Better Bootstrapping in Fully Homomorphic Encryption

Craig Gentry, IBM
Shai Halevi, IBM
Nigel P. Smart, University of Bristol

December 15, 2011

Abstract

Gentry's bootstrapping technique is currently the only known method of obtaining a "pure" fully homomorphic encryption (FHE) schemes, and it may offers performance advantages even in cases that do not require pure FHE (such as when using the new noise-control technique of Brakerski-Gentry-Vaikuntanathan).

The main bottleneck in bootstrapping is the need to evaluate homomorphically the reduction of one integer modulo another. This is typically done by emulating a binary modular reduction circuit, using bit operations on binary representation of integers. We present a simpler approach that bypasses the homomorphic modular-reduction bottleneck to some extent, by working with a modulus very close to a power of two. Our method is easier to describe and implement than the generic binary circuit approach, and is likely to be faster in practice. In some cases it also allows us to store the encryption of the secret key as a single ciphertext, thus reducing the size of the public key.

We also show how to combine our new method with the SIMD homomorphic computation techniques of Smart-Vercauteren and Gentry-Halevi-Smart, to get a bootstrapping method that works in time quasi-linear in the security parameter. This last part requires extending the techniques from prior work to handle arithmetic not only over fields, but also over some rings. (Specifically, our method uses arithmetic modulo a power of two, rather than over characteristic-two fields.)

Keywords. Fully Homomorphic Encryption, Bootstrapping

Acknowledgments. The first and second authors are sponsored by DARPA and ONR under agreement number N00014-11C-0390. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

The third author is sponsored by DARPA and AFRL under agreement number FA8750-11-2-0079. The same disclaimers as above apply. He is also supported by the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II and via an ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO, by EPSRC via grant COED–EP/I03126X, and by a Royal Society Wolfson Merit Award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the European Commission or EPSRC.

Contents

1 Introduction

Fully Homomorphic Encryption (FHE) [14, 10] is a powerful technique to enable a party to compute an arbitrary function on a set of encrypted inputs; and hence obtain the encryption of the function's output. Starting from Gentry's breakthrough result [9, 10], all known FHE schemes are constructed from *Somewhat Homomorphic Encryption* (SWHE) schemes, that can only evaluate functions of bounded complexity. The ciphertexts in these SWHE schemes include some "noise" to ensure security, and this noise grows when applying homomorphic operations until it becomes so large that it overwhelms the decryption algorithm and causes decryption errors. To overcome the growth of noise, Gentry used a *bootstrapping* transformation, where the decryption procedure is run homomorphically on a given ciphertext, using an encryption of the secret key that can be found in the public key,[1] resulting in a new ciphertext that encrypts the same message but has potentially smaller noise.

Over the last two years there has been a considerable amount of work on developing new constructions and optimizations [8, 16, 12, 5, 17, 4, 11, 3, 13], but all of these constructions still have noise that keeps growing and must be reduced before it overwhelms the decryption procedure. The techniques of Brakerski et al. [3] yield SWHE schemes where the noise grows slower, only linearly with the depth of the circuit being evaluated, but for any fixed public key one can still only evaluate circuits of fixed depth. The only way to get "pure" FHE that can evaluate arbitrary functions with a fixed public key is by using bootstrapping. Also, bootstrapping can be used in conjunction with the techniques from [3] to get better parameters (and hence faster homomorphic evaluation), as described in [3, 13].

In nearly all SWHE schemes in the literature that support bootstrapping, decryption is computed by evaluating some ciphertext-dependent linear operation on the secret key, then reducing the result modulo a public odd modulus q into the range $(-q/2, q/2]$, and then taking the least significant bit of the result. Namely, denoting reduction modulo q by $[\cdot]_q$, we decrypt a ciphertext c by computing $a = [[L_c(s)]_q]_2$ where L_c is a linear function and s is the secret key. Given an encryption of the secret key s, computing an encryption of $L_c(s)$ is straightforward, and the bulk of the work in homomorphic decryption is devoted to reducing the result modulo q. This is usually done by computing encryptions of the bits in the binary representation of $L_c(s)$ and then emulating the binary circuit that reduces modulo q.

The starting point of this work is the observation that when q is very close to a power of two, the decryption procedure takes a particularly simple form. Specifically, we can compute the linear function $L_c(s)$ modulo a power of two, and then XOR the top and bottom bits of the result. We then explain how to implement this simple decryption formula homomorphically, and also how the techniques of Gentry et al. from [13] can be used to compute this homomorphic decryption with only polylogarithmic overhead. We note that the techniques from [13] can in principle be applied to any function, but applying them to homomorphic decryption is not straightforward since the input and output are not presented in the right form. Also, for our case we need to extend the results from [13] slightly, since we are computing a function over a ring (modulo a power of two) and not over a field.

We point out that in all work prior to [13], bootstrapping required adding to the public key many ciphertexts, encrypting the individual bits (or coefficients) of the secret key. This resulted in very large public keys, of size at least $\lambda^2 \cdot \text{polylog}(\lambda)$ (where λ is the security parameter). Using the techniques from [17, 3, 13], it is possible to encrypt the secret key in a "packed" form, hence reducing the number of ciphertexts to $O(\log \lambda)$ (so we can get public keys of size quasi-linear in λ). Using our technique from this work, it is even possible to store an encryption of the secret key as a single ciphertext, as described in Section 5. We next outline our main bootstrapping technique in a few more details.

[1]This transformation relies on the underlying SWHE being circularly secure.

Our method applies mainly to "leveled" schemes that use the noise control mechanism of Brakerski-Gentry-Vaikuntanathan [3].[2] Below and throughout this paper we concentrate on the BGV ring-LWE-based scheme, since it offers the most efficient homomorphic operations and the most room for optimizations.[3] The scheme is defined over a ring $R = \mathbb{Z}[X]/F(X)$ for a monic, irreducible polynomial $F(X)$ (over the integers $\mathbb{Z}$). For an arbitrary integer modulus n (not necessarily prime) we denote the ring $R_n \stackrel{\mathrm{def}}{=} R/nR = (\mathbb{Z}/n\mathbb{Z})[X]/F(X)$. The scheme is parametrized by the number of levels that it can handle, which we denote by L, and by a set of decreasing odd moduli $q_0 \gg q_1 \gg \cdots \gg q_L$, one for each level.

The plaintext space is given by the ring R_2, while the ciphertext space for the i'th level consists of 2-vectors over R_{q_i}. Secret keys are polynomials $\mathfrak{s} \in R$ with "small" coefficients, and we view $\mathfrak{s}$ as the second element of the 2-vector $\mathbf{s} = (1, \mathfrak{s})$. A level-$i$ ciphertext $\mathbf{c} = (c_0, c_1)$ encrypts a plaintext polynomial $m \in R_2$ with respect to $\mathbf{s} = (1, \mathfrak{s})$ if we have the equality over R, $[\langle \mathbf{c}, \mathbf{s} \rangle]_{q_i} = [c_0 + \mathfrak{s} \cdot c_1]_{q_i} \equiv m \pmod 2$, and moreover the polynomial $[c_0 + \mathfrak{s} \cdot c_1]_{q_i}$ is "small", i.e. all its coefficients are considerably smaller than q_i. Roughly, that polynomial is considered the "noise" in the ciphertext, and its coefficients grow as homomorphic operations are performed. The crux of the noise-control technique from [3] is that a level-i ciphertext can be publicly converted into a level-$(i+1)$ ciphertext (with respect to the same secret key), and that this transformation reduces the noise in the ciphertext roughly by a factor of q_{i+1}/q_i.

Secret keys too are associated with levels, and the public key includes some additional information that (roughly speaking) makes it possible to convert a ciphertext with respect to level-i key $\mathbf{s}_i$ into a ciphertext with respect to level-$(i+1)$ key $\mathbf{s}_{i+1}$. In what follows we will only be interested in the secret keys at level L and level zero; which we will denote by $\mathbf{s}$ and $\tilde{\mathbf{s}}$ respectively to ease notation.

For bootstrapping, we have as input a level-L ciphertext (i.e. a vector $\mathbf{c} \in R/q_L R$ modulo the smallest modulus q_L). This means that the noise-control technique can no longer be applied to reduce the noise, hence (essentially) no more homomorphic operations can be performed on this ciphertext. To enable further computation, we must therefore "recrypt" the ciphertext $\mathbf{c}$, to obtain a new ciphertext that encrypts the same element of R with respect to some lower level $i < L$.

Our first observation is that the decryption at level L can be made more efficient when q_L is close to a power of two, specifically $q_L = 2^r + 1$ for an integer r, and moreover the coefficients of $Z = \langle \mathbf{c}, \mathbf{s} \rangle \bmod F(X)$ are much smaller than q_L^2 in magnitude. In particular if z is one of the coefficients of the polynomial Z then $[[z]_{q_L}]_2$ can be computed as $z\langle r \rangle \oplus z\langle 0 \rangle$, where $z\langle i \rangle$ is the i'th bit of z.

To evaluate the decryption formula homomorphically, we temporarily extend the plaintext space to polynomials modulo 2^{r+1} (rather than modulo 2). The level-L secret key is $\mathbf{s} = (1, \mathfrak{s})$, where all the coefficients of $\mathfrak{s}$ are small (in the interval $(-2^r, +2^r)$). We can therefore consider $\mathfrak{s}$ as a plaintext polynomial in $R/2^{r+1}R$, encrypt it inside a level-0 ciphertext, and keep that ciphertext in the public key. Thus, given the level-L ciphertext $\mathbf{c}$, we can evaluate the inner product $[\langle \mathbf{c}, \mathbf{s} \rangle \bmod F(X)]$ homomorphically, obtaining a level-0 ciphertext that encrypts the polynomial Z.

For simplicity, assume for now that what we get is an encryption of all the coefficients of Z separately. Given an encryption of a coefficient z of Z (which is an element in $\mathbb{Z}/2^{r+1}\mathbb{Z}$) we show how to extract (encryptions of) the zero'th and r'th bit using a data-oblivious algorithm. Hence we can finally recover a new ciphertext, encrypting the same binary polynomial at a lower level $i < L$.

To achieve efficient bootstrapping, we exploit the ability to perform operations on elements modulo 2^{r+1} in a SIMD fashion; much like in prior work [17, 3, 13]. Some care must be taken when applying these

[2]Our method can be used also with other schemes, as long as the scheme allows us to choose a modulus very close to a power of two. For example they can be used with the schemes from [5, 4].

[3]Our description of the BGV cryptosystem below assumes modulo-2 plaintext arithmetic, generalizing to modulo-p arithmetic for other primes $p > 2$ is straightforward.

techniques in our case, since the inputs and outputs of the bootstrapping procedure are not in the correct format: Specifically, these techniques require that inputs and outputs be in CRT representation, whereas decryption (and therefore recryption) inherently deals with polynomials in coefficient representation. We therefore must use explicit conversion to CRT representation, and ensure that these conversions are efficient enough. See details in Section 5.

Also, the techniques from prior work must be extended somewhat to be usable in our case: Prior work demonstrated that SIMD operations can be performed homomorphically when the underlying arithmetic is over a field, but in our case we have operations over the ring $\mathbb{Z}/2^{r+1}\mathbb{Z}$, which is not a field. The algebra needed to extend the SIMD techniques to this case is essentially an application of the theory of local fields [6]. We prove many of the basic results that we need in Section 4, and refer the reader to [6] for a general introduction and more details.

