_z(\cdot)$*, for* $\beta = O(\varsigma B\sqrt{q \cdot Nd_{\mathsf{g}} \cdot \ell})$.

Let us start with the following fact.

Fact 12 (Spectral Norm of Decomposed Matrices). *Let* $\mathbf{C}^{(i)} \leftarrow E_z(v^{(i)})$ *be fresh encryptions of* $v^{(i)} \in \mathbb{Z}_q$ *for all* $i \leq \ell = \omega(\sqrt{\log n})$*, and assume that the* $\mathbf{C}^{(i)}$*'s are indistinguishable from random without the knowledge of* z*. Consider* $\mathsf{ACC}^{(\ell)}$ *as the value of* ACC *after the sequence of operations:*

$$\mathsf{ACC} \leftarrow v^{(0)}; \qquad \textit{\textbf{for}}\ i = 1 \ldots \ell\ \textit{\textbf{do}}\ \mathsf{ACC} \overset{+}{\leftarrow} \mathbf{C}^{(i)}.$$

Set $\mathbf{D}^{(i)} = [\mathbf{D}_1 \ldots \mathbf{D}_{d_{\mathsf{g}}}]$ *to be the decomposition of* $u^{-1}\mathsf{ACC}^{(i)} = \sum_{j=1}^{d_{\mathsf{g}}} B_{\mathsf{g}}^{j-1}\mathbf{D}_j$*. Then, with overwhelming probability we have*

$$s_1\left(\left[\mathbf{D}^{(0)}, \mathbf{D}^{(1)} \ldots, \mathbf{D}^{(\ell-1)}\right]\right) = O(B_{\mathsf{g}}\sqrt{Nd_{\mathsf{g}} \cdot \ell}).$$

Proof. Because the spectral norm $s_1([\mathbf{D}^{(0)}, \mathbf{D}^{(1)} \ldots, \mathbf{D}^{(\ell-1)}])$ is efficiently computable from the $\mathbf{C}^{(i)}$'s, we can assume without loss of generality that the $\mathbf{C}^{(i)}$'s are truly uniformly random. We prove by induction on ℓ that

1. for $1 \leq i \leq \ell$, the $\mathbf{D}^{(i)}$'s follow independents uniform distributions in $\mathcal{R}_{[B_{\mathsf{g}}]}^{2d_{\mathsf{g}} \times 2d_{\mathsf{g}}}$ where $\mathcal{R}_{[B_{\mathsf{g}}]}$ is the set of polynomials with coefficients in $\{\frac{1-B_{\mathsf{g}}}{2} \ldots \frac{B_{\mathsf{g}}-1}{2}\}$.
2. for $0 \leq i \leq \ell$, $\mathbf{D}^{(i)}$ is invertible with overwhelming probability.

The implication 1. ⇒ 2. follows from Lemma 4. Indeed the uniform distribution over $\mathcal{R}_{[B_{\mathsf{g}}]}$ is still a uniform distribution when taken mod3 since 3 divides B_{g}. Note that $\mathbf{D}^{(0)} = Y^{v_0} \cdot \mathbf{I}_{2D}$ is invertible.

We may now start the induction and assume that $\mathbf{D}^{(\ell-1)}$ is invertible. It follows that $\mathsf{ACC}^{(\ell)} = \mathbf{D}^{(\ell-1)} \cdot \mathbf{C}^{(\ell)}$ is uniformly random in $\mathcal{R}_Q^{2d_{\mathsf{g}} \times D}$ and independent of all $\mathbf{D}^{(i)}$ for $i < \ell$. We conclude the induction using the fact that the decomposition step is a bijective map $\mathcal{R}_Q^{2d_{\mathsf{g}} \times 2} \to \mathcal{R}_{[B_{\mathsf{g}}]}^{2d_{\mathsf{g}} \times 2d_{\mathsf{g}}}$.

