Bootstrapping BGV Ciphertexts With A Wider Choice of p and q

E. Orsini, J. van de Pol and N.P. Smart

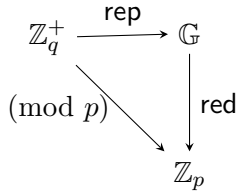
Dept. Computer Science,
University of Bristol,
United Kingdom.
{Emmanuela.Orsini,Joop.VandePol}@bristol.ac.uk, {nigel}@cs.bris.ac.uk

Abstract. We describe a method to bootstrap a packed BGV ciphertext which does not depend (as much) on any special properties of the plaintext and ciphertext moduli. Prior "efficient" methods such as that of Gentry et al (PKC 2012) required a ciphertext modulus q which was close to a power of the plaintext modulus p. This enables our method to be applied in a larger number of situations. Also unlike previous methods our depth grows only as $\log \log q$ as opposed to the $\log q$ of previous methods. The basic bootstrapping technique makes use of a representation of the group $\mathbb{Z}_q^+$ over the finite field $\mathbb{F}_p$ (either based on polynomials or elliptic curves). This technique is then extended to the full BGV packed ciphertext space, using a method whose depth depends only logarithmically on the number of packed elements. This method may be of interest as an alternative to the method of Alperin-Sheriff and Peikert (CRYPTO 2013). To aid efficiency we utilize the ring/field switching technique of Gentry et al (SCN 2012, JCS 2013).

1 Introduction

Since the invention of Fully Homomorphic Encryption (FHE) by Gentry in 2009 [11], the main open question in the field has been how to "bootstrap" a Somewhat Homomorphic Encryption (SHE) scheme into a FHE scheme. Recall an SHE scheme is one which can evaluate circuits of a limited multiplicative depth, whereas an FHE scheme is one which can evaluate circuits of arbitrary depth. The most efficient SHE schemes are ones which allow SIMD style operations, by packing a number of plaintext elements into independent "slots" in the plaintext space. The most studied of such "SIMD friendly" schemes being the BGV scheme [4] based on the Ring-LWE Problem.

Prior Work on Bootstrapping: One can divide the bootstrapping of all efficient currently known SHE schemes into three distinct sub-problems. The first problem is to homomorphically evaluate the reduction mod p map on the group $\mathbb{Z}_q^+$, where the domain takes representatives centred around zero. To do this the group $\mathbb{Z}_q^+$ is first mapped to a set $\mathbb{G}$ in which one can perform operations native to the homomorphic cryptosystem. In other words we first need to specify a *representation* $\mathsf{rep} : \mathbb{Z}_q^+ \longrightarrow \mathbb{G}$,which takes an integer in the range $(-q/2, \ldots, q/2]$ and maps it to the set $\mathbb{G}$. The group operation on $\mathbb{Z}_q^+$ needs to induce a group operation on $\mathbb{G}$ which can be evaluated homomorphically by the underlying SHE scheme. Then we describe the induced map $\mathsf{red} : \mathbb{G} \longrightarrow \mathbb{Z}_p$ as a algebraic operation, which can hence be evaluated homomorphically.



The second problem is to encode the secret key in a way that one can publicly, using a function dec-eval, create a set of ciphertexts which encrypt the required input to the function red. And thirdly one needs a method to extend this to packed ciphertexts.

To solidify ideas we now expand on these problems in the context of the BGV scheme [4]. Recall for BGV we have a set of $L+1$ moduli, corresponding to the levels of the scheme, $q_0 < q_1 < \ldots < q_L$, and a (global) ring R, which is often the ring of integers of a cyclotomic number field. We let p denote the (prime) plaintext modulus, i.e. the plaintexts will be elements in R_p (the localisation of R at the prime p), and to ease notation we set $q = q_0$. The secret key $\mathfrak{sk}$ is a small element in R. A "fresh" ciphertext encrypting $\mu' \in R_p$ is an element $\mathfrak{ct}' = (c'_0, c'_1)$ in $R^2_{q_L}$ such that

$$\left(c'_0 + \mathfrak{sk} \cdot c'_1 \pmod{q_L}\right) \pmod{p} = \mu'.$$

After the evaluation of L levels of multiplication one obtains a ciphertext $\mathfrak{ct} = (c_0, c_1)$ in R^2_q, encrypting a plaintext μ, such that

$$(c_0 + \mathfrak{sk} \cdot c_1 \pmod{q}) \pmod{p} = \mu.$$

At this point to perform further calculations one needs to bootstrap, or recrypt, the ciphertext to one of a higher level.

Assume for the moment that each plaintext only encodes a single element of $\mathbb{Z}_p$, i.e. each plaintext is a constant polynomial in polynomial basis for R_p. To perform bootstrapping we need to place a "hint" in the public key $\mathfrak{pk}$ (usually an encryption of $\mathfrak{sk}$ at level L), which allows the following operations. Firstly, we can evaluate homomorphically a function dec-eval which takes $\mathfrak{ct}$ and the "hint", and outputs a representation of the $\mathbb{Z}_q$ element corresponding to the constant term of the element $c_0 + \mathfrak{sk} \cdot c_1 \pmod{q}$. This representation is an encryption of an element in $\mathbb{G}$, i.e. dec-eval also evaluates the rep map as well as the decryption map. Then we apply homomorphically the function red to this representation to obtain a fresh encryption of the plaintext. Since to homomorphically evaluate red we need the input to red to be defined over the plaintext space, this means the representation of $\mathbb{Z}_q$ must be *defined over* $\mathbb{F}_p$. One is then left with the task of extending such a procedure to packed ciphertexts.