Notations

Throughout the paper we denote by $[z]_q$ the reduction of z mod q into the interval $(-\frac{q}{2}, \frac{q}{2}]$. We also denote the i'th bit in the binary representation of the integer z by $z\langle i\rangle$. Similarly, when a is an integer polynomial of degree d with coefficients $(a_0, a_1, \ldots, a_d)$, we denote by $a\langle i\rangle$ the 0-1 degree-d polynomial whose coefficients are all the i'th bits $(a_0\langle i\rangle, a_1\langle i\rangle, \ldots, a_d\langle i\rangle)$. If $\mathbf{c}, \mathbf{s}$ are two same-dimension vectors, then $\langle \mathbf{c}, \mathbf{s}\rangle$ denotes their inner product.

Organization

We present the simplified decryption formula in Section 2 and explain how to evaluate it homomorphically in Section 3. Then we recall some algebra in Section 4, and in Section 5 we explain how to use techniques similar to [13] to run bootstrapping in time quasi-linear in the security parameter. Some further optimizations are described in Section 6, and all the proofs are deferred to Appendix A.

2 A simpler decryption formula

Our first observation is that if the small modulus q_L has a special form – in particular, if it equals $u \cdot 2^r + v$ for some integer r and for some small positive odd integers u, v – then the mod-q_L decryption formula can be made to have a particularly simple form. Below, we will focus on the case of $q_L = 2^r + 1$, which suffices for our purposes.

So, assume that $q_L = 2^r + 1$ for some integer r and that we decrypt by setting $a \leftarrow [[\langle \mathbf{c}, \mathbf{s}\rangle \bmod F(X)]_{q_L}]_2$. Consider now the coefficients of the integer polynomial $Z = \langle \mathbf{c}, \mathbf{s}\rangle \bmod F(X)$, without the reduction mod q_L. Since $\mathbf{s}$ has small coefficients (and we assume that reduction mod-$F(X)$ does not increase the coefficients by much) then all the coefficients of Z are much smaller than q_L^2. Consider one of these integer coefficients, denoted by z, so we know that $|z| \ll q_L^2 \approx 2^{2r}$. We consider the binary representation of z as a $2r$-bit integer, and assume for now that $z \geq 0$ and also $[z]_{q_L} \geq 0$. We claim that in this case, the bit $[[z]_{q_L}]_2$ can be computed simply as the sum of the lowest bit and the r'th bit of z, i.e., $[[z]_{q_L}]_2 = z\langle r\rangle \oplus z\langle 0\rangle$. (Recall that $z\langle i\rangle$ is the i'th bit of z.)

Lemma 1. *Let $q = 2^r + 1$ for a positive integer r, and let z be a non-negative integer smaller than $\frac{q^2}{2} - q$, such that $[z]_q$ is also non-negative, $[z]_q \in [0, \frac{q}{2}]$. Then $[[z]_q]_2 = z\langle r\rangle \oplus z\langle 0\rangle$.*

The proof of Lemma 1 is in Appendix A. We note that the proof can easily be extended for the case $q = u2^r + v$, if that the bound on z is stengthen by a factor of v. To remove the assumption that both z and $[z]_q$ are non-negative, we use the following easy corollary (whose proof is also in Appendix A):

Corollary 1. *Let $r \geq 3$ and $q = 2^r + 1$ and let z be an integer with absolute value smaller than $\frac{q^2}{4} - q$, such that $[z]_q \in (-\frac{q}{4}, \frac{q}{4})$. Denote $z' = z + (q^2 - 1)/4$, then $[[z]_q]_2 = z'\langle r\rangle \oplus z'\langle 0\rangle$.*

Using Corollary 1 we can get our simplified decryption formula. First, we set our parameters such that $q_L = 2^r + 1$ and all the coefficients of the integer polynomial $Z = \langle \mathbf{c}, \mathbf{s}\rangle \bmod F(X)$ are smaller than $\frac{q_L^2}{4} - 1$ in absolute value, and moreover they are all less than $\frac{q_L - 1}{4}$ away from a multiple of q_L. Let $c^* = \mathbf{1} \cdot \frac{q_L^2 - 1}{4}$ be the fixed integer polynomial that has all of its coefficients equal to the integer $\frac{q_L^2-1}{4}$. Given a two-element ciphertext $\mathbf{c} = (c_0, c_1) \in ((\mathbb{Z}/q_L\mathbb{Z})[X]/F(X))^2$, we compute a new ciphertext $\mathbf{c}' = (c_0 + c^*, c_1)$, and next we decrypt $\mathbf{c}'$ using our new formula.

Specifically, since the first entry in $\mathbf{s}$ is 1, we have $Z' = \langle \mathbf{c}', \mathbf{s}\rangle = \langle \mathbf{c}, \mathbf{s}\rangle + c^* = Z + c^*$, over $\mathbb{Z}[X]/F(X)$ (without reduction modulo q_L). Hence for every coefficient z' in Z', the corresponding coefficient in Z is $z = z' + \frac{q_L^2-1}{4}$, and we can use Corollary 1. Putting it all together, to decrypt $\mathbf{c}$ we set $\mathbf{c}' = (c_0 + c^*, c_1)$, then compute $Z' \leftarrow \langle \mathbf{c}', \mathbf{s}\rangle \bmod F(X)$ over the integers (without reduction mod q_L), and finally recover the plaintext as $Z'\langle r\rangle + Z'\langle 0\rangle$. Ultimately, we obtain the plaintext polynomial $a \in \mathbb{F}_2[X]/F(X)$, where each coefficient in a is obtained as the XOR of bits 0 and r of the corresponding coefficient in Z'.

2.1 Working modulo 2^{r+1}

Since we are only interested in the contents of bit positions 0 and r in the polynomial Z', we can compute Z' modulo 2^{r+1} rather than over the integers. Observing that when $q_L = 2^r + 1$ then $\frac{q_L^2-1}{4} \equiv 2^{r-1} \pmod{2^{r+1}}$, our simplified decryption of a ciphertext vector $\mathbf{c} = (c_0, c_1)$ proceeds as follows (denoting $c^* = \mathbf{1} \cdot 2^{r-1}$):

0. Before Recryption, we post-process the ciphertext to get $\mathbf{c}' \leftarrow ([c_0 + c^*]_{2^{r+1}}, [c_1]_{2^{r+1}})$;

1. Compute $Z' \leftarrow [\langle \mathbf{c}', \mathbf{s}\rangle \bmod F(X)]_{2^{r+1}}$;

2. Recover the 0-1 plaintext polynomial $a = [Z'\langle r\rangle + Z'\langle 0\rangle]_2$.

(Note that when reducing modulo a power of 2, the bit representation is the same when reducing into the interval $[-2^r, 2^r - 1]$ as when reducing into the interval $[0, 2^{r+1} - 1]$. We can then interpret the reduction mod 2^{r+1} as done into either interval.)

3 Basic Homomorphic Decryption

To get a homomorphic implementation of the simplified decryption formula from above, we use an instance of our homomorphic encryption scheme with underlying plaintext space $\mathbb{Z}_{2^{r+1}}$. Namely, denoting by $\tilde{\mathbf{s}}$ the level-0 secret-key and by q_0 the largest modulus, a ciphertext encrypting $a \in (\mathbb{Z}/2^{r+1}\mathbb{Z})[X]/F(X)$ with respect to $\tilde{\mathbf{s}}$ and q_0 is a 2-vector $\tilde{\mathbf{c}}$ over $(\mathbb{Z}/q_0\mathbb{Z})[X]/F(X)$ such that $|[\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}\rangle \bmod F(X)]_{q_0}| \ll q_0$ and $[\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}\rangle \bmod F(X)]_{q_0} \equiv a \pmod{2^{r+1}}$.

Recall that the ciphertext before bootstrapping is with respect to secret key $\mathbf{s}$ and modulus $q_L = 2^r + 1$. In this section we only handle the simple case where the public key includes an encryption of each coefficient of the secret-key $\mathbf{s}$ separately. Namely, denoting $\mathbf{s} = (1, \mathfrak{s})$ and $\mathfrak{s}(X) = \sum_{j=0}^{d-1} \mathfrak{s}_j X^j$, we encode for each j the coefficient $\mathfrak{s}_j$ as the constant polynomial $\mathfrak{s}_j \in (\mathbb{Z}/2^{r+1}\mathbb{Z})[X]/F(X)$. (I.e., the degree-$d$ polynomial whose free term is $\mathfrak{s}_j \in [-2^r + 1, 2^r]$ and all the other coefficients are zero.) Then for each j we include in the public key a ciphertext $\tilde{\mathbf{c}}_j$ that encrypts this constant polynomial $\mathfrak{s}_j$ with respect to $\tilde{\mathbf{s}}$ and q_0. Below we abuse notations somewhat, using the same notation to refer both to a constant polynomial $z \in (\mathbb{Z}/2^r\mathbb{Z})[X]/F(X)$ and the free term of that polynomial $z \in (\mathbb{Z}/2^r\mathbb{Z})$.

3.1 Computing Z′ Homomorphically

Given the q_L-ciphertext $\mathbf{c} = (c_0, c_1)$ (that encrypts a plaintext polynomial $a \in \mathbb{F}_2[X]/F(X)$), we post-process it to get $\mathbf{c}' = (c_0 + c^*, c_1) \bmod 2^{r+1}$ as above, and then use the encryption of $\mathbf{s}$ from the public key to compute the simple decryption formula from above. Computing an encryption of $Z' = [\langle \mathbf{c}', \mathbf{s}\rangle \bmod F(X)]_{2^{r+1}}$ is easy, since the coefficients of Z' are just affine functions (over $(\mathbb{Z}/2^{r+1}\mathbb{Z})$) of the coefficients of $\mathfrak{s}$, which we can compute from the encryption of the $\mathfrak{s}_j$'s in the public key.

3.2 Extracting the Top and Bottom Bits

Now that we have encryptions of the coefficients of Z', we need to extract the top and bottom bits in each of these coefficients and add them (modulo 2) to get encryptions of the plaintext coefficients. In more details, given a ciphertext $\tilde{\mathbf{c}}$ satisfying $[\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}\rangle \bmod F(X)]_{q_0} \equiv z \pmod{2^{r+1}}$ where z is some constant polynomial, we would like to compute another ciphertext $\tilde{\mathbf{c}}'$ satisfying $[\langle \tilde{\mathbf{c}}', \tilde{\mathbf{s}}\rangle \bmod F(X)]_{q_0} \equiv z\langle 0\rangle + z\langle r\rangle \pmod 2$ (with $[\langle \tilde{\mathbf{c}}', \tilde{\mathbf{s}}\rangle \bmod F(X)]_{q_0}$ still much smaller then q_0 in magnitude). To this end, we describe a procedure to compute for all $i = 0, 1, \ldots, r$ a ciphertext $\tilde{\mathbf{c}}_i$ satisfying $[\langle \tilde{\mathbf{c}}_i, \tilde{\mathbf{s}}\rangle \bmod F(X)]_{q_0} \equiv z\langle i\rangle \pmod 2$. Clearly, we can immediately set $\tilde{\mathbf{c}}_0 = \tilde{\mathbf{c}}$, we now describe how to compute the other $\tilde{\mathbf{c}}_i$'s.