The coefficients of $\mathbf{D} = [\mathbf{D}^{(1)} \ldots, \mathbf{D}^{(\ell-1)}] \in \mathcal{R}^{2d_{\mathsf{g}} \times 2d_{\mathsf{g}}\ell}$ are independents subgaussian variables with parameter $O(B_{\mathsf{g}})$. It follows by lemma 1 that

$$s_1(\left[\mathbf{D}^{(0)}, \mathbf{D}^{(1)} \ldots, \mathbf{D}^{(\ell-1)}\right]) \leq O(B_{\mathsf{g}}\sqrt{Nd_{\mathsf{g}} \cdot \ell}).$$

Proof (of Lemma 11). Applying ℓ times Fact 9, we can show that $\mathsf{ACC}^{(\ell)}$ has the form

$$\mathsf{ACC}^{(\ell)} = [\mathbf{A}, \mathbf{A} \cdot z + \mathbf{e}] + uX^v\mathbf{G} \quad \text{with } \mathbf{e} = \sum_{i=1}^{\ell} \mathbf{D}^{(i-1)}\mathbf{e}^{(i)}$$

where $\mathbf{e}^{(i)}$ is the error used in the encryption $\mathbf{C}^{(i)} \leftarrow E_z(v^{(i)})$. The final error in $\mathbf{c}$ is $e = [\overrightarrow{0}^t, \mathbf{t}^t, \overrightarrow{0}^t, \ldots, \overrightarrow{0}^t] \cdot \overrightarrow{\mathbf{e}}$. We rewrite

$$e = \left[\overrightarrow{0}^t, \mathbf{t}^t, \overrightarrow{0}^t, \ldots, \overrightarrow{0}^t\right] \cdot \left[\mathbf{D}^{(0)}, \overset{\Rightarrow}{\ldots}, \mathbf{D}^{(\ell-1)}\right] \cdot \overrightarrow{\left(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(\ell)}\right)}.$$

Recall that $\|\mathbf{t}\| = \sqrt{q/2}$, and by Fact 12, we have that $[\mathbf{D}^{(0)}, \overset{\Rightarrow}{\ldots}, \mathbf{D}^{(\ell-1)}]$ has spectral norm $O(B_\mathrm{g}\sqrt{Nd_\mathrm{g} \cdot \ell})$. We can rewrite $e = \mathbf{v} \cdot \overrightarrow{(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(\ell)})}$ where $\|\mathbf{v}\| = O(B_\mathrm{g}\sqrt{q \cdot Nd_\mathrm{g} \cdot \ell})$ and $\overrightarrow{(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(\ell)})}$ is a subgaussian vector of parameter ς. We conclude that the final error is subgaussian of parameter $\beta = O(\varsigma B_\mathrm{g}\sqrt{qNd_\mathrm{g} \cdot \ell})$.

5.3 Efficient Accumulator Increment

To efficiently implement the accumulator increment Incr, one needs to keep the accumulator ACC, as well as the precomputed ciphertexts from the bootstrapping key, in FFT/NTT format.

Algorithm 3. $\mathsf{Incr}(\widehat{\mathsf{ACC}} \in \widehat{\mathcal{R}}^{2d_\mathrm{g} \times 2}, \widehat{\mathbf{C}} \in \widehat{\mathcal{R}}^{2d_\mathrm{g} \times 2})$

```
Compute $\mathsf{ACC} \leftarrow \mathsf{FFT}^{-1}(\widehat{\mathsf{ACC}})$
Decompose $u^{-1}\mathsf{ACC} = \sum_{i=1}^{d_\mathrm{g}} B_\mathrm{g}^{i-1}\mathbf{D}_i$, and set $\mathbf{D} = [\mathbf{D}_1 \ldots \mathbf{D}_{d_\mathrm{g}}] \in \mathcal{R}^{2d_\mathrm{g} \times 2d_\mathrm{g}}$
Compute $\widehat{\mathbf{D}} \leftarrow \mathsf{FFT}(\mathbf{D})$
Return $\widehat{\mathbf{D}} \odot \widehat{\mathbf{C}}$
```

Each increment requires $4d_\mathrm{g}$ backward FFT's and $4d_\mathrm{g}^2$ forward FFT's. If one uses the Number Theoretic Transform rather than the complex FFT, $4d_\mathrm{g}$ forward transforms can be traded for a few additions $\bmod Q$ by computing $\widehat{\mathbf{D}}_1 = u^{-1}\widehat{\mathsf{ACC}} - \sum_{i=2}^{d_\mathrm{g}} B_\mathrm{g}^{i-1} \cdot \widehat{\mathbf{D}}_i \bmod Q$.