In the original bootstrapping technique of Gentry [11], implemented in [12], the function dec-eval is obtained from a process of bit-decomposition. Thus the representation $\mathbb{G}$ of $\mathbb{Z}_q$ is the bit-representation of an integer in the range $(-q/2, \ldots, q/2]$. The function to evaluate red is then the circuit which performs reduction modulo p. The extension of this technique to packed ciphertexts, in the context of the Smart–Vercauteren SIMD optimisations [23] of Gentry's SHE scheme, was given in [24]. Due to the use of bit-decomposition techniques this method is mainly suited to the case of $p = 2$, although one can extend it to other primes by applying a p-adic decomposition and then using an arithmetic circuit to evaluate the reduction modulo p map.

In [14] the authors present a bootstrapping technique, primarily targeted at the BGV scheme, which does away with the need for evaluating the "standard" circuit for the reduction modulo p map. This is done by choosing q close to a power of p, i.e. one selects $q = p^t \pm a$ for some t and a small value of a, typically $a \in \{-1, 1\}$. The paper [14] expands on this idea for the case of $p = 2$, but the authors mention it can be clearly extending to arbitrary p. The advantage is that the mapping red can now be expressed as an algebraic formula; in fact a formula of multiplicative depth $\log_2 q$. The operation dec-eval obtains the required representation for $\mathbb{Z}_q$ by embedding it into $\mathbb{Z}_{p^{t+1}}$. This requires the extension of the modulus of the plaintext ring (for which all the required properties of R_p carry over, assuming that p does not ramify). The extension to packed ciphertexts is performed using an elaborate homomorphic evaluation of the Fourier Transform.

To enable the faster evaluation of this Fourier Transform step, a method for ring/field switching is presented in [13]. This enables the ring R to be changed to a sub-ring S (both for the ciphertext and plaintext spaces). In [1] this use of field switching is combined with the red map from [14] to obtain an asymptotically efficient bootstrapping method for BGV style SHE schemes. In [22] this method is implemented, with

surprisingly efficient runtimes, for the case of plaintext space $\mathbb{F}_2$; i.e. $p = 2$ and no plaintext SIMD-packing is supported.

In another line of work, the authors of [2] and [6] present a bootstrapping technique for the GSW [16] homomorphic encryption scheme. The GSW scheme is one based on matrices, and this property is exploited in [2] by taking a matrix representation of $\mathbb{Z}_q$ and then expressing the map red via a very simple algebraic relationship on the associated matrices. In particular the authors represent elements of $\mathbb{Z}_q$ by matrices (of some large dimension) over $\mathbb{F}_p$.

Thus we see *all* bootstrapping techniques require us to come up with a representation $\mathbb{G}$ of $\mathbb{Z}_q$ for which there is an algebraic method over $\mathbb{F}_p$ to evaluate the induced mapping red, from the said representation of $\mathbb{Z}_q$, to $\mathbb{Z}_p$. Since SHE schemes usually homomorphically have add and multiply operations as their basic homomorphic operations, this implies we are looking for representations of $\mathbb{Z}_q^+$ as a subgroup of an algebraic group over $\mathbb{F}_p$.

Our Contribution: We return to the case of bootstrapping the Ring-LWE based BGV scheme. However, instead of concentrating on the case of plaintext moduli p for which a power of which is close to q, we instead look at a much larger class plaintext moduli. Recall the most efficient prior technique, based on [1] and [14], requires a method whose multiplicative depth is $O(\log q)$, and for which q is close to a power of p. As p increases the ability to select a suitable modulus q which is both close to a power of p, is of the correct size for most efficient implementation (i.e. the smallest needed to ensure security), and has other properties related to efficiency (i.e. the ring R_q has a double-CRT representation as in [15]) diminishes.

To allow a wider selection for p we utilize two "new" (for bootstrapping) representations of the ring $\mathbb{Z}_q$, in much the same way as [2] used an $\mathbb{F}_p$-matrix representation (a.k.a. a linear algebraic group) of $\mathbb{Z}_q^+$. The first one, used for much of paper for ease of presentation, is based on a polynomial representation for $\mathbb{Z}_q^+$ over $\mathbb{F}_p$, the second one (which is less efficient but allows a greater freedom in selecting q) is based on a representation via elliptic curves. The evaluation of the mapping red using these representations can then be done in expected multiplicative depth $O(\log \log q)$, i.e. a much shallower circuit than used in prior works.

To ensure this method works, and is efficient, we do not have completely free reign in selecting q for the first polynomial representation. Whilst [14] required $q = p^t \pm a$, for a small value of a, we instead will require that q divides

$$\mathsf{lcm}\left(p^{k_1} - 1, \ldots, p^{k_t} - 1\right)$$

for some pairwise co-prime values k_i. Thus the freedom on selecting q is much greater than for the method in [14], especially for large values of p. In the second representation, see Section 8, we simply need to find elliptic curves over $\mathbb{F}_{p^{k_i}}$ whose group order is divisible by e_i where $\prod e_i = q$. For the elliptic curve based version we do not need pairwise co-prime values of k_i. Indeed on setting $t = 1$ we simply need one curve $E(\mathbb{F}_{p^{k_1}})$ whose group order is divisible by q, which is highly likely to exist by the near uniform distribution of elliptic curve group orders in the Hasse interval.