The basic observation underlying this procedure is that modulo a power of 2, the second bit of $z - z^2$ is the same as that of z, but the LSB is zero-ed out. Thus setting $z' = (z - z^2)/2$ (which is an integer), we get that the LSB of z' is the second bit of z. More generally, we have the following lemma, whose proof is in Appendix A:

Lemma 2. *Let z be an r-bit integer with binary representation $z = \sum_{i=0}^{r} 2^i z\langle i\rangle$. Define $w_0 \overset{\text{def}}{=} z$, and*

$$\forall i > 1,\ w_i \overset{\text{def}}{=} \frac{z - \sum_{j=0}^{i-1} 2^j w_j^{2^{i-j}} \bmod 2^{r+1}}{2^i} \quad \textit{(division by } 2^i \textit{ over the rationals/integers).}$$

Then the w_i's are integers and we have $w_i\langle 0\rangle = z\langle i\rangle$ for all i.

Our procedure for computing the ciphertexts $\tilde{\mathbf{c}}_i$ mirrors the process of Lemma 2. Specifically, we are given the ciphertext $\tilde{\mathbf{c}} = \tilde{\mathbf{c}}_0$ that encrypts $z = w_0 \bmod 2^{r+1}$, and we iteratively compute ciphertexts $\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \ldots$ such that $\tilde{\mathbf{c}}_i$ encrypts $w_i \bmod 2^{r-i+1}$. Eventually we get $\tilde{\mathbf{c}}_r$ that encrypts $w_r \bmod 2$, which is what we need (since the LSB of w_r is the r'th bit of z).

Note that most of the operations in Lemma 2 are carried out in $(\mathbb{Z}/2^{r+1}\mathbb{Z})$, and therefore can be evaluated homomorphically in our $(\mathbb{Z}/2^{r+1}\mathbb{Z})$-homomorphic cryptosystem. The only exception is the the division by 2^i in Equation (2), and we now show how this division can also be evaluated homomorphically.

To implement division we begin with an arbitrary ciphertext vector $\tilde{\mathbf{c}}$ that encrypts a plaintext element $a \in (\mathbb{Z}/2^j\mathbb{Z})[X]/F(X)$ (for some j) with respect to the level-0 key $\tilde{\mathbf{s}}$ and modulus q_0. Namely, we have the equality over $\mathbb{Z}[X]$:

$$(\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}}\rangle \bmod F(X)) = a + 2^j \cdot S + q_0 \cdot T$$

for some polynomials $S, T \in \mathbb{Z}[X]/F(X)$, where the coefficient norm of $a + 2^j S$ is much smaller than q_0. Assuming that a is divisible by 2 over the integers (i.e., all its coefficients are even) consider what happens when we multiply $\tilde{\mathbf{c}}$ by the integer $(q_0+1)/2$ (which is the inverse of 2 modulo-q_0). Then we have

$$\begin{aligned}
(\left\langle \tfrac{q_0+1}{2}\cdot\tilde{\mathbf{c}},\tilde{\mathbf{s}}\right\rangle \bmod F(X)) &= \tfrac{q_0+1}{2}\cdot(\langle\tilde{\mathbf{c}},\tilde{\mathbf{s}}\rangle \bmod F(X)) \\
&= \frac{(q_0+1)\cdot a}{2}+\frac{(q_0+1)\cdot 2^j\cdot S}{2}+\frac{q_0\cdot(q_0+1)\cdot T}{2} \\
&= (q_0+1)\cdot(a/2) \;+\; (q_0+1)\cdot 2^{j-1}S \;+\; q_0\cdot\tfrac{q_0+1}{2}\cdot T \\
&= a/2 \;+\; 2^{j-1}\cdot S \;+\; q_0\cdot\left(a/2+2^{j-1}S+\tfrac{q_0+1}{2}T\right)
\end{aligned}$$

Clearly the coefficients of $a/2+2^{j-1}S$ are half the size of those of $a+2^jS$, hence they are much smaller than q_0. It follows that $\tilde{\mathbf{c}}'=[\tilde{\mathbf{c}}\cdot(q_0+1)/2]_{q_0}$ is a valid ciphertext that encrypts the plaintext $a/2\in(\mathbb{Z}/2^{j-1}\mathbb{Z})[X]/F(X)$ with respect to secret key $\tilde{\mathbf{s}}$ and modulus q_0.

The same argument shows that if a is divisible by 2^i over the integers (for some $i<j$) then $[\tilde{\mathbf{c}}\cdot((q_0+1)/2)^i]_{q_0}$ is a valid ciphertext encrypting $a/2^i\in(\mathbb{Z}/2^{j-i}\mathbb{Z})[X]/F(X)$. Combining this division-by-two procedure with homomorphic exponentiation mod 2^{r+1}, the resulting homomorphic bit-extraction procedure is described in Figure 1.

Bit-Extraction($\tilde{\mathbf{c}},r,q_0$):

Input: A ciphertext $\tilde{\mathbf{c}}$ encrypting a constant $b\in(\mathbb{Z}/2^{r+1}\mathbb{Z})$ relative to secret key $\tilde{\mathbf{s}}$ and modulus q_0.

Output: A ciphertext $\tilde{\mathbf{c}}'$ encrypting the constant $b\langle 0\rangle\oplus b\langle r\rangle\in\mathbb{F}_2$ relative to secret key $\tilde{\mathbf{s}}$ and modulus q_0.

```
1. Set $\tilde{\mathbf{c}}_0 \leftarrow \tilde{\mathbf{c}}$                          // $\tilde{\mathbf{c}}$ encrypt $z$ w.r.t. $\tilde{\mathbf{s}}$
2. For $i = 1$ to $r$
3.     Set $\mathbf{acc} \leftarrow \tilde{\mathbf{c}}$                       // $\mathbf{acc}$ is an accumulator
4.     For $j = 0$ to $i-1$                       // Compute $z - \sum_j 2^j w_j^{i-1}$
5.         Set $\mathbf{tmp} \leftarrow \mathsf{HomExp}(\tilde{\mathbf{c}}_j, 2^{i-j})$   // Homomorphic exponentiation to the power $2^{i-j}$
6.         Set $\mathbf{acc} \leftarrow \mathbf{acc} - 2^j\cdot\mathbf{tmp} \bmod q_0$
7.     Set $\tilde{\mathbf{c}}_i \leftarrow \mathbf{acc}\cdot((q_0+1)/2)^i \bmod q_0$      // $\tilde{\mathbf{c}}_i$ encrypts $z\langle i\rangle$
8. Output $\tilde{\mathbf{c}}_0 + \tilde{\mathbf{c}}_r \bmod q_0$
```

$\mathsf{HomExp}(\tilde{\mathbf{c}},n)$ *uses native homomorphic multiplication to multiply* $\tilde{\mathbf{c}}$ *by itself* n *times. To aid exposition, this code assumes that the modulus and secret key remain fixed, else modulus-switching and key-switching should be added (and the level should be increased correspondingly to some* $i>0$*).*

Figure 1: A Homomorphic Bit-Extraction Procedure.

3.3 Packing the Coefficients

Now that we have encryption of all the coefficients of a, we just need to "pack" all these coefficients back in one polynomial. Namely, we have encryption of the constant polynomials $a_0,a_1,\ldots$, and we want to get an encryption of the polynomial $\sum_i a_iX^i$. This can be done simply by generating encryptions of the monomials X^i, [4] using the native homomorphism of the cryptosystem to multiply each monomial X^i by the corresponding constant a_i, and then add them all.

[4]For example, the vector $(X^i,0)$ is always an encryption of the monomial X^i when the key has the form $(1,\mathfrak{s})$.

4 Algebraic Background

Below we describe the algebra needed when using the techniques of Gentry et al. from [13] of computing on packed ciphertexts for our needs. Recall that the BGV scheme is defined over a polynomial ring $R = \mathbb{Z}[X]/F(X)$. If the polynomial $F(X)$ factors modulo two into distinct irreducible polynomials $F_0(X) \times \cdots \times F_{\ell-1}(X)$, then, by the Chinese Remainder Theorem, the plaintext space factors into a product of finite fields

$$R_2 \cong \mathbb{F}_2[X]/F_0(X) \times \cdots \times \mathbb{F}_2[X]/F_{\ell-1}(X).$$

This factorization is used in [17, 3, 13] to "pack" a vector of ℓ elements (one from each $\mathbb{F}_2[X]/F_i(X)$) into one plaintext polynomial, which is then encrypted in one ciphertext; each of the ℓ components called a plaintext slot. The homomorphic operations (add/mult) are then applied to the different slots in a SIMD fashion. When $F(X)$ is the m-th cyclotomic polynomial, $F(X) = \Phi_m(X)$, then the field $\mathbb{Q}[X]/F(X)$ is Galois (indeed Abelian) and so the polynomials $F_i(X)$ all have the same degree (which we will denote by d). It was shown in [13] how to evaluate homomorphically the application of the Galois group on the slots, and in particular this enables homomorphically performing arbitrary permutations on the vector of slots in time quasi-linear in m. This, in turn, is used in [13] to evaluate arbitrary arithmetic circuits (of average width $\tilde{\Omega}(\lambda)$) with overhead only polylog(λ).

However, the prior work only mentions the case of plaintext spaces taken modulo a prime (in our case two), i.e. R_2. In this work we will need to also consider plaintext spaces which are given by a power of a prime, i.e. R_{2^t} for some positive integer t. (We stress that by R_{2^t} we really do mean $(\mathbb{Z}/2^t\mathbb{Z})[X]/F(X)$ and not $\mathbb{F}_{2^t}[X]/F(X)$.) We now show how the techniques from [13] extends also to this case. We start by considering general polynomials, and then specialize to cyclotomic polynomials.

4.1 General $F(X)$

It turns out that for monic and square free $F(X)$ modulo 2, all the properties we had modulo two carry over to when working modulo a power of two (i.e. the ℓ slots, the Galois actions, etc). The "high brow" way of seeing this is to consider the message space modulo 2^t as the precision t approximation to the 2-adic integers; namely we need to consider the localization of the field $K = \mathbb{Q}[X]/F(X)$ at the prime 2. We sketch the underlying algebra for those who are interested, and provide elementary proofs of some of the main results, so as to obtain a somewhat self-contained treatment of what we require. However, the reader if referred to [6] for a complete treatment of local fields and their Galois theory.

We present the basic theory for a general prime p, to demonstrate their is nothing special about the prime 2. As usual when dealing with local fields we use the notation $\mathbb{Z}_p$ to denote the p-adic integers, this is the ring of formal power series in p, i.e.

$$\mathbb{Z}_p = \left\{ \sum_{i=0}^{\infty} a_i \cdot p^i : a_i \in \mathbb{F}_p \right\}.$$

In holding an element in $\mathbb{Z}_p$ we only hold it to a given p-adic precision, by which we mean we hold it as an integer in the ring $(\mathbb{Z}/p^t\mathbb{Z})$. Then addition and multiplication in $\mathbb{Z}_p$ is defined by the induced operations obtained from the equivalent operations on $(\mathbb{Z}/p^t\mathbb{Z})$. Intuitively one should think of $\mathbb{Z}_p$ as representing the rings $(\mathbb{Z}/p^t\mathbb{Z})$, for all values of t at once; indeed $\mathbb{Z}_p$ is the projective limit of $(\mathbb{Z}/p^t\mathbb{Z})$ as $t \longrightarrow \infty$. The field of fractions of $\mathbb{Z}_p$ is denoted by $\mathbb{Q}_p$, which is the field of p-adic numbers. A p-adic number can be held as a numerator/denominator pair $(\mathsf{num}, \mathsf{den})$ where $\mathsf{num} \in \mathbb{Z}_p$ and $\mathsf{den} = p^v$ for some $v \in \mathbb{N}$.