5.4 Asymptotic Parameters and Efficiency

Secret Keys and Errors. We choose the secret key $\mathbf{s}$ of the LWE scheme to be binary in order to minimize the final error parameter $\mathcal{E}(nd)$ that depends on $\|\mathbf{s}\|$ (Theorem 10). The hardness of LWE for such a distribution of secrets was established in [7]. The randomized rounding used for errors in the switching key $\mathbf{k}_{i,j,v} \leftarrow \mathsf{LWE}_\mathbf{s}^{q/q}(v \cdot z_i \cdot d_\mathsf{ks}^j)$, is $\chi_\sigma(x) = D_{\mathbb{Z},x,\sigma}$, the discrete gaussian of standard deviation σ centered in x.

The secret $z \in \mathcal{R}$ of the Ring-GSW scheme follows the discrete gaussian distribution $\chi_\varsigma(0)$, and the errors follow the gaussian randomized rounding function χ_ς.

Parameters. For simplicity, we take the base $B_{\mathsf{g}}, B_{\mathsf{r}}, B_{\mathsf{ks}} = \Theta(1)$ to be fixed, which sets $d_{\mathsf{g}}, d_{\mathsf{r}}, d_{\mathsf{ks}} = O(\log n)$ provided that $q, Q = \text{poly}(n)$. Error parameters are set to $\sigma, \varsigma = \omega(\sqrt{\log n})$. For the dimension of the Ring-GSW scheme, we take $2N = q = \Theta(n)$. It remains to set $Q = n^2 \cdot \log n \cdot \omega(\log n)$, and we obtain a refreshing error $\mathcal{E}(nd) = O(n) \leq q/16$.

Efficiency and Comparison. The running time of the $\mathsf{Refresh}$ operation is dominated by dn homomorphic operations. For comparison, the scheme of [2] requires $dn \cdot O(\log^3 q/\log\log q)$ homomorphic operations.

In practice this polylogarithmic is far from negligible, *e.g.* $q = 2\cdot3\cdot5\cdot7\cdot11 = 2310$ gives a factor $2^2+3^2+5^2+7^2+11^2 = 208$. Memory usage is also decreased by a factor $O(\log^2 q/\log\log q)$, that is a factor 28 in our previous example.

Also, we do not rely on randomized decomposition for the increment operation $\mathsf{ACC} \overset{+}{\leftarrow} \mathbf{C}$. While this randomization is asymptotically less expensive than the FFT step by a factor $\log N$, avoiding it makes the implementation simpler and potentially faster considering the cost of randomness in practice.

Finally, our $\mathsf{Refresh}$ procedure (before key and modulus switching) produces a ciphertext with subgaussian error of parameter $\alpha = O(n^2 \log n)$ in our scheme against $\alpha = \Theta(n^{5/2}\log^3 n/\log\log n)$ in [2].

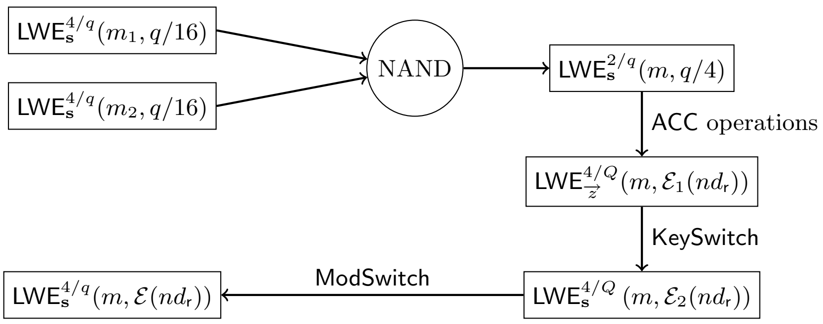
$\mathsf{LWE}_{\mathbf{s}}^{4/q}(m_1, q/16)$
$\mathsf{LWE}_{\mathbf{s}}^{4/q}(m_2, q/16)$
NAND
$\mathsf{LWE}_{\mathbf{s}}^{2/q}(m, q/4)$
ACC operations
$\mathsf{LWE}_{\vec{z}}^{4/Q}(m, \mathcal{E}_1(nd_{\mathsf{r}}))$
KeySwitch
$\mathsf{LWE}_{\mathbf{s}}^{4/Q}(m, \mathcal{E}_2(nd_{\mathsf{r}}))$
ModSwitch
$\mathsf{LWE}_{\mathbf{s}}^{4/q}(m, \mathcal{E}(nd_{\mathsf{r}}))$

Fig. 1. Cycle for a simple NAND gate, using the Homomorphic property of Section 3

6 Parameters, Implementation and Benchmark

We start by presenting the methodology to evaluate the security of our scheme in Section 6.1, propose parameters in Section 6.2, discuss FFT implementation details in Section 6.3 and conclude with the benchmarks in Section 6.4.