In the polynomial representation, one does not have complete freedom on selecting the k_i values. If we let $E = \sum k_i$ and $M = \sum k_i \cdot (k_i - 1)$ then the depth of the circuit (which is approximately $\log_2 \log_2 q - \log_2 \log_2 E$) to evaluate red will decrease as E grows, but the number of multiplications required, which is a monotonically increasing function of M, will increase. Note, we can asymptotically make $M = O(\sum k_i \cdot \log k_i)$ using FFT techniques, or $M = O(\sum k_i^{1.58})$ using Karatsuba based techniques, but in practice the k_i will be too small to make such optimizations fruitful. For the ellptic curve based version we replace the above E by $E + 1$ and we replace M by a constant multiple of M. However, the depth required by our elliptic curve based version increases.

We then extend this bootstrapping method to packed ciphertexts, using a form of p-adic decomposition and a matrix representation of the ciphertext ring. When combined with ring switching we are able to bootstrap a set of ciphertexts in one step. We end by working out the range of parameters the method would support for various plaintext moduli p.

2 Preliminaries

Throughout this work vectors are written using bold lower-case letters, whereas bold upper-case letters are used for matrices. We denote by $M_{a\times b}(K)$ the set of $a \times b$ dimensional matrices with entries in K. For an integer modulus q, we let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the quotient ring of integers modulo q, and $\mathbb{Z}_q^+$ its additive group. This notation naturally extends to the localisation R_q of a ring R at q.

2.1 Algebraic Background

Let m be a positive integer we define the mth cyclotomic field to be the field $\mathbb{K} = \mathbb{Q}[X]/\Phi_m(X)$, where $\Phi_m(X)$ is the mth cyclotomic polynomial. $\Phi_m(X)$ is a monic irreducible polynomial over the rational, and $\mathbb{K}$ is a field extension of degree $N = \phi(m)$ over $\mathbb{Q}$ since $\Phi_m(X)$ has degree N. Let ζ_m be a fixed primitive mth roots of unity, we have that $\mathbb{K} \cong \mathbb{Q}(\zeta_m)$ by identifying ζ_m with X. In the same way, let us denote by R the mth cyclotomic ring $\mathbb{Z}[\zeta_m] \cong \mathbb{Z}[X]/\Phi_m(X)$, with "power basis" $\{1, \zeta_m, \ldots, \zeta_m^{N-1}\}$. The complex embeddings of $\mathbb{K}$ are $\sigma_i : \mathbb{K} \to \mathbb{C}$, defined by $\sigma_i(X) = \zeta_m^i$, $i \in \mathbb{Z}_m^*$. In particular $\mathbb{K}$ is Galois over $\mathbb{Q}$ and $\mathrm{Gal}(\mathbb{Q}(\zeta_m)/\mathbb{Q}) \cong \mathbb{Z}_m^*$. As a consequence we can define the $\mathbb{Q}$-linear (field) trace $\mathrm{Tr}_{\mathbb{K}/\mathbb{Q}} : \mathbb{K} \to \mathbb{Q}$ as the sum of the embeddings σ_i, i.e. $\mathrm{Tr}_{\mathbb{K}/\mathbb{Q}}(a) = \sum_{i\in\mathbb{Z}_m^*} \sigma_i(a) \in \mathbb{Q}$. Concretely, these embeddings map ζ_m into each of its conjugates, and they are the only field homomorphisms from $\mathbb{K}$ to $\mathbb{C}$ that fix every element of $\mathbb{Q}$. The *canonical embedding* $\sigma : \mathbb{K} \to \mathbb{C}^N$ is the concatenation of all the complex embeddings, i.e. $\sigma(a) = (\sigma_i(a))_{i\in\mathbb{Z}_m^*}$, $a \in \mathbb{K}$.

Looking ahead, we will use the ring R and its localisation R_q, for some modulus q. Given a polynomial $a \in R$, we denote by $\|\mathbf{a}\|_\infty = \max_{0\le j\le N-1} |a_j|$ the standard l_∞-norm. All estimates of noise are taken with respect to the *canonical embedding norm* $\|a\|_\infty^{\mathsf{can}} = \|\sigma(a)\|_\infty$, $a \in R$. When considering R_q, we need the canonical embedding norm induced modulo q:

$$|a|_q^{\mathsf{can}} = \min\{\|a'\|_\infty^{\mathsf{can}} : a' \in R \text{ and } a' \equiv a \mod q\}.$$

To map from norms in the canonical embedding to norms on the coefficients of the polynomial defining the elements of R, we have $\|\mathbf{a}\|_\infty \le c_m \cdot \|a\|_\infty^{\mathsf{can}}$, where c_m is the *ring constant*. For more details about c_m see [10].