We will write $R_{p^\infty} = \mathbb{Z}_p[X]/F(X)$ and note that the approximation to p-adic precision t of this ring is our desired space R_{p^t}. The field of fractions of R_{p^∞} is given by $\mathbb{Q}_p[X]/F(X)$. Our first goal is to understand the algebra of R_{p^∞}, and in so doing, to understand the algebra of R_{p^t}. The following Lemma is key:

Lemma 3. *Let p be a prime, let $t \geq 1$ be an integer, and let $G, H, F \in \mathbb{Z}[X]$ be monic integer polynomials, such that G, H are co-prime modulo p, and $G \cdot H = F \pmod{p^t}$. Then there exist monic polynomials $\tilde{G}, \tilde{H} \in \mathbb{Z}[X]$ such that $G \equiv \tilde{G} \pmod{p^t}$ and $H \equiv \tilde{H} \pmod{p^t}$ and $\tilde{G} \cdot \tilde{H} = F \pmod{p^{t+1}}$.*

All proofs are given in Appendix A. Note, the process in the proof of Lemma 3 of turning a solution modulo p^t to a solution modulo p^{t+1} is called "Hensel Lifting" in the literature.

Corollary 2. *Let p be a prime, let $t \geq 1$ be an integer and let $F \in \mathbb{Z}[X]$ be a monic integer polynomial, which is square-free modulo p. Then F is irreducible modulo p^t if and only if it is irreducible modulo p.*

Theorem 1. *Let p be a prime, let $t \geq 1$ be an integer and let $F \in \mathbb{Z}[X]$ be a monic integer polynomial, which is square-free modulo p. Then the factorization of F modulo p^t is determined uniquely by the factorization of F modulo p.*

The above theorem tells us that the factorization of R_{p^t} into plaintext slots is the same as the factorization of R_p into plaintext slots, for all positive t. Thus if $\mathbf{a} = (a_0, \ldots, a_{\ell-1}) \in R_p$ consisting of a vector of ℓ values, where each value a_i is modulo $F_i(X)$, then we can "lift" it to an element $\overline{\mathbf{a}} = (\overline{a_0}, \ldots, \overline{a_{\ell-1}}) \in \mathbb{R}_{p^t}$ where $a_i \equiv \overline{a_i} \pmod{p}$ and $\overline{a_i}$ is defined modulo $\overline{F_i}(X)$, where $F_i \equiv \overline{F_i} \pmod{p}$ and $\overline{F_i}$ divides F modulo p^t.

4.2 Cyclotomic Polynomials

For our purposes, we will select $F(X)$ to be the m'th cyclotomic polynomial $\Phi_m(X)$. If d is the smallest integer such that m divides $p^d - 1$ then $\Phi_m(X)$ will factor into $\ell = \phi(m)/d$ irreducible polynomials modulo p, each of degree d, i.e. $\Phi_m(X) = \prod_{j=0}^{\ell-1} F_j(X) \pmod{p}$. By the above results we also see that Φ_m factors mod p^t as $\Phi_m(X) = \prod_{j=0}^{\ell-1} \overline{F_j}(X) \pmod{p^t}$ where $F_j \equiv \overline{F_j} \pmod{p}$ and the $\overline{F_j}$'s are monic and irreducible mod p^t.

So if we select $F(X)$ to be $\Phi_m(X)$ then the algebra R_{p^t} splits into ℓ rings $(\mathbb{Z}/p^t\mathbb{Z})[X]/\overline{F_j}(X)$ each of degree d. The following standard result from the theory of p-adic extensions can be found in either Cassels [6, Lemma 2.1] or Cohen [7, Lemma 4.4.26].

Lemma 4. *For every d, upto isomorphism there exists exactly one unramified extension K of $\mathbb{Q}_p$ of degree d, which is the splitting field of $X^q - X$ over $\mathbb{Q}_p$, where $q = p^d$.*

This implies that our ℓ rings $(\mathbb{Z}/p^t\mathbb{Z})[X]/\overline{F_j}(X)$ are isomorphic to each other; via linear maps defined over $\mathbb{Z}/p^{\mathbb{Z}}$ (on any linear basis of $(\mathbb{Z}/p^t\mathbb{Z})[X]/\overline{F_j}(X)$ when considered as a $\mathbb{Z}/p^t\mathbb{Z}$-module). Indeed the above Lemma is the p-adic analogue of the result from finite field theory that all finite fields of a given degree are isomorphic. Just as in [13, 17] (which concerned themselves with the case of finite fields), this allows the following presentational (and computational) simplification. We can pick a single irreducible (over $\mathbb{F}_p$) polynomial of degree d; let us call this polynomial $G(X)$. We then define the ring $R_{p^t}^d$ to be $(\mathbb{Z}/p^t\mathbb{Z})[X]/G(X)$, then each of our ℓ slots in $\mathbb{R}_{p^t}$ are isomorphic to $R_{p^t}^d$; so we can think of operations in $\mathbb{R}_{p^t}$ as being operations on $(R_{p^t}^d)^\ell$. Moreover, again by the above Lemma, we know $R_{p^t}^d$ contains all of the dth roots of unity.

We now turn to examining the Galois Theory of the rings R_{p^t} and $R_{p^t}^d$. The same description as in [13] applies. Namely, that the Galois group of R_{p^t} is isomorphic to the group $(\mathbb{Z}/m\mathbb{Z})^*$. The elements of the Galois group providing the map $X \longrightarrow X^i$ on R_{p^t} for $i \in (\mathbb{Z}/m\mathbb{Z})^*$. The group $\mathcal{G}\mathsf{al}$ contains the decomposition group at p, namely the subgroup $\mathcal{G}$ generated by the element p, which is also the Galois group of $R_{p^t}^d$. The decomposition group has order d, and its elements map roots of $\overline{F_j}$ to roots of $\overline{F_j}$ for the same value of j. The generator of the decomposition group maps the interdeterminant X to X^p in all of our rings, and it commutes with the isomorphisms between the representations $(\mathbb{Z}/p^{\mathbb{Z}})[X]/\overline{F_j}(X)$ and $R_{p^t}^d$. Thus we can apply the Frobenius map to elements of R_{p^t} and think of it applying simultaneously to the representation $\prod(\mathbb{Z}/p^{\mathbb{Z}})[X]/F_j$ and the representation $(R_{p^t}^d)^\ell$.

The quotient group $\mathcal{H} \cong \mathcal{G}\mathsf{al}/\mathcal{G}$ is of order ℓ and given a set of coset representatives for $\mathcal{H}$ we can produce via Galois conjugation, addition and multiplication an arbitrary permutation on the slots of $\mathbb{R}_{p^t}$; just as was done in [13] for the case $t = 1$.

5 Homomorphic Decryption with Packed Ciphertexts

The homomorphic decryption procedure from Section 3 is rather inefficient, mostly because we need to repeat the bit-extraction procedure from Figure 1 for each coefficient separately. Instead, we would like to pack many coefficients in one ciphertext and extract the top bits of all of them together. To this end we employ a batching technique, similar to Smart-Vercauteren (and later works) [3, 13, 17], using Chinese remaindering over the ring of polynomials to pack many "plaintext slots" inside a single plaintext polynomial. These techniques rely on polynomial arithmetic modulo cyclotomic polynomials, so for the rest of this paper we assume $F(X) = \Phi_m(X)$ for an appropriately chosen integer m.

Using these techniques for bootstrapping is not quite straightforward, however. The main difficulty is that the input and output of are not presented in a packed form: The input is a single q_L-ciphertext that encrypts a single plaintext polynomial a (which may or may not have many plaintext elements packed in its slots), and similarly the output needs to be a single ciphertext that encrypts the same polynomial a (but with respect to a larger modulus). Moreover, the bit extraction procedure yields the coefficients of a, again not in a packed form. Our "packed bootstrapping" procedure consists of the following steps:

1. Using the encryption of the q_L-secret-key with respect to the modulus q_0, we convert the initial q_L-ciphertext into a q_0-ciphertext encrypting the polynomial $Z' \in (\mathbb{Z}/2^{r+1}\mathbb{Z})[X]/\Phi_m(X)$.

2. Next we apply a homomorphic inverse-DFT transformation to get encryption of polynomials that have the coefficients of Z' in their plaintext slots.

3. Now that we have the coefficients of Z' in the plaintext slots, we can apply the bit extraction procedure to all these slots in parallel. The result is encryption of polynomials that have the coefficients of a in their plaintext slots.

4. Finally, we apply a homomorphic DFT transformation to get back a ciphertext that encrypts the polynomial a itself.

Below we describe each of these steps in more detail. We note that the main challenge is to get an efficient implementation of Steps 2 and 4.

5.1 Encrypting the q_L-Secret-Key

As in Section 3, we use an encryption scheme with underlying plaintext space modulo 2^{r+1} to encrypt the q_L-secret-key $\mathbf{s}$ under the q_0-secret-key $\tilde{\mathbf{s}}$. The q_L-secret-key is a vector $\mathbf{s} = (1, \mathfrak{s})$, where $\mathfrak{s} \in \mathbb{Z}[X]/\Phi_m(X)$ is an integer polynomial with small coefficients. Viewing these small coefficients as elements in $\mathbb{Z}/2^{r+1}\mathbb{Z}$, we encrypt $\mathfrak{s}$ as a q_0-ciphertext $\tilde{\mathbf{c}} = (\tilde{\mathfrak{c}}_0, \tilde{\mathfrak{c}}_1)$ with respect to the q_0-secret-key $\tilde{\mathbf{s}} = (1, \tilde{\mathfrak{s}})$, namely we have

$$[\langle \tilde{\mathbf{c}}, \tilde{\mathbf{s}} \rangle \bmod \Phi_m]_{q_0} = [\tilde{\mathfrak{c}}_0 + \tilde{\mathfrak{c}}_1 \cdot \tilde{\mathfrak{s}} \bmod \Phi_m]_{q_0} = 2^{r+1}\tilde{k} + \mathbf{s} \quad \text{(equality over } \mathbb{Z}[X])$$

for some polynomial $\tilde{k}$ with small coefficients.