6.1 Security Estimation

The security estimate methodology follows the analysis of [27]. To build an ϵ-distinguisher against LWE in dimension n, modulus q and a randomized rounding function χ of standard deviation σ, Lindner and Peikert estimate that the best known attack by lattice reduction requires to achieve a *root Hermite factor* of

$$\delta = \delta\text{-}\mathsf{LWE}(n, q, \sigma, \epsilon) = 2^{(\log_2^2 \rho)/(4n \log_2 q)} \quad \text{where } \rho = (q/\sigma) \cdot \sqrt{2 \ln(1/\varepsilon)} \quad (5)$$

To estimate the security of $\mathsf{binLWE}_{n,q,\sigma}$, going through the security reduction of [7] would be a very pessimistic approach. Still, $\mathsf{binLWE}_{n,q,\sigma}$ doesn't enjoy as much concrete security as $\mathsf{LWE}_{n,q,\sigma}$. Indeed, binary secrets allow an attacker to switch to a smaller modulus q' without affecting the relative error $1/\rho$ much (which is actually the property we exploit for the correctness of our scheme). Indeed, switching from modulus q to q', one obtains essentially binLWE samples with errors parameter $\sigma' = \sqrt{(q'/q)^2\sigma^2 + \|\mathbf{s}\|/12} \approx \sigma q'/q$, following Lemma 5. For comparison, such modulus switch on usual LWE produces errors of parameter $\sigma' = \sqrt{(q'/q)^2\sigma^2 + \sigma^2 O(n)} \approx \sigma\sqrt{n}$.

In light of this attack, we compute the root Hermite factor for binLWE as follows:

$$\delta\text{-}\mathsf{binLWE}(n, q, \sigma, \epsilon) = \min_{q' \leq q} \delta\text{-}\mathsf{LWE}(n, q', \sigma' = \sqrt{(q'/q)^2\sigma^2 + n/24}, \epsilon). \quad (6)$$

Such minimum will be computed using standard numerical analysis tools for the security estimation of our set of parameters below.

6.2 Proposed Parameters

Relaxed Constraints on B_{g} and Q. In practice we will ignore the constraints of the correctness statement (Theorem 10) that B_{g} is a power of 3 and Q is a power of B_{g}. Those constraints are artifact of our proofs, we will only require that $B_{\mathsf{g}}^{d_{\mathsf{g}}} \geq Q$. We have verified that in practice this relaxation does not signficantly affects the distribution of $\mathsf{err}(\mathsf{Refresh}(\mathbf{c}))$.

Accumulated Errors. According to the central limit heuristic, the final error $\mathsf{err}(\mathsf{Refresh}(\mathbf{c}))$ of a refreshed ciphertext behaves as a Gaussian of standard deviation:

$$\beta = \sqrt{\frac{q^2}{Q^2}\left(\varsigma^2 \cdot \frac{B_{\mathsf{r}}^2}{12} \cdot nd_{\mathsf{r}} \cdot \frac{q}{2} \cdot 2Nd' + \sigma^2 N d_{\mathsf{ks}}\right) + \frac{\|\mathbf{s}\|^2 + 1}{12}}.$$

The factors $\frac{1}{12}$ follows from the fact that a uniform random variable in $[-\frac{1}{2}, \frac{1}{2}]$ has variance $\frac{1}{12}$.

The factor $2d'$ (instead of $2d_{\mathsf{g}}$) takes account that the final coordinate of a decomposition of an element mod$Q$ over base B_{g} is bounded by $Q/2B_{\mathsf{g}}^{d_{\mathsf{g}}}$ rather than $B_{\mathsf{g}}/2$. Therefore we set $d' = B_{\mathsf{g}} - 1 + Q/B_{\mathsf{g}}^{d_{\mathsf{g}}}$ (in the following parameters we have $d' = 2.5$ instead of $d_{\mathsf{g}} = 3$).