Let m' be a positive integer such that $m'|m$. As before we define $\mathbb{K}' \cong \mathbb{Q}(\zeta_{m'})$ and $S \cong \mathbb{Z}[\zeta_{m'}]$, such that $\mathbb{K}'$ has degree $n = \phi(m')$ over $\mathbb{Q}$ and $\mathrm{Gal}(\mathbb{K}'/\mathbb{Q}) \cong \mathbb{Z}_{m'}^*$. It is trivial to show that $\mathbb{K}$ and R are a field and a ring extension of $\mathbb{K}'$ and R', respectively, both of dimension N/n. In particular we can see S as a subring of R via the ring embedding that maps $\zeta_{m'} \mapsto \zeta_m^{m/m'}$.

It is a standard fact that if $\mathbb{Q} \subseteq \mathbb{K}' \subseteq \mathbb{K}$ is a tower of number field, then $\mathrm{Tr}_{\mathbb{K}/\mathbb{Q}}(a) = \mathrm{Tr}_{\mathbb{K}'/\mathbb{Q}}(\mathrm{Tr}_{\mathbb{K}/\mathbb{K}'}(a))$, and that all the $\mathbb{K}'$-linear map $L : \mathbb{K} \to \mathbb{K}'$ are exactly the maps of the form $\mathrm{Tr}_{\mathbb{K}/\mathbb{K}'}(r \cdot a)$, for some $r \in \mathbb{K}$.

2.2 Plaintext Slots

Let p be a prime integer, coprime to m, and R_p the localisation of R at p. The polynomial $\Phi_m(X)$ factors modulo p into $\ell(R)$ irreducible factors, i.e. $\Phi_m(X) \equiv \prod_{i=1}^{\ell(R)} F_i(X) \pmod p$. Each $F_i(X)$ has degree $d(R) =$

$\phi(m)/\ell^{(R)}$, where $d^{(R)}$ is the multiplicative order of p in $\mathbb{Z}_m^*$. Looking ahead, each of these $\ell^{(R)}$ factors corresponds to a "plaintext slot", i.e.

$$R_p \cong \mathbb{Z}_p[X]/F_1(X) \times \cdots \times \mathbb{Z}_p[X]/F_{\ell^{(R)}}(X) \cong (\mathbb{F}_{p^{d^{(R)}}})^{\ell^{(R)}}.$$

More precisely, we have $\ell^{(R)}$ isomorphisms $\psi_i : \mathbb{Z}_p[X]/F_i(X) \to \mathbb{F}_{p^{d^{(R)}}}$, $i = 1, \ldots, \ell^{(R)}$, that allow to represent $\ell^{(R)}$ plaintext elements of $\mathbb{F}_{p^{(d)}}$ as a single element in R_p. By the Chinese Remainder Theorem, addition and multiplication correspond to SIMD operations on the slots and this allows to process $\ell^{(R)}$ input values at once.

2.3 Ring Switching

As mentioned in the introduction, our technique uses a method for ring/field switching from [13]. We use two different cyclotomic rings R and S such that $S \subseteq R$. This procedure permits to transform a ciphertext $\mathfrak{ct} \in (R_q)^2$ corresponding to a plaintext $\mu \in R_p$ with respect to a secret key $\mathfrak{sk} \in R$, into a ciphertext $\mathfrak{ct}' \in (S_q)^2$ corresponding to a plaintext $\mu' \in S_p$ with respect to a secret key $\mathfrak{sk}' \in S$. The security of this method relies on the hardness of the ring-LWE problem in S ([20]). At a high level the ring switching consists of three steps. Given an input ciphertext $\mathfrak{ct} \in (R_q)^2$:

- First, it switches the secret key; it uses the "classical" key-switching ([5],[4]), getting a ciphertext $\bar{\mathfrak{ct}} \in (R_q)^2$, still encrypting $\mu \in R_p$, but with respect to a secret key $\mathfrak{sk}' \in S$.
- Second, it multiplies $\bar{\mathfrak{ct}}$ by a fixed element $r \in R$, which is determined by a S-linear function $L : R_p \to S_p$ corresponding to the induced projection function $P : (\mathbb{F}_{p^{d^{(R)}}})^{\ell^{(R)}} \to (\mathbb{F}_{p^{d^{(S)}}})^{\ell^{(S)}}$ (see [13] for details).
- Finally, it applies to $\bar{\mathfrak{ct}}$ the trace function $\mathrm{Tr}_{R/S} : R \to S$. In such a way the output of the ring-switching is a ciphertext $\mathfrak{ct} \in S$ with respect to the secret key $\mathfrak{sk}'$ and encrypting the plaintext $\mu' = L(\mu)$.

We conclude this section noting that, while big-ring ciphertexts correspond to $\ell^{(R)}$ plaintext slots, small-ring ciphertexts only correspond to $\ell^{(S)} \leq \ell^{(R)}$ plaintext slots. The input ciphertexts to our bootstrapping procedure are defined over $(S_q)^2$, and so are of degree n and contain $\ell^{(S)}$ slots. We take $\ell^{(R)}/n$ of these ciphertexts and use the dec-eval map to encode the coefficients of the plaintext polynomials in the slots of a single big-ring ciphertext. Eventually, via ring switching and polynomial interpolation, we return to $\ell^{(R)}/n$ ciphertexts which have been bootstrapped and are at level one (or more). These fresh ciphertexts may be defined over the big ring or the small ring (depending when ring switching occurs). However, our parameter estimates imply that ring switching is best performed at the lowest level possible, and so our bootstrapped ciphertexts will be in the big ring. We could encode all of the slots of the bootstrapped ciphertexts in a big-ring single ciphertext, or not, depending on the application, since slot manipulation is a linear operation.