5.2 Step One: Computing Z' Homomorphically

Given a q_L-ciphertext $\mathbf{c} = (c_0, c_1)$ we first post-process it by adding c^* to c_0, getting $c_0' = [c_0 + c^*]_{2^{r+1}}$. Then, given the q_0 ciphertext $\tilde{\mathbf{c}} = (\tilde{\mathfrak{c}}_0, \tilde{\mathfrak{c}}_1)$ that encrypts $\mathfrak{s}$, we compute the mod-2^{r+1} inner product homomorphically by setting

$$\tilde{\mathbf{z}} = \big(\, [c_0' + c_1\tilde{\mathfrak{c}}_0 \bmod \Phi_m]_{q_0},\;\; [c_1\tilde{\mathfrak{c}}_1 \bmod \Phi_m]_{q_0} \,\big). \tag{1}$$

We claim that $\tilde{\mathbf{z}}$ is a q_0-ciphertext encrypting our Z' with respect to the secret key $\tilde{\mathbf{s}}$ (and plaintext space modulo 2^{r+1}). To see that, recall that we have the following two equalities over $\mathbb{Z}[X]$,

$$(c_0' + c_1\mathfrak{s} \bmod \Phi_m) = 2^{r+1}k + Z' \quad \text{and} \quad (\tilde{\mathfrak{c}}_0 + \tilde{\mathfrak{c}}_1\tilde{\mathfrak{s}} \bmod \Phi_m) = q_0\tilde{k} + 2^{r+1}\tilde{k}' + \mathfrak{s},$$

where $k, \tilde{k}, \tilde{k}' \in \mathbb{Z}[X]/\Phi_m$, the coefficients of $2^{r+1}k + Z'$ are smaller than $2q_L^2 \ll q_0$, and the coefficients of $2^{r+1}\tilde{k}' + \mathfrak{s}$ are also much smaller than q_0. It follows that:

$$\begin{aligned}
(\langle \tilde{\mathbf{z}}, \tilde{\mathbf{s}} \rangle \bmod \Phi_m) \quad &= \quad [c_0' + c_1\tilde{\mathfrak{c}}_0 \bmod \Phi_m]_{q_0} + (\tilde{\mathfrak{s}} \cdot [c_1\tilde{\mathfrak{c}}_1 \bmod \Phi_m]_{q_0} \bmod \Phi_m) \\
&= \quad (c_0' + c_1(\tilde{\mathfrak{c}}_0 + \tilde{\mathfrak{c}}_1\tilde{\mathfrak{s}}) \bmod \Phi_m) + q_0\kappa \\
&= \quad (c_0' + c_1(2^{r+1}\tilde{k}' + \mathfrak{s}) \bmod \Phi_m) + q_0\kappa' \\
&= \quad (c_0' + c_1\mathfrak{s} \bmod \Phi_m) + q_0\kappa' + 2^{r+1}(c_1 \cdot \tilde{k}' \bmod \Phi_m) \\
&= \quad q_0\kappa' + 2^{r+1}(k + c_1\tilde{k}' \bmod \Phi_m) + Z' \qquad \text{(equality over } \mathbb{Z}[X])
\end{aligned}$$

for some $\kappa, \kappa' \in \mathbb{Z}[X]/\Phi_m$. Moreover, since the coefficients of c_1 are smaller than $q_L \ll q_0$ then the coefficients of $2^{r+1}(k + c_1\tilde{k}' \bmod \Phi_m) + Z'$ are still much smaller than q_0. Hence $\tilde{\mathbf{z}}$ is decrypted under $\tilde{\mathbf{s}}$ and q_0 to Z', with plaintext space 2^{r+1}.

5.3 Step Two: Switching to CRT Representation

Now that we have an encryption of the polynomial Z', we want to perform the homomorphic bit-extraction procedure from Figure 1. However, this procedure should be applied to each coefficient of Z' separately, which is not directly supported by the native homomorphism of our cryptosystem. (For example, homomorphically squaring the ciphertext yields an encryption of the polynomial $Z'^2 \bmod \Phi_m$ rather than squaring each coefficient of Z' separately.) We therefore need to convert $\tilde{\mathbf{z}}$ to CRT-based "packed" ciphertexts that hold the coefficients of Z' in their plaintext slots.

The system parameter m was chosen so that $m = \tilde{\Theta}(\lambda)$ and $\Phi_m(X)$ factors modulo 2 (and therefore also modulo 2^{r+1}) as a product of degree-d polynomials with $d = O(\log m)$, $\Phi_m(X) = \prod_{j=0}^{\ell-1} F_j(X)$ $(\mathrm{mod}\ 2^{r+1})$. This allows us to view the plaintext polynomial $Z'(X)$ as having ℓ slots, with the j'th slot holding the value $Z'(X) \bmod (F_j(X), 2^{r+1})$. This way, adding/multipliying/squaring the plaintext polynomials has the effect of applying the same operation on each of the slots separately.

In our case, we have $\phi(m)$ coefficients of $Z'(X)$ that we want to put in the plaintext slots, and each ciphertext has only $\ell = \phi(m)/d$ slots, so we need d ciphertexts to holds them all. The transformation from the single ciphertext $\tilde{\mathbf{z}}$ that encrypts Z' itself to the collection of d ciphertexts that hold the coefficients of Z' in their slots is described in Section 5.6 below. (We describe that step last, since it is the most complicated and it builds on machinery that we develop for Step Four in Section 5.5.)

5.4 Step Three: Extracting the Top and Bottom Bits

Once we have the coefficients of Z' in the plaintext slots, we can just repeat the procedure from Figure 1. The input to the the bit-extraction procedure is a collection of some d ciphertexts, each of them holding $\ell = \phi(m)/d$ of the coefficients of Z' in its ℓ plaintext slots. (Recall that we chose $m = \tilde{O}(\lambda)$ such that $d = O(\log m)$.) Applying the procedure from Figure 1 to these ciphertexts will implicitly apply the bit extraction of Lemma 2 to each plaintext slot, thus leaving us with a collection of d ciphertexts, each holding ℓ of the coefficients of a in its plaintext slots.

5.5 Step Four: Switching Back to Coefficient Representation

To finally complete the recryption process, we need to convert the d ciphertexts holding the coefficients of a in their plaintext slots into a single ciphertext that encrypts the polynomial a itself. For this transformation, we appeal to the result of Gentry et al. [13], which says that every depth-L circuit of average-width $\tilde{\Omega}(\lambda)$ and size T can be evaluated homomorphically in time $O(T) \cdot \text{poly}(L, \log \lambda)$, provided that the inputs and outputs are presented in a packed form. Below we show that the transformation we seek can be computed on cleartext by a circuit of size $T = \tilde{O}(m)$ and depth $L = \text{polylog}(m)$, and hence (since $m = \tilde{\Theta}(\lambda)$) it can be evaluated homomorphically in time $\tilde{O}(m) = \tilde{O}(\lambda)$.

To use the result of Gentry et al. we must first reconcile an apparent "type mismatch": that result requires that both input and output be presented in a packed CRT form, whereas we have input in CRT form but output in coefficient form. We therefore must interpret the output as "something in CRT representation" before we can use the result from [13]. The solution is obvious: since we want the output to be a in coefficient representation, then it is a polynomial that holds the value $A_j = a \bmod F_j$ in the j'th slot for all j.

Hence the transformation that we wish to compute takes as input the coefficients of the polynomials $a(X)$, and produces as output the polynomials $A_j = a \bmod F_j$ for $j = 0, 1, \ldots, \ell - 1$. It is important to note that our output consists of ℓ values, each of them a degree-d binary polynomial. Since this output is produced by an arithmetic circuit, then we need a circuit that operates on degree-d binary polynomials, in other words an arithmetic circuit over $\mathbb{GF}(2^d)$. This circuit has $\ell \cdot d$ inputs (all of which happen to be elements of the base field $\mathbb{F}_2$), and ℓ outputs that belong to the extension field $\mathbb{GF}(2^d)$.

Theorem 2. *Fix $m \in \mathbb{Z}$, let $d \in \mathbb{Z}$ be the smallest such that $m | 2^d - 1$, denote $\ell = \phi(m)/d$ and let $G \in \mathbb{F}_2[X]$ be a degree-d irreducible polynomial over $\mathbb{F}_2$ (that fixes a particular representation of $\mathbb{GF}(2^d)$). Let $F_0(X), F_1(X), \ldots, F_{\ell-1}(X)$ be the irreducible (degree-d) factors of the m-th cyclotomic polynomial $\Phi_m(X)$ modulo 2.*

Then there is an arithmetic circuit Π_m over $\mathbb{F}_2[X]/G(X) = \mathbb{GF}(2^d)$ with $\phi(m)$ inputs $a_0, a_1, \ldots, a_{\phi(m)-1}$ and ℓ outputs $z_0, z_1, \ldots, z_{\ell-1}$, for which the following conditions hold:

- *When the inputs are from the base field ($a_i \in \mathbb{F}_2\ \forall i$) and we denote $a(X) = \sum_i a_i X^i \in \mathbb{F}_2[X]$, then the outputs satisfy $z_j = a(X) \bmod (F_j(X), 2) \in \mathbb{F}_2[X]/G(X)$.*

- Π_m *has depth $O(\log m)$ and size $O(m \log m)$.*

The proof is in Appendix A. An immediate corollary of Theorem 2 and the Gentry et al. result [13, Thm. 3], we have:

Corollary 3. *There is an efficient procedure that given d ciphertexts, encrypting d polynomials that hold the coefficients of a in their slots, computes a single ciphertext encrypting a. The procedure works in time $O(m) \cdot$ polylog(m) (and uses at most polylog(m) levels of homomorphic evaluation).*

5.6 Details of Step Two

The transformation of Step Two is roughly the inverse of the transformation that we described above for Step Four, with some added complications. In this step, we have the polynomial $Z'(X)$ over the ring $\mathbb{Z}/2^{r+1}\mathbb{Z}$, and we view it as defining ℓ plaintext slots with the j'th slot containing $B_j \overset{\text{def}}{=} Z' \bmod (F_j, 2^{r+1})$. Note that the B_j's are degree-d polynomials, and we consider them as elements in the "extension ring" $R^d_{2^{r+1}} \overset{\text{def}}{=} \mathbb{Z}[X]/(G(X), 2^{r+1})$ (where G is some irreducible degree-d polynomial modulo 2^{r+1}).

Analogous to Theorem 2, we would like to argue that there is an arithmetic circuit over $R^d_{2^{r+1}}$ that get as input the B_j's (as elements of $R^d_{2^{r+1}}$), and outputs all the coefficients of Z' (which are elements of the base ring $\mathbb{Z}/2^{r+1}\mathbb{Z}$). Then we could apply again to the result of Gentry et al. [13] to conclude that this circuit can be evaluated homomorphically with only polylog overhead.

For the current step, however, the arithmetic circuit would contain not only addition and multiplication gates, but also Frobenius map gates. Namely, gates $\rho_k(\cdot)$ (for $k \in \{1, 2, \ldots, d-1\}$) computing the functions

$$\rho_k\big(u(X)\big) \;=\; u(X^{2^k}) \bmod (G(X), 2^{r+1}).$$

It was shown in [13] that arithmetic circuits with Frobenius map gates can also be evaluated homomorphically with only polylog overhead. The Frobenius operations being simply an additional automorphism operation which can be applied homomorphically to ciphertexts.

Theorem 3. *Fix $m, r \in \mathbb{Z}$, let $d \in \mathbb{Z}$ be the smallest such that $m | 2^d - 1$, denote $\ell = \phi(m)/d$ and let $G(X)$ be a degree-d irreducible polynomial over $\mathbb{Z}/2^{r+1}\mathbb{Z}$ (that fixes a particular representation of $R^d_{2^{r+1}}$). Let $F_0(X), F_1(X), \ldots, F_{\ell-1}(X)$ be the irreducible (degree-d) factors of the m-th cyclotomic polynomial $\Phi_m(X)$ modulo 2^{r+1}.*

Then there is an arithmetic circuit $\Psi_{m,r}$ with Frobenius-map gates over $R^d_{2^{r+1}}$ that has ℓ input $B_0, B_1, \ldots, B_{\ell-1}$ and $\phi(m)$ outputs $b_0, b_1, \ldots, b_{\phi(m)-1}$, for which the following conditions hold:

- *On any inputs $B_0, \ldots, B_{\ell-1} \in R^d_{2^{r+1}}$, the outputs of $\Psi_{m,r}$ are all in the base ring, $Z'_i \in \mathbb{Z}/2^{r+1}\mathbb{Z}\ \forall i$. Moreover, denoting $Z'(X) = \sum_i Z'_i X^i$, it holds that $Z'(X) \bmod (F_j(X), 2^{r+1}) = B_j$ for all j.*

- Π_m *has depth $O(\log m + d)$ and size $O(m(d + \log m))$.*

The proof is in Appendix A. As before, a corollary of Theorem 3 and the result from [13], is the following; where we apply the above theorem for b to our polynomial Z'.