Additionally, we assume that $\|\mathbf{s}\| \leq n/2$, which is true for half of the random secrets $\mathbf{s} \in \{0,1\}^n$. If not, one may simply discard this $\mathbf{s}$ during key generation and resample a fresh secret key. To thwart an attack that would shift all coordinates of $\mathbf{s}$ by $-1/2$, we also randomize the signs of each entry of $\mathbf{s}$, which intuitively, can only increase the security (and does not affect the error analysis).

We evaluate the error probability as the probability that two independently refreshed ciphertexts $\mathbf{c}_1, \mathbf{c}_2$ verify $|\mathsf{err}(\mathbf{c}_1)+\mathsf{err}(\mathbf{c}_2)| < q/8$, which is sufficient but looser than $|\mathsf{err}(\mathbf{c}_i)| < q/16$.

Parameters.

LWE parameters:	$n = 500 \quad Q = 2^{32}, \sigma = 2^{17}, \quad q = 2^9.$
Ring-GSW parameters:	$N = 2^{10}, \ \varsigma = 1.4.$
Gadget Matrix:	$B_{\mathsf{g}} = 2^{11}, d_{\mathsf{g}} = 3, \quad u = \frac{Q}{8} + 1.$
Bootstrapping Key parameters:	$B_{\mathsf{r}} = 23, \ d_{\mathsf{r}} = 2.$
Key Switching Key parameters:	$B_{\mathsf{ks}} = 24, \ d_{\mathsf{ks}} = 7.$

Efficiency.

Bootstrapping Key Size:	$4nNd_{\mathsf{r}}B_{\mathsf{r}}d_{\mathsf{g}} \log_2 Q$ bits	= 1032 MBytes.
Key Switching Key Size:	$nNB_{\mathsf{ks}}d_{\mathsf{ks}} \log_2 Q$ bits	= 314 MBytes.
FFTs per NAND gate:	$4nd_{\mathsf{r}}d_{\mathsf{g}}(d_{\mathsf{g}}+1)$	$= 48,000$ FFTs.

Correctness.

Final error parameter: $\beta = 6.94.$

Pr. of error per NAND: $p = 1 - \mathrm{erf}(r/\sqrt{2}) \leq 2^{-31}$ where $r = \frac{q/8}{\sqrt{2}\beta}$.

The error probability can be brought down to 2^{-45} by applying the HomNAND operation *before* KeySwitch and ModSwitch.

Security.

Security of the LWE scheme	$\delta\text{-}\mathsf{binLWE}(n, Q, \sigma, 2^{-64})$	= 1.0064.
Security of the Ring-GSW scheme	$\delta\text{-}\mathsf{LWE}(N, Q, \varsigma, 2^{-64})$	= 1.0064.

The security of the Ring-GSW scheme is evaluated ignoring the ring structure, since there are yet no known algorithms that exploit such structure.

According to the predictions of [10], the BKZ algorithm requires a block size greater than 190 to reach a root hermite factor of 1.0065. For such block size, each of the many calls to the enumeration routine would visit more than 2^{100} nodes of the pruned enumeration tree. This is to be considered as a preliminary security analysis, demonstrating the feasibility of our construction. For a more precise security analysis, one should include the more involved results of Liu and Nguyen [28], and any new advances on lattice cryptanalysis.

6.3 FFT Implementation

To avoid implementation technicalities related to working in a prime field $\mathbb{F}_Q$ and potentially expensive reduction mod$Q$, we choose to rely on the complex

FFT rather than the Number Theoretic Transform, that is, we use the complex primitive $2N$-th root of unity $\omega = \exp(2\pi\imath/2N)$ rather than a primitive root in $\mathbb{F}_Q$. This allows us to rely on a flexible and optimized library for FFT, namely, *the Fastest Fourier Transform in the West* [13] and choose Q as a power of two, essentially offering reductions mod$Q$ for free.

Technically, one wishes to compute the so-called negacyclic-FFT of rank N, which can be extracted from the FFT in rank $2N$ by only keeping the odd indexes of the result. Nevertheless, a factor 2 is saved considering that we are computing FFT on real-data.

Due to vectorized instructions, this implementation of FFT at double-precision reaches up to 6 Gflops on a single 64-bits Intel Core running at 3 Ghz. We measure a running time of 10 microseconds per FFT at dimension $2N = 2048$; which fits the predictions[2]. While it is unclear if either the choice of FFT over NTT is optimal, or if this particular implementation is, this prototype is enough to support our claim.