3 Evaluating the Map red $\circ$ rep : $\mathbb{Z}_q^+ \longrightarrow \mathbb{F}_p$ (Simple Version)

As explained in the introduction at the heart of every bootstrapping procedure is a method to evaluate the induced mapping $\mathsf{red} \circ \mathsf{rep} : \mathbb{Z}_q^+ \longrightarrow \mathbb{F}_p$. In this section we present our simpler technique for doing this based on polynomials over $\mathbb{F}_p$, in Section 8 we present a more general (and complicated in terms of depth) technique based on elliptic curves. The key, in this and in all techniques, is to find a representation $\mathbb{G}$ for $\mathbb{Z}_q^+$ for which the reduction modulo p map can be evaluated algebraically over $\mathbb{F}_p$. This means that the

representation of $\mathbb{Z}_q$ must defined over $\mathbb{F}_p$. Prior work has looked at the bit-representation (when $p = 2$), the p-adic representation and a matrix representation; we use a polynomial representation.

We select a coprime factorization $q = \prod_{i=1}^{t} e_i$ (with the e_i not necessarily prime, but pairwise coprime) such that e_i divides $p^{k_i} - 1$ for some k_i, *and* such that $\mathbb{F}_{p^{k_i}}$ has a subgroup of order e_i. We fix a polynomial representation of $\mathbb{F}_{p^{k_i}}$, i.e. an irreducible polynomial $f_i(x)$ of degree k_i such that $\mathbb{F}_{p^{k_i}} = \mathbb{F}_p[x]/f_i(x)$. Let $g_i \in \mathbb{F}_{p^{k_i}}$ denote a fixed element of order e_i in $\mathbb{F}_{p^{k_i}}$, which exists by assumption.

By the Chinese Remainder Theorem we therefore have a group isomorphism

$$\mathsf{rep} : \begin{cases} \mathbb{Z}_q^+ \longrightarrow \mathbb{G} = \prod_{i=1}^{t} \mathbb{F}^*_{p^{k_i}} \\ a \longmapsto (g_1^{a_1}, \ldots, g_t^{a_t}) \end{cases} \tag{1}$$

where $a_i = a \pmod{e_i}$. Without loss of generality we can assume that the k_i are also coprime. Given this group representation of $\mathbb{Z}_q^+$ in $\mathbb{G}$, addition in $\mathbb{Z}_q^+$ translates into multiplication in $\mathbb{G}$. With one addition in $\mathbb{Z}_q^+$ translating into $M = \frac{1}{2}\sum_{i=1}^{t} k_i \cdot (k_i - 1)$ multiplications in $\mathbb{F}_p$ (and a comparable number of additions; assuming school book multiplication is used). Each element in the image of rep requires $E = \sum_{i=1}^{t} k_i$ elements in $\mathbb{F}_p$ to represent it.

There will be a map $\mathsf{red} : \mathbb{G} \to \mathbb{F}_p$ such that $\mathsf{red} \circ \mathsf{rep}$, is the reduction modulo p map; and red can be defined by *algebraically* from the coefficient representation of $\mathbb{G}$ to $\mathbb{F}_p$. Here algebraically refers to algebraic operations over $\mathbb{F}_p$.

An arbitrary algebraic expression on E variables of degree d will contain $\sum_{i=1}^{d} {}^{E}C_i \approx E^d$ terms. Thus, by interpolating we expect the degree d of the map red to be the smallest d such that $\sum_{i=1}^{d} {}^{E}C_i > q$, which means we expect $d \approx \log q / \log E$. Thus the larger E is, the smaller d will be. The algebraic circuit which implements the map red can hence be described as a circuit of depth $\lceil \log_2 d \rceil$ which requires $D(E, d) = {}^{E+d}C_d - (E + 1)$ multiplications (corresponding to the number of distinct monomials in E variables of degree between two and d).

4 The BGV Somewhat Homomorphic Encryption Scheme

In this section we outline what we need about the BGV SHE scheme [4]. As anticipated in Section 2, we present the scheme with the option of utilizing two rings, and hence at some point we will make use of the ring/field switching procedure from [13]. We first define two rings $R = \mathbb{Z}[X]/F(X)$ and $S = \mathbb{Z}[X]/f(X)$, where $F(X)$ (resp. $f(x)$) is an irreducible polynomial over Z of degree N (resp. n). We assume that S is a subring of R and that n divides N, and that the associated number fields are Galois. In practice both $F(X)$ and $f(X)$ will likely be a cyclotomic polynomials. There is an embedding $\iota : S \longrightarrow R$ which maps elements in S to their appropriate equivalent in R. The map ι can be expressed as a linear mapping on the coefficients of the polynomial representation of the elements in S, to the coefficients of the polynomial representation of the elements in R.