Corollary 4. *There is an efficient procedure that given a single ciphertext encrypting Z' outputs d ciphertexts encrypting d polynomials that hold the coefficients of Z' in their plaintext slots. The procedure works in time $\tilde{O}(m)$ (and uses at most polylog(m) levels of homomorphic evaluation).*

Remark. We note that the transformation from Theorem 3 is nontrivial, mainly because it involves homomorphically reducing a degree-$(m-1)$ polynomial modulo $\Phi_m(X)$. This can be done for any m in depth polylog(m) and size $\tilde{O}(m)$, (see, e.g., [15, Sec. 7.2] or [1, Sec. 17]), but this procedure is rather involved. In some special cases (e.g., when m is prime) reduction modulo $\Phi_m(X)$ is immediate.

5.7 An Alternative Variant

The procedure from Section 5.6 works in time $\tilde{O}(m)$, but it is still quite expensive. One alternative is to put in the public key not just one ciphertext encrypting the q_L-secret-key $\mathfrak{s}$, but rather several ciphertexts (approximately d), encrypting polynomials that hold the coefficients of $\mathfrak{s}$ in their plaintext slots. Then, rather than using the simple formula from Equation (1) above, we evaluate homomorphically the inner product of $\mathbf{s} = (1, \mathfrak{s})$ and $\mathbf{c}' = (c_0 + c^*, c_1)$ modulo $\Phi_m(X)$ and 2^{r+1}. This procedure will be even faster if instead of the coefficients of $\mathfrak{s}$ we encrypt their transformed image under length-m DFT. Then we can compute the DFT of c_1 (in the clear), multiply it homomorphically by the encrypted transformed $\mathfrak{s}$ (in SIMD fashion) and then homomorphically compute the inverse-DFT and the reduction modulo Φ_m. Unfortunately this procedure still requires that we compute the reduction mod-$\Phi_m(X)$ homomorphically, which is likely to be the most complicated part of bootstrapping. Finding a method that does not require this homomorphic polynomial modular reduction is an interesting open problem.

6 Lower-Degree Bit Extraction

As described in Figure 1, extracting the r'th bit requires computing polynomials of degree upto 2^r, here we describe a simple trick to lower this degree. Recall our simplified decryption process: we first post-process the ciphertext to get $\mathbf{c}' \leftarrow [c_0 + c^*, c_1]_{2^{r+1}}$ (where $c^* = \mathbf{1} \cdot 2^{r-1}$), then set $Z' \leftarrow [\langle \mathbf{c}', \mathbf{s} \rangle \bmod \Phi_m(X)]_{2^{r+1}}$, and finally recover $a = [Z'\langle r \rangle + Z'\langle 0 \rangle]_2$.

Consider what happens if we add q_L to all the odd coefficients in $\mathbf{c}'$, call the resulting vector $\mathbf{c}''$: On one hand, now all the coefficients of $\mathbf{c}''$ are even. On the other hand, the coefficients of $Z'' = \langle \mathbf{c}'', \mathbf{s} \rangle \bmod \Phi_m(X)$ are still small enough to use Lemma 1 (since they are at most $c_m \cdot q \cdot \|\mathbf{s}\|_1$ larger than those of Z' itself, where c_m is the ring constant of mod-$\Phi_m(X)$ arithmetic and $\|s\|_1$ is the l_1-norm of $\mathbf{s}$). Since $\mathbf{c}'' = \mathbf{c}' \pmod{q_L}$ then we have

$$[[\langle \mathbf{c}', \mathbf{s} \rangle \bmod \Phi_m(X)]_{q_L}]_2 = [[\langle \mathbf{c}'', \mathbf{s} \rangle \bmod \Phi_m(X)]_{q_L}]_2 = Z''\langle r \rangle + Z''\langle 0 \rangle$$

However, since $\mathbf{c}''$ is even then so is Z''. This means that $Z''\langle 0 \rangle = 0$, and if we divide Z'' by two (over the integers), $Z^* = Z''/2$, then we have $[[\langle \mathbf{c}', \mathbf{s} \rangle \bmod \Phi_m(X)]_{q_L}]_2 = Z^*\langle r-1 \rangle$. We thus have a variation of the simple decryption formula that only needs to extract the $r-1$'st bit, so it can be realized using polynomials of degree upto 2^{r-1}. Note that we can implement this variant of the decryption formula homomorphically, because for an even Z'' we can easily convert an q_0-encryption of Z'' into an encryption of $Z''/2$ by multiplying by $\frac{q_0+1}{2}$ modulo q_0 (as described in Section 3.2).

This technique can be pushed a little further, adding to $\mathbf{c}'$ multiples of q so that it is divisible by 4, 8, 16, etc., and reducing the required degree correspondingly to 2^{r-2}, 2^{r-3}, 2^{r-4}, etc. The limiting factor is that we must maintain that $\langle \mathbf{c}'', \mathbf{s} \rangle$ has coefficients sufficiently smaller than q_L^2, in order to be able to use Lemma 1. Clearly, if $\mathbf{c}'' = \mathbf{c}' + q\kappa$ where all the coefficients of κ are smaller than some bound B (in absolute value), then the coefficients of $\langle \mathbf{c}'', \mathbf{s} \rangle$ can be larger than the coefficients of $Z' = \langle \mathbf{c}', \mathbf{s} \rangle$ (in absolute value) by at most $c_m \cdot q \cdot B \cdot \|\mathbf{s}\|_1$. (Heuristically we expect the difference to depend on the l_2 norm of $\mathbf{s}$ more than its l_1 norm.)

If we choose our parameters such that the l_1-norm of $\mathbf{s}$ is below m, and work over a ring with $c_m = O(1)$, then the coefficients of Z' can be made as small as $c_m \cdot m \cdot q$, and we can make the coefficients of κ as large as $B \approx q/(4c_m \cdot m)$ in absolute value while maintaining the invariant that the coefficients of Z'' are smaller than $q^2/4$ (which is what we need to be able to use Lemma 1). By choosing an appropriate κ, we can ensure that the least significant $\lfloor \log(q/(4c_m m)) \rfloor = r - \lceil \log(4c_m m) \rceil$ bits of $\mathbf{c}''$ are all zero. This means that we can implement bit extraction using only polynomials of degree at most $2^{\lceil \log(4c_m m) \rceil} < 8c_m m = O(m)$. (Heuristically, we should even be able to get polynomials of degree $O(\sqrt{m})$ since the l_2 norm of $\mathbf{s}$ is only $O(\sqrt{m})$.)

To get an even smaller degree we can use squashing: Recall that we have $\mathbf{s} = (1, \mathfrak{s})$ for some secret polynomial $\mathfrak{s}$. We can add to the public key a small number $t = O(\log m)$ of polynomials $u_1, u_2, \ldots, u_t$, such that for some "very sparse polynomials" $v_1, \ldots, v_t$ it holds that $\sum_i u_i v_i = \mathfrak{s} \pmod{\Phi_m(X), q_L}$. The v_i's are even much sparser than $\mathfrak{s}$ itself, specifically the l_1 norm of all of them together is only m^ϵ for some constant $\epsilon < 1$. Given the ciphertext $\mathbf{c} = (c_0, c_1)$ we can post-process it by setting $f_i = c_1 \cdot u_i \bmod (\Phi_m(X), q_L)$, then decryption of the post-processed ciphertext $\mathbf{f} = (c_0, f_1, \ldots, f_t)$ using the new secret key $\mathbf{v} = (1, v_1, \ldots, v_m)$ is done in principle using the same formula $a = [[\langle \mathbf{f}, \mathbf{v} \rangle \bmod \Phi_m(X)]_{q_L}]_2$. We can now simplify the decryption formula and evaluate it homomorphically exactly as we did in Sections 2, 3, and 5. The advantage of doing this is that the l_1 norm of the new key $\mathbf{v}$ is much smaller, only m^ϵ as opposed to m, so using the same trick as above the bit extraction procedure can be computed by polynomials of degree only $O(m^\epsilon)$. (This squashing technique is essentially equivalent to using key-switching in order to switch the q_L ciphertext $\mathbf{c}$ into a ciphertext $\mathbf{f}$ with respect to the very sparse key $\mathbf{v} = (1, v_1, \ldots, v_m)$.)

References

[1] Daniel J. Bernstein. Fast multiplication and its applications. *Algorithmic Number Theory, MSRI Publications*, 44:325–384, 2008.

[2] Leo I. Bluestein. A linear filtering approach to the computation of the discrete fourier transform. Northeast Electronics Research and Engineering Meeting Record 10, 1968.

[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science (ITCS'12)*, 2012. Available at `http://eprint.iacr.org/2011/277`.

[4] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS'11*. IEEE Computer Society, 2011.

[5] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[6] John William Scott Cassels. *Local Fields*, volume 3 of *LMS Student Texts*. Cambridge University Press, 1986.

[7] Henri Cohen. *Number Theory: Volume II: Analytic and Modern Tools: Analytic and Modern Tools*, volume 240 of *GTM*. Springer-Verlag, 2007.

[8] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010. Full version available on-line from `http://eprint.iacr.org/2009/616`.

[9] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[10] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.

[11] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *FOCS'11*. IEEE Computer Society, 2011.

[12] Craig Gentry and Shai Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.

[13] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully Homomorphic Encryption with Polylog Overhead. Manuscript at `http://eprint.iacr.org/2011/566`, 2011.

[14] Ron Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180, 1978.

[15] Victor Shoup. A new polynomial factorization algorithm and its implementation. *Journal of Symbolic Computation*, 20:363–397, 1995.

[16] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography - PKC'10*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[17] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. Manuscript at `http://eprint.iacr.org/2011/133`, 2011.