Precision Issues. One crucial question when using complex FFT is the precision requirement. In our case (see Section 5.3), FFT is used to multiply two integer polynomials with coefficients in $\mathbb{Z}_Q$, yet one of them is guaranteed to have coefficients smaller than $B_{\mathsf{g}}/2$. Without reduction mod$Q$, the resulting product is expected to have coefficients of size $S = B_{\mathsf{g}}Q\sqrt{N}/4$. The final result is guaranteed to be correct if the final relative error ϵ verifies $S\epsilon \leq 1/2$. For our set of parameters, we have $S = 2^{46}$.

Asymptotically, the relative error growth during FFT is known to be $O(\log N)$ in the worst case and $O(\sqrt{\log N})$ on average [14,33]. In practice, at double precision ($\epsilon_0 = 2^{-54}$ relative error for each operation) FFTW [13] in rank $2N = 2048$ is reported[3] to produce errors of standard deviation $\epsilon = 2^{-52}$ (which match $\approx \epsilon_0 \cdot \sqrt{\log N}$). It seems barely sufficient to ensure perfect correctness of each computation of a products of polynomials. Yet, if small errors are introduced by floating-point approximations, this doesn't necessary breaks the correctness of the scheme. Indeed, this errors can simply be considered as a small extra error term introduced at each operation on the accumulator.

A formal claim would require a more detailed study. The fact that our implementation works in practice, and that the measurements of errors fit our prediction is sufficient for our purpose.

6.4 Benchmark and Source Code

Our implementation performs a $\mathsf{HomNAND}$ and a $\mathsf{Refresh}$ operation every 0.69 seconds on a single 64-bits Intel core at 3GHz, which conforms to our prediction of 0.5 seconds from the count of FFT operations (the key switching step is having non negligible cost because it hasn't been vectorized yet). It consumes 2.2Gbytes of memory, which is approximately twice the prediction. This is explained by the

[2] http://www.fftw.org/speed/
[3] http://www.fftw.org/accuracy/

fact that for efficiency, the Bootstrapping Key is stored in FFT form, at `double` precision.

We can expect those performance figures to be improved by further implementation efforts. Yet, our prototype implementation already performs within one order of magnitude of the *amortized* cost of bootstrapping in HElib [23]. A more precise comparison is hard to state considering our scheme has a different security parameters, and does not offers the same set of gates. Sophisticated benchmarking would not be very useful until this new scheme is optimized and generalized to reach its full potential.

The source code is reasonably concise and simple, consisting of about 600 lines of `C++` code, excluding the library FFTW. It is available on `github` [11].

7 Extensions, Conclusions and Future Work

We have shown that a complete bootstrappable homomorphic computation can be performed in a fraction of a second, much faster than any previous solution. We achieved the result by addressing the simplest form of bootstrappable computation (the computation of a single binary gate that is complete for boolean circuits), and introducing new techniques for this homomorphic computation. We remark that the techniques presented in the paper are not limited to NAND gates. For example, it is immediate to extend our solution to compute a majority gate that on input 3 bits $x_1, x_2, x_3 \in \{0,1\}$, outputs 1 if at least two of the inputs are 1, and 0 if at least two of the inputs are zero. To see this, recall that our solution to the NAND problem resorted to viewing bits are integers modulo $t = 4$, and then encoding the NAND operation in terms of addition. Still using arithmetic modulo 4, one can compute the majority of x_1, x_2, x_3 by taking the sum $y = x_1 + x_2 + x_3 \in \{0,1,2,3\}$, and checking if the result is at least 2. The final test is easily performed by applying our most significant bit extraction procedure to the shifted sum $y - 0.5$. As we are adding three input ciphertexts, this may require slighly smaller noise, but the computation is almost identical to the NAND gate described in this paper.

This can be further generalized to (weigthed) threshold gates $\sum_i w_i x_i > h$, where the number of inputs, weigths w_i and the threshold h are arbitrary, by using arithmetic modulo a larger $t > 2 \sum |w_i|$.