Let R_q (resp. S_q) denote the localisation of R (resp S) at q, i.e. $\mathbb{Z}_q[X]/F(X)$ (resp. $\mathbb{Z}_q[X]/f(X)$), which can be constructed for any positive integer q. Let p be a prime number, which does not ramify in either R or S. Since the rings are Galois, the ring R_p (resp. S_p) splits into $\ell^{(R)}$ (resp. $\ell^{(S)}$) "slots"; with each slot being a finite field extension of $\mathbb{F}_p$ of degree $d^{(R)} = N/\ell^{(R)}$ (resp. $d^{(S)} = n/\ell^{(S)}$). We make the assumption that n divides $\ell^{(R)}$. This is not strictly necessary but it ensures that we can perform bootstrapping of a single ciphertext with the smallest amount of memory. In fact our method will support the bootstrapping of $\ell^{(R)}/n$ ciphertexts in parallel.

There will be two secret keys for our scheme; depending on whether the ciphertexts/plaintexts are associated with the ring R or the ring S. We denote these secret keys by $\mathfrak{sk}^{(R)}$ and $\mathfrak{sk}^{(S)}$, which are "small"

elements in the ring R (resp. S). The modulus $q = q_0 = p_0$ will denote the smallest modulus in the set of BGV levels. Fresh ciphertexts are defined for the modulus $Q = q_L = \prod_{i=0}^{L} p_i$ and live in the ring R_Q^2 (thus at some point we not only perform modulus switching but also ring switching). We assume L_1 levels are associated with the big ring R and L_2 levels are associated with the small ring S, hence $L_1 + L_2 = L$ (level zero is clearly associated with the small ring S, but we do not count it in the number of levels in L_2). Thus we encrypt at level L; perform standard homomorphic operations down to level zero, with a single field switch at level $L_2 + 1$. For ease of analysis we assume no multiplications are performed at level $L_2 + 1$. This means that we can evaluate a depth $L - 1$ circuit.

A ciphertext at level $i > L_2$, encrypting a message $\mu \in R_p$, is a pair $\mathfrak{ct} = (c_0, c_1) \in R_{q_i}^2$, where $q_i = \prod_{j=0}^{i} p_j$, such that

$$\Big(c_0 + \mathfrak{sk}^{(R)} \cdot c_1 \pmod{q_i}\Big) \pmod{p} = \mu.$$

We let $\mathsf{Enc}_{\mathfrak{pk}}(\mu)$ denote the encryption of a message $\mu \in R_p$, this produces a ciphertext at level L. A similar definition holds for ciphertexts at level $i < L_2$, for messages in S_p and secret keys/ciphertexts elements in S_{q_i}. When performing a ring switching operation between levels $L_2 + 1$ and L_2, the $\ell^{(R)}$ plaintext slots, associated with the input ciphertext at level $L_2 + 1$, become associated with $\ell^{(R)}/\ell^{(S)}$ distinct ciphertexts at level L_2.

We want to "bootstrap" a set of BGV ciphertexts. Such a ciphertext is a pair $\mathfrak{ct}_j = (c_0^{(j)}, c_1^{(j)}) \in S_q^2$, for $j = 1, \ldots, \ell^{(R)}/n$, such that

$$\Big(c_0^{(j)} + \mathfrak{sk}^{(S)} \cdot c_1^{(j)} \pmod{q}\Big) \pmod{p} = \mu_j, \text{ for } j = 1, \ldots, \ell^{(R)}/n.$$

5 A Product of Powers of SIMD Vectors

Before proceeding with our method to turn the above methodology for reduction modulo p into a bootstrapping method for our set of BGV ciphertexts, we first examine how to homomorphically compute the following function

$$\mathbf{v} \cdot \prod_{k=0}^{\lambda} \mathbf{v}_k^{\mathbf{M}_k},$$

where each $\mathbf{v}$ and $\mathbf{v}_k$, $k = 0, \ldots, \lambda$, represents a set of E ciphertexts, each of which encode (in a SIMD manner) $\ell^{(R)}$ elements in $\mathbb{F}_p$. The multiplication of two such sets of E ciphertexts is done with respect to the multiplication operation in $\mathbb{G}$, and thus requires M homomorphic multiplications (this is for our simple variation of red, for the variant based on elliptic curve the number of ciphertexts and the complexity of the group operation in $\mathbb{G}$ increase a little). The values $\mathbf{M}_k$ are matrices in $M_{\ell^{(R)} \times \ell^{(R)}}(\mathbb{F}_p)$. By the notation $\mathbf{u} = \mathbf{v}^{\mathbf{M}}$, where $\mathbf{M} = (m_{i,j})$, we mean the vector with components

$$u_i = \prod_{j=1}^{\ell^{(R)}} v_j^{m_{i,j}}, \quad i \in \{1, \ldots, \ell^{(R)}\}.$$

Notice that each u_i and v_j is a vector of E elements in $\mathbb{F}_p$ representing a single element in $\mathbb{G}$. In what follows we divide this operation into three sub-procedures and compute the number of multiplications, and the depth required, to evaluate the function.