A Proofs

Lemma 1. *Let $q = 2^r + 1$ for a positive integer r, and let z be a non-negative integer smaller than $\frac{q^2}{2} - q$, such that $[z]_q$ is also non-negative, $[z]_q \in [0, \frac{q}{2}]$. Then $[[z]_q]_2 = z\langle r\rangle \oplus z\langle 0\rangle$.*

Proof. Let $z_0 = [z]_q \in [0, \frac{q}{2}]$, and consider the sequence of integers $z_i = z_0 + iq$ for $i = 0, 1, 2, \cdots$. Since we assume that $z \geq 0$ then z can be found in this sequence, say the k'th element $z = z_k = z_0 + kq$. Also since $z < \frac{q^2}{2} - q$ then $k = \lfloor z/q \rfloor < \frac{q}{2} - 1$. The bit that we want to compute is $[[z]_q]_2 = z_0\langle 0\rangle$. We claim that $z_0\langle 0\rangle = z_k\langle 0\rangle + z_k\langle r\rangle \pmod 2$. This is because

$$z_k \;=\; z_0 + kq \;=\; z_0 + k(2^r + 1) \;=\; (z_0 + k) + k2^r,$$

which in particular means that $z_k\langle 0\rangle = z_0\langle 0\rangle + k\langle 0\rangle \pmod 2$. But since $0 \leq z_0 \leq q/2$ and $0 \leq k < q/2 - 1$ then $0 \leq z_0 + k < q - 1 = 2^r$, so there is no carry bit from the addition $z_0 + k$ to the r'th bit position. It follows that the r'th bit of z_k is equal to the 0'th bit of k (i.e., $z_k\langle r\rangle = k\langle 0\rangle$), and therefore

$$z_k\langle 0\rangle \;=\; z_0\langle 0\rangle + k\langle 0\rangle \;=\; z_0\langle 0\rangle + z_k\langle r\rangle \pmod 2,$$

which implies that $z_0\langle 0\rangle = z_k\langle 0\rangle + z_k\langle r\rangle \pmod 2$, as needed. □

Corollary 1. *Let $r \geq 3$ and $q = 2^r + 1$ and let z be an integer with absolute value smaller than $\frac{q^2}{4} - q$, such that $[z]_q \in (-\frac{q}{4}, \frac{q}{4})$. Denote $z' = z + (q^2-1)/4$, then $[[z]_q]_2 = z'\langle r\rangle \oplus z'\langle 0\rangle$.*

Proof. Note that

$$z' \;=\; z + (q^2-1)/4 \;=\; z + (q+1)(q-1)/4 \;=\; \big(z + \frac{q-1}{4}\big) \;+\; q \cdot \frac{q-1}{4},$$

and since $(q-1)/4 = 2^{r-2}$ is an even integer, then $z' \equiv z + \frac{q-1}{4} \pmod q$. Moreover since $[z]_q \in (-\frac{q}{4}, \frac{q}{4}]$ then $[z]_q + \frac{q-1}{4} \in [0, q/2]$, hence $[z']_q = [z]_q + \frac{q-1}{4}$ (over the integers), and as $\frac{q-1}{4}$ is an even integer then $[z]_q = [z']_q \pmod 2$, or in other words $[[z]_q]_2 = [[z']_q]_2$.

Since $z > -\frac{q^2}{4}$ and z is an integer then $z \geq -\frac{q^2-1}{4}$ and therefore $z' = z + \frac{q^2-1}{4} \geq 0$. Thus z' satisfies all the conditions set in Lemma 1, so applying that lemma we have $[[z]_q]_2 = [[z']_q]_2 = z'\langle r\rangle \oplus z'\langle 0\rangle$. □

Lemma 2. *Let z be an r-bit integer with binary representation $z = \sum_{i=0}^{r} 2^i z\langle i\rangle$. Define $w_0 \overset{\text{def}}{=} z$, and*

$$\forall i > 1,\; w_i \;\overset{\text{def}}{=}\; \frac{z - \sum_{j=0}^{i-1} 2^j w_j^{2^{i-j}} \bmod 2^{r+1}}{2^i} \quad \textit{(division by } 2^i \textit{ over the rationals/integers).} \qquad (2)$$

Then the w_i's are integers and we have $w_i\langle 0\rangle = z\langle i\rangle$ for all i.

Proof. The lemma clearly holds for $i = 0$. Now fix some $i \geq 1$, assume that the lemma holds for all $j < i$, and we prove that it holds also for i. It is easy to show by induction that for any integer u and all $j \leq r$ we have

$$u^{2^j} \bmod 2^{r+1} = u\langle 0\rangle + 2^{j+1}t \text{ for some integer } t.$$

Namely, the LSB of $u^{2^j} \bmod 2^{r+1}$ is the same as the LSB of u, and the next j bits are all zero. This means that the bit representation of $v_j \overset{\text{def}}{=} 2^j w_j^{2^{i-j}} \bmod 2^{r+1}$ has bits $0, 1, \ldots, j-1$ all zero (due to the multiplication by 2^j), then $v_j\langle j\rangle = w_j\langle 0\rangle = z\langle j\rangle$ (by the induction hypothesis), and the next $i-j$ bits are again zero (by the observation above). In other words, the lowest $i+1$ bits of v_j are all zero, except the j'th bit which is equal to the j'th bit of z.

This means that the lowest i bits of the sum $\sum_{j=0}^{i-1} v_j$ are the same as the lowest i bits of z, and the $i+1$'st bit of the sum is zero. Hence the lowest i bits of $z - \sum_{j=0}^{i-1} v_j$ are all zero, and the $i+1$'st bit is $z\langle i\rangle$. Hence $z - \sum_{j=0}^{i-1} v_j$ is divisible by 2^i (over the integers), and the lowest bit of the result is $z\langle i\rangle$, as needed. □

Lemma 3. *Let p be a prime, let $t \geq 1$ be an integer, and let $G, H, F \in \mathbb{Z}[X]$ be monic integer polynomials, such that G, H are co-prime modulo p, and $G \cdot H = F \pmod{p^t}$. Then there exist monic polynomials $\tilde{G}, \tilde{H} \in \mathbb{Z}[X]$ such that $G \equiv \tilde{G} \pmod{p^t}$ and $H \equiv \tilde{H} \pmod{p^t}$ and $\tilde{G} \cdot \tilde{H} = F \pmod{p^{t+1}}$.*

Proof. Considering the product $G \cdot H$ modulo p^{t+1}, we know that it is of the form $G \cdot H = F + p^t \cdot F'$ for some $F' \in \mathbb{F}_p[X]$ (since it equals F modulo p^t). Moreover, since G, H are monic then the top coefficient of $G \cdot H = F + p^t \cdot F'$ is one, which means that F' must have lower degree than F. We next prove that there exist $G', H' \in \mathbb{F}_p[X]$ such that $(G + p^t \cdot G') \cdot (H + p^t \cdot H') = F \pmod{p^{t+1}}$. Opening parentheses, removing terms that vanish modulo p^{t+1}, and substituting $F + p^t \cdot F'$ for $G \cdot H$, we get the equivalent formula:

$$\begin{aligned} F &= (G + p^t \cdot G') \cdot (H + p^t \cdot H') \\ &= G \cdot H + p^t \cdot (G \cdot H' + G' \cdot H) + p^{2t} \\ &= (F + p^t \cdot F') - p^t \cdot (G \cdot H' + G' \cdot H) \\ &= F + p^t \cdot (F' - G \cdot H' - G' \cdot H) \pmod{p^{t+1}}, \end{aligned}$$

which after subtracting F and dividing by p^t is equivalent to the condition $G \cdot H' + G' \cdot H = F' \pmod{p}$.

Since G, H are co-prime over the field $\mathbb{F}_p$, then there exists an inverse of H modulo $(G(X), p)$ and vice versa. Defining $G' = F' \cdot H^{-1} \bmod (G(X), p)$ and $H' = G^{-1} \bmod H(X), p$, we see

$$G \cdot H' + G' \cdot H \;=\; 0 \cdot H' + (F' \cdot H^{-1}) \cdot H \;=\; F' \pmod{G(X), p}$$

and similarly $G \cdot H' + G' \cdot H = F' \pmod{H(X), p}$. By Chinese remainders (and using $G \cdot H = F \pmod{p}$) we have $GH' + G'H = F' \pmod{F(X), p}$. But the degree of $G \cdot H' + G' \cdot H$ is smaller than that of $F = G \cdot H$, and so is the degree of F', hence we obtain $G \cdot H' + G' \cdot H = F' \pmod{p}$, without the reduction modulo F.

It follows that for $\tilde{G} = G + p^t \cdot G' \bmod p^{t+1}$ and $\tilde{H} = G + p^t \cdot H'$, we have $\tilde{G} \cdot \tilde{H} = F \pmod{p^{t+1}}$, and since G, H are monic and have degrees larger than G', H', respectively, then also $\tilde{G}$ and $\tilde{H}$ are monic. □

Corollary 2. One direction is obvious: if F is *not* irreducible modulo p^t then there exist two nontrivial monic polynomials $G, H \in \mathbb{Z}[X]$ such that $G \cdot H = F \pmod{p^t}$, so in particular $G \cdot H = F \pmod{p}$.

For the other direction, fix F which is square free but not irreducible modulo p, and we show by induction on t that it has two nontrivial monic factors modulo p^t, which are co-prime over $\mathbb{F}_p$. The base case $t = 1$ is obvious, and the induction step is exactly Lemma 3. □

Theorem 1. *Let p be a prime, let $t \geq 1$ be an integer and let $F \in \mathbb{Z}[X]$ be a monic integer polynomial, which is square-free modulo p. Then the factorization of F modulo p^t is determined uniquely by the factorization of F modulo p.*

Proof. Suppose F modulo p^t factors into irreducible factors $\overline{F_0}, \ldots, \overline{F_{\ell-1}}$. Then by setting $F_i = \overline{F_i} \pmod{p}$ we obtain the factorization of F modulo p. Since $\overline{F_i}$ is irreducible modulo p^t, by Corollary 2 we have that F_i is irreducible modulo p. In addition we cannot have $F_i = F_j$ modulo p since F is square free.

Going in the other direction: Given a factorization modulo p we can obtain a factorization modulo p^t by repeated application of Lemma 3 □

Theorem 2. *Fix $m \in \mathbb{Z}$, let $d \in \mathbb{Z}$ be the smallest such that $m | 2^d - 1$, denote $\ell = \phi(m)/d$ and let $G \in \mathbb{F}_2[X]$ be a degree-d irreducible polynomial over $\mathbb{F}_2$ (that fixes a particular representation of $\mathbb{GF}(2^d)$). Let $F_0(X), F_1(X), \ldots, F_{\ell-1}(X)$ be the irreducible (degree-d) factors of the m-th cyclotomic polynomial $\Phi_m(X)$ modulo 2.*

Then there is an arithmetic circuit Π_m over $\mathbb{F}_2[X]/G(X) = \mathbb{GF}(2^d)$ with $\phi(m)$ inputs $a_0, a_1, \ldots, a_{\phi(m)-1}$ and ℓ outputs $z_0, z_1, \ldots, z_{\ell-1}$, for which the following conditions hold:

- *When the inputs are from the base field ($a_i \in \mathbb{F}_2\ \forall i$) and we denote $a(X) = \sum_i a_i X^i \in \mathbb{F}_2[X]$, then the outputs satisfy $z_j = a(X) \bmod (F_j(X), 2) \in \mathbb{F}_2[X]/G(X)$.*
- *Π_m has depth $O(\log m)$ and size $O(m \log m)$.*

Proof. Recall that $\mathbb{GF}(2^d)$ contains a primitive m-th root of unity τ, and that Φ_m factors into linear terms over $\mathbb{GF}(2^d)$, $\Phi_m(X) = \prod_{i\in(\mathbb{Z}/m\mathbb{Z})^*}(X - \tau^i)$. Moreover, each of the irreducible factors of Φ_m modulo 2 is a product of the terms $(X - \tau^i)$ from one conjugacy class: namely there is a size-ℓ set of representative indexes $L = \{e_0, e_1, \ldots, e_{\ell-1}\} \subset (\mathbb{Z}/m\mathbb{Z})^*$ such that for each $j = 0, 1, \ldots, \ell-1$ it holds that

$$F_j(X) = (X - \tau^{e_j})(X - \tau^{2e_j})(X - \tau^{4e_j}) \cdots (X - \tau^{2^{d-1}e_j}).$$

The circuit for computing all the polynomials $A_j \overset{\text{def}}{=} a \bmod F_j$, consists of first applying the DFT to the coefficients of a, thus computing the evaluations $a(\tau^j)$ for all $j \in (\mathbb{Z}/m\mathbb{Z})^*$, and then computing each A_j from the d values $a(\tau^{e_j}), a(\tau^{2e_j}), \ldots, a(\tau^{2^{d-1}e_j})$.