Further generalizations are possible by replacing our msbExtract procedure with a more complex test that checks membership for many subsets of $\mathbb{Z}_t$. Precisely, membership test may be extended to any anti-symmetric set $\mathcal{S} \subset \mathbb{Z}_t$ ($x \in \mathcal{S} \Leftrightarrow x + \frac{t}{2} \notin \mathcal{S}$). For example, with $t = 6$ arbitrary large xor's $x_1 \oplus \ldots \oplus x_k$ can be performed in just one Refresh operation using the membership test $x_1 + \ldots + x_k \bmod 6 \in \{1,3,5\}$. With this generalization, our technique also offers *xor-for-almost-free*, as in previous FHE schemes.

Additionally, taking weighted linear combinations of k input bits $\sum_i 2^i x_i$, and checking membership in subsets of $\mathbb{Z}_{2^{k+2}}$, one can (at least in principle) implement arbitrary boolean gates (adders, S-boxes, etc.), but the complexity grows exponentially in the number of inputs k.

We also remark that since the membership test is much less expensive than the rest of the Refresh procedure, one may test several function of the same input for almost free. In other words, gates with several outputs would not be much more expensive than gates with only one output. For $t = 6$, this already allows to perform an *add-with-carry* gate (3 inputs, 2 outputs) in a single shot (instead of 5 using binary gates).

Fully exploring the use of our techniques to realize more complex gates is left to future work. Other interesting open problems are finding ways to fully exploit the message space offered by ring LWE encryption in our accumulator implementation, and combining our framework with the CRT techniques of [2].

Acknowledgments. The authors wish to thank Igors Stepanovs for interesting conversations about circular security that lead us to the new homomorphic NAND procedure, as well as the anonymous EUROCRYPT'15 reviewers for their careful reading of the paper and their diligent comments.

References

1. Alperin-Sheriff, J., Peikert, C.: Practical bootstrapping in quasilinear time. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 1–20. Springer, Heidelberg (2013)
2. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 297–314. Springer, Heidelberg (2014)
3. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
4. Blum, A., Furst, M.L., Kearns, M., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer, Heidelberg (1994)
5. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
6. Brakerski, Z., Gentry, C., Halevi, S.: Packed ciphertexts in LWE-based homomorphic encryption. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 1–13. Springer, Heidelberg (2013)
7. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Boneh, D., Roughgarden, T., Feigenbaum, J., (eds.), 45th ACM STOC, pp. 575–584. ACM Press, June 2013
8. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Ostrovsky, R. (ed.) 52nd FOCS, pp. 97–106. IEEE Computer Society Press, October 2011
9. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
10. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)

11. Ducas, L., Micciancio, D.: Implementation of FHEW (2014). https://github.com/lducas/FHEW
12. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 335–352. Springer, Heidelberg (2014)
13. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proceedings of the IEEE **93**(2), 216–231 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation"
14. Gentleman, W.M., Sande, G.: Fast fourier transforms: For fun and profit. In: Proceedings of the November 7–10, 1966, Fall Joint Computer Conference, AFIPS 1966 (Fall), pp. 563–578. ACM, New York, NY, USA (1966)
15. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC, pp. 169–178. ACM Press, May / June 2009
16. Gentry, C., Halevi, S.: Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In: Ostrovsky, R. (ed.) 52nd FOCS, pp. 107–109. IEEE Computer Society Press, October 2011
17. Gentry, C., Halevi, S.: Implementing Gentry's fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
18. Gentry, C., Halevi, S., Peikert, C., Smart, N.P.: Ring switching in BGV-style homomorphic encryption. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 19–37. Springer, Heidelberg (2012)
19. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 1–16. Springer, Heidelberg (2012)
20. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
21. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
22. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)
23. Halevi, S., Shoup, V.: Algorithms in HElib. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 554–571. Springer, Heidelberg (2014)
24. Halevi, S., Shoup, V.: Bootstrapping for HElib. IACR Cryptology ePrint Archive, (2014). http://eprint.iacr.org/2014/873
25. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
26. Lidl, R., Niederreiter, H.: Finite Fields. Reading, MA: Addison-Wesley. Encyclopedia of Mathematics and its Applications **20** (1983)
27. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
28. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: An update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013)

29. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
30. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer, Heidelberg (2013)
31. Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In: 43rd FOCS, pp. 356–365. IEEE Computer Society Press, November 2002
32. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R., (eds.), 37th ACM STOC, pp. 84–93. ACM Press, May 2005
33. Schatzman, J.C.: Accuracy of the discrete fourier transform and the fast fourier transform. SIAM J. Sci. Comput. **17**(5), 1150–1166 (1996)