5.1 SIMD Raising of an Encrypted Vector to the Power of a Public Vector

The first step is to take a vector $\mathbf{v}$ which is the SIMD encryption of E sets of $\ell^{(R)}$ elements in $\mathbb{F}_p$, and raise it to the power of some public vector $\mathbf{c} = (c_1, \ldots, c_{\ell^{(R)}})$, i.e. we want to compute

$$\mathbf{x} = \mathbf{v}^{\mathbf{c}}.$$

In particular $\mathbf{v}$ actually consists of E vectors each with $\ell^{(R)}$ components in their slots. We write

$$\mathbf{v} = (\mathbf{v}_{1,0}, \ldots, \mathbf{v}_{1,k_1-1}, \ldots, \mathbf{v}_{t,0}, \ldots, \mathbf{v}_{t,k_t-1}).$$

Note, multiplying such a vector by another vector of the same form requires M homomorphic multiplications and depth 1. We first write

$$\mathbf{c} = \mathbf{c}_0 + 2 \cdot \mathbf{c}_1 + \ldots + 2^{\lceil \log_2 p \rceil} \cdot \mathbf{c}_{\lceil \log_2 p \rceil},$$

where $\mathbf{c}_i \in \{0,1\}^{\ell^{(R)}}$. We let $\mathbf{c}_i^*$ denote the bitwise complement of $\mathbf{c}_i$. Thus to compute $\mathbf{x} = \mathbf{v}^{\mathbf{c}}$ we use the following three steps:

Step 1: Compute $\mathbf{v}^{2^i}$ for $i = 1, \ldots, \lceil \log_2 p \rceil$, by which we mean every element in $\mathbf{v}$ is raised to the power 2^i. This requires $\lceil \log_2 p \rceil \cdot M$ homomorphic multiplications and depth $\lceil \log_2 p \rceil$.

Step 2: For $i \in \{0, \ldots, \lceil \log_2 p \rceil\}$, $j \in \{1, \ldots, t\}$ and $k = \{0, \ldots, k_t - 1\}$ compute,

$$\mathbf{w}_{j,k}^{(i)} = \begin{cases} \mathsf{Enc}_{\mathfrak{pk}}(\mathbf{c}_i) \cdot \mathbf{v}_{j,k}^{2^i} & k \neq 0, \\ \mathsf{Enc}_{\mathfrak{pk}}(\mathbf{c}_i) \cdot \mathbf{v}_{j,k}^{2^i} + \mathsf{Enc}_{\mathfrak{pk}}(\mathbf{c}_i^*) & k = 0. \end{cases}$$

Where $\mathsf{Enc}_{\mathfrak{pk}}(\mathbf{c}_i)$ means encrypt the vector $\mathbf{c}_i$ so that the jth component of $\mathbf{c}_i$ is mapped to the jth plaintext slot of the ciphertext. The above procedure selects the values which we want to include in the final product. This involves a homomorphic multiplication by a constant in $\{0,1\}$ and the homomorphic addition of a constant in $\{0,1\}$ for each entry, and so is essentially fast (and moderately bad on the noise, so we will ignore this and call it depth $1/2$).

Step 3: We now compute $\mathbf{x}$ as

$$\mathbf{x} = \prod_{i=0}^{\lceil \log_2 p \rceil} \mathbf{w}^{(i)}$$

where we think of $\mathbf{w}^{(i)}$ as a vector of E SIMD encryptions. This step (assuming a balanced multiplication tree) requires depth $\lceil \log_2 \lceil \log_2 p \rceil \rceil$ and $M \cdot \lceil \log_2 p \rceil$ multiplications.

Executing all three steps above therefore requires a depth of $\frac{1}{2} + \lceil \log_2 p \rceil + \lceil \log_2 \lceil \log_2 p \rceil \rceil$, and $2 \cdot M \cdot \lceil \log_2 p \rceil$ multiplications.

5.2 Computing $\mathbf{u} = \mathbf{v}^{\boldsymbol{M}}$

Given the previous subsection, we can now evaluate $u_i = \prod_{j=1}^{\ell^{(R)}} v_j^{m_{i,j}}, i = 1, \ldots, \ell^{(R)}$, where $\mathbf{v}$ is a SIMD vector consisting of E vectors encoding $\ell^{(R)}$ elements, as is the output $\mathbf{u}$. For this we use a trick for systolic matrix-vector multiplication in [17], but converted into multiplicative notation.

We write the matrix $\mathbf{M}$ as $\ell^{(R)}$ SIMD vectors $\mathbf{d}_i$, for $i = 1, \ldots, \ell^{(R)}$, so that $\mathbf{d}_{i,j} = m_{j,(j+i-1) \pmod{\ell^{(R)}}}$ for $j = 1, \ldots, \ell^{(R)}$. We let $\mathbf{v} \lll i$ denote the SIMD vector $\mathbf{v}$ rotated left i positions (with wrap around). Since $\mathbf{v}$ actually consists of E SIMD vectors this can be performed using time proportional to E multiplications, but with no addition to the overall depth (it is an expensive in terms of time, but cheap in terms of noise. See the operations in Table 1 of [17]).

Step 1: First compute, for $i = 1, \ldots, \ell^{(R)}$,

$$\mathbf{x}_i = (\mathbf{v} \lll (i-1))^{\mathbf{d}_i}$$

using the method previously described in Subsection 5.1. This requires a depth of $\frac{1}{2} + \lceil \log_2 p \rceil + \lceil \log_2 \lceil \log_2 p \rceil \rceil$, and $\ell^{(R)} \cdot (E + 2 \cdot M \cdot \lceil \log_2 p \rceil)$ multiplications.

Step 2: All we need now do is compute

$$\mathbf{u} = \prod_{i=1}^{\ell^{(R)}} \mathbf{x}_i.$$

This requires (assuming a balanced multiplication tree) a depth of $\lceil \log_2 \ell^{(R)} \rceil$ and $\ell^{(R)}$ multiplications.