The first part, computing the size-m DFT over $\mathbb{GF}(2^d)$, can be accomplished by a circuit of depth $O(\log m)$ and size-$\tilde{O}(m \log m)$ using FFT techniques (e.g., using Bluestein's trick [2]). The result is the m-vector of values $a(\tau^i)$ for $i = 0, 1, \ldots, m-1$, from which we extract only the values $a(\tau^i)$ for $i \in (\mathbb{Z}/m\mathbb{Z})^*$.

Next, consider an arbitrary index $j \in \{0, 1, \ldots, \ell-1\}$ for which we want to compute $A_j = a \bmod F_j \in \mathbb{GF}(2^d)$. We show that A_j can be obtained as a fixed linear combination over $\mathbb{GF}(2^d)$ of the values $a(\tau^{e_j}), a(\tau^{2e_j}), \ldots, a(\tau^{2^{d-1}e_j})$. To see that, first note that for all $i \in \{0, 1, \ldots, d-1\}$ we have $F_j(\tau^{e_j 2^i}) = 0$. Since $A_j = a \bmod F_j = a - \kappa F_j$ (for some κ), then we get

$$A_j(\tau^{e_j 2^i}) \;=\; a(\tau^{e_j 2^i}) - \kappa(\tau^{e_j 2^i}) \underbrace{F_j(\tau^{e_j 2^i})}_{=0} \;=\; a(\tau^{e_j 2^i}).$$

The coefficients of the degree-$(d-1)$ polynomial A_j can therefore be computed from the d values $A_j(\tau^{e_j 2^i}) = a(\tau^{e_j 2^i})$, for $i = 0, 1, \ldots, d-1$. Specifically, let $V_j \in \mathbb{GF}(2^d)^{d\times d}$ be the Vandermonde matrix on the points $\tau^{2^i e_j}$:

$$V_j \overset{\text{def}}{=} \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \tau^{e_j} & \tau^{2e_j} & \tau^{3e_j} & \ldots & \tau^{(d-1)e_j} \\ 1 & \tau^{2e_j} & \tau^{4e_j} & \tau^{6e_j} & \ldots & \tau^{2(d-1)e_j} \\ 1 & \tau^{4e_j} & \tau^{8e_j} & \tau^{12e_j} & \ldots & \tau^{4(d-1)e_j} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \tau^{2^{d-1}e_j} & \tau^{2^{d-1}\cdot 2e_j} & \tau^{2^{d-1}\cdot 3e_j} & \ldots & \tau^{2^{d-1}(d-1)e_j} \end{bmatrix}.$$

If we denote by $\mathbf{u}$ the column vector $\mathbf{u} \overset{\text{def}}{=} \left(a(\tau^{e_j}), a(\tau^{2e_j}), a(\tau^{4e_j}), \ldots, a(\tau^{2^{d-1}e_j})\right)^t \in \mathbb{GF}(2^d)^d$, then the coefficient vector of A_j is obtained as $V_j^{-1}\mathbf{u}$.

Moreover, we can get the field element $A_j \in \mathbb{GF}(2^d)$ from its vector of coefficients by multiplying the free term by $1 \in \mathbb{GF}(2^d)$, the next coefficient by $X \in \mathbb{GF}(2^d)$, the one after that by $X^2 \in \mathbb{GF}(2^d)$, etc., and then summing all these products. Denoting $\mathbf{x} \overset{\text{def}}{=} \left(1, X, X^2, \ldots, X^{d-1}\right) \in \mathbb{GF}(2^d)^d$ and $\mathbf{w} \overset{\text{def}}{=} \mathbf{x}V_j^{-1}$, we have

$$\langle \mathbf{w}, \mathbf{u} \rangle \;=\; \mathbf{x}V_j^{-1}\mathbf{u} \;=\; \left(1, X, X^2, \ldots, X^{d-1}\right) \cdot \left(V_j^{-1}\mathbf{u}\right) \;=\; A_j.$$

We conclude that once we have all the values $a(\tau^i)$, they can be partitioned into the ℓ different conjugacy classes, and then each A_j can be computed as a fixed linear combination of the values in the corresponding conjugacy class. Thus following the FFT circuit we need to compute ℓ linear combinations of d terms each, which takes an additional circuit of size $\ell \cdot d = O(m)$ (and insignificant depth). $\square$

Theorem 3. *Fix $m, r \in \mathbb{Z}$, let $d \in \mathbb{Z}$ be the smallest such that $m|2^d - 1$, denote $\ell = \phi(m)/d$ and let $G(X)$ be a degree-d irreducible polynomial over $\mathbb{Z}/2^{r+1}\mathbb{Z}$ (that fixes a particular representation of $R^d_{2^{r+1}}$). Let $F_0(X), F_1(X), \ldots, F_{\ell-1}(X)$ be the irreducible (degree-d) factors of the m-th cyclotomic polynomial $\Phi_m(X)$ modulo 2^{r+1}.*

Then there is an arithmetic circuit $\Psi_{m,r}$ with Frobenius-map gates over $R^d_{2^{r+1}}$ that has ℓ input $B_0, B_1, \ldots$, $B_{\ell-1}$ and $\phi(m)$ outputs $b_0, b_1, \ldots, b_{\phi(m)-1}$, for which the following conditions hold:

- *On any inputs $B_0, \ldots, B_{\ell-1} \in R^d_{2^{r+1}}$, the outputs of $\Psi_{m,r}$ are all in the base ring, $Z'_i \in \mathbb{Z}/2^{r+1}\mathbb{Z}\ \forall i$. Moreover, denoting $Z'(X) = \sum_i Z'_i X^i$, it holds that $Z'(X) \bmod (F_j(X), 2^{r+1}) = B_j$ for all j.*
- *Π_m has depth $O(\log m + d)$ and size $O(m(d + \log m))$.*

Proof. Recall that the structure of $R^d_{2^{r+1}}$ is determined by that of $\mathbb{GF}(2^d)$. In particular $R^d_{2^{r+1}}$ contains a primitive m-th root of unity τ, and Φ_m factors into linear terms over $R^d_{2^{r+1}}$, $\Phi_m(X) = \prod_{i \in (\mathbb{Z}/m\mathbb{Z})^*}(X - \tau^i)$. Moreover, each of the irreducible factors of Φ_m modulo 2^{r+1} is a product of the terms $(X - \tau^i)$ from one conjugacy class: namely for the same size-ℓ set of representative indexes as in the proof of Theorem 2, $L = \{e_0, e_1, \ldots, e_{\ell-1}\} \subset (\mathbb{Z}/m\mathbb{Z})^*$, it holds that $F_j(X) = (X - \tau^{e_j})(X - \tau^{2e_j}) \cdots (X - \tau^{2^{d-1}e_j})$ for all j. Conceptually, the circuit $\Psi_{m,r}$ consists of first computing the coefficients of the input polynomials B_j, then computing the values $Z'(\tau^i)$, and finally computing the coefficients of b (but the first two parts can be combined into one step).

For the first part, we need a circuit that takes as input the ℓ extension-ring elements $B_0, \ldots, B_{\ell-1} \in R^d_{2^{r+1}}$ and outputs the $\phi(m)$ elements of the base ring which are the coefficients of these extension-ring elements (as represented by degree-d polynomials in $\mathbb{Z}[X]/(G(X), 2^{r+1})$).

We claim that for each extension-ring element $u \in R^d_{2^{r+1}}$ the coefficients of $u(X)$ can be computed as fixed linear combinations of the d Frobenius maps $\mu_k = \rho_k(u)$: Indeed, for the extension-ring element $u(X) = \sum_i u_i X^i$ we have the equalities over $R^d_{2^{r+1}}$ $\rho_k(u(X)) = u(X^{2^k}) = \sum_{i=0}^{d-1} u_i X^{i2^k}$ for all k. Writing these equalities in matrix form, we have

$$\begin{bmatrix} u(X) \\ \rho_1(u(X)) \\ \rho_2(u(X)) \\ \vdots \\ \rho_{d-1}(u(X)) \end{bmatrix} = \begin{bmatrix} 1 & X & X^2 & & X^{d-1} \\ 1 & X^2 & X^4 & \ldots & X^{2(d-1)} \\ 1 & X^4 & X^8 & & X^{4(d-1)} \\ & \vdots & & \ddots & \\ 1 & X^{2^{d-1}} & X^{2 \cdot 2^{d-1}} & & X^{(d-1)2^{d-1}} \end{bmatrix} \cdot \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{d-1} \end{bmatrix}$$

Denoting the above matrix by M (namely $M_{i,k} = X^{i2^k} \bmod (G(X), 2^{r+1}) \in R^d_{2^{r+1}}$), we can obtain the coefficient vector of u as $M^{-1} \cdot \big(u, \rho_1(u), \rho_2(u), \ldots, \rho_{d-1}(u)\big)^t$.

Once we have the coefficients of the B_j's, we can evaluate them at the m-th roots of unity τ^i by multiplying by the corresponding Vandermonde matrices. As in the proof of Theorem 2, we have $B_j(\tau^i) = Z'(\tau^i)$ for the indexes i in the j'th conjugacy class (namely $i = e_j \cdot 2^k$ for some k). Denoting again the Vandermonde matrix for the j'th conjugacy class by V_j, we can therefore compute the values $Z'(\tau^i)$ for the j'th conjugacy class as $\big(Z'(\tau^{e_j}), Z'(\tau^{2e_j}), \ldots, Z'(\tau^{2^{d-1}e_j})\big)^t = V_j M^{-1}\big(B_j, \rho_1(B_j), \rho_2(B_j), \ldots, \rho_{d-1}(B_j)\big)^t$.

The first two parts of the circuit $\Psi_{m,r}$ therefore consist of applying to each input B_j the d Frobenius maps, and then multiplying each of the resulting d-vectors by the corresponding matrices $V_j M^{-1}$. Hence for these parts we have size $O(\ell \cdot d^2) = O(m \cdot d)$ and depth at most $O(\log d)$.

Once we have all the values $Z'(\tau^i)$ for $i \in (\mathbb{Z}/m\mathbb{Z})^*$, we can compute the coefficients of Z' using an inverse-DFT transform, and moreover this can be done in depth polylog(m) and size $\tilde{O}(m)$ using FFT techniques. We note that this transformation is nontrivial, since computing the size-m inverse DFT transformation requires also the values $Z'(\tau^i)$ for i which is not co-prime with m. One option is to pad with zeros and compute the size-m inverse DFT to get the coefficients of degree-$(m-1)$ polynomial Z'' such that $Z'' = Z' \pmod{\Phi_m(X), 2^{r+1}}$, and then reduce modulo $\Phi_m(X)$ homomorphically to get Z' itself. The reduction modulo Φ_m can be done generically in depth polylog(m) and size $\tilde{O}(m)$, (see, e.g., [15, Sec. 7.2] or [1, Sec. 17]), even though this procedure is rather involved. $\square$