Thus far, for the operations in Subsection 5.1 and this subsection we have used a total depth of $\frac{1}{2} + \lceil \log_2 \ell^{(R)} \rceil + \lceil \log_2 p \rceil + \lceil \log_2 \lceil \log_2 p \rceil \rceil$ and $\ell^{(R)} \cdot (1 + E + 2 \cdot M \cdot \lceil \log_2 p \rceil)$ multiplications.

5.3 Computing $\mathbf{v} \cdot \prod_{\boldsymbol{k}=0}^{\boldsymbol{\lambda}} \mathbf{v}_{\boldsymbol{k}}^{\boldsymbol{M_k}}$

To evaluate our required output we need to execute the above steps λ times, to obtain the elements which we then multiply together. Thus in total we have a depth of

$$\frac{1}{2} + \lceil \log_2 \ell^{(R)} \rceil + \lceil \log_2 p \rceil + \lceil \log_2 \lceil \log_2 p \rceil \rceil + \lceil \log_2 \lambda \rceil$$

and

$$\lambda \cdot \left(1 + \ell^{(R)} \cdot (1 + E + 2 \cdot M \cdot \lceil \log_2 p \rceil)\right)$$

multiplications.

6 Bootstrapping a Set of Ciphertexts

To perform our bootstrapping operation we introduce another representation, this time more standard. This is the matrix representation of the ring S_q. Since S_q can be considered a vector space over $\mathbb{Z}_q$ by the usual polynomial embedding, we can associate an element a to its coefficient vector $\mathbf{a}$. We can also associate an element b to a $n \times n$ matrix $\mathbf{M}_b$ over $\mathbb{Z}_q$ such that the vector

$$\mathbf{c} = \mathbf{M}_b \cdot \mathbf{a}$$

is the coefficient vector of c where $c = a \cdot b$. This representation, which associates an element in S_q to a matrix, is called the matrix representation.

Recall we want to bootstrap $\ell^{(R)}/n$ ciphertexts in one go. We let recall the maps red and rep from Section 3 and define $\tau = \mathsf{red} \circ \mathsf{rep}$ to be the reduction modulo p map on $\mathbb{Z}_q^+$. To do this we can first extend rep and τ to the whole of S_q^+ by linearity, with images in $\mathbb{G}^n$ and $\mathbb{F}_p^n$ respectively. Similarly, we can extend rep and τ to $S_q^{\ell^{(R)}/n}$ to obtain maps $\overline{\mathsf{rep}} : (S_q^+)^{\ell^{(R)}/n} \longrightarrow \mathbb{G}^{\ell^{(R)}}$ and $\overline{\tau} : (S_q^+)^{\ell^{(R)}/n} \longrightarrow \mathbb{F}_p^{\ell^{(R)}}$, as in Section 3. Again this induces a map $\overline{\mathsf{red}}$, which is just the SIMD evaluation of red on the image of $\overline{\mathsf{rep}}$ in $\mathbb{G}^{\ell^{(R)}}$. We let $\overline{\mathsf{rep}}_{j,i}$ denote the restriction of $\overline{\mathsf{rep}}$ to the $(i-1)$th coefficient of the j-th S_q component, for $1 \leq i \leq n$ and $1 \leq j \leq \ell^{(R)}/n$.

We can then rewrite the decryption equation of our $\ell^{(R)}/n$ ciphertexts as

$$\begin{aligned}\left(\left(c_0^{(j)} + \mathfrak{sk}^{(S)} \cdot c_1^{(j)} \pmod q\right) \pmod p\right)_{j=1}^{\ell^{(R)}/n} & \\ = \overline{\mathsf{red}}\left(\overline{\mathsf{rep}}\left(c_0^{(1)} + \mathfrak{sk}^{(S)} \cdot c_1^{(1)}, \ldots, c_0^{(\ell^{(R)}/n)} + \mathfrak{sk}^{(S)} \cdot c_1^{(\ell^{(R)}/n)}\right)\right) & \\ = \overline{\mathsf{red}}\left(\overline{\mathsf{rep}}\left(\mathbf{x}\right)\right), &\end{aligned}$$

where $\mathbf{x}$ is the vector consisting of S_q elements $c_0^{(j)} + \mathfrak{sk}^{(S)} \cdot c_1^{(j)}$, for $j = 1, \ldots, \ell^{(R)}/n$. Thus, if we can compute $\overline{\mathsf{rep}}(\mathbf{x})$. then to perform the bootstrap we need only evaluate (in $\ell^{(R)}$-fold SIMD fashion) the arithmetic circuit of multiplicative depth

$$\lceil \log_2 d \rceil$$

representing $\overline{\mathsf{red}}$. Since we have enough slots, $\ell^{(R)}$, in the large plain text ring, we are able to do this homomorphically on fully packed ciphertexts. The total number of monomials in the arithmetic circuit (i.e. the multiplications we would need to evaluate $\overline{\mathsf{red}}$) being $D(E, d)$.

6.1 Homomorphically Evaluating $\overline{\mathsf{rep}}(\mathbf{x})$

We wish to homomorphically evaluate $\overline{\mathsf{rep}}(\mathbf{x})$ such that the output is a set of E ciphertexts and if we took the $i + (j-1) \cdot \ell^{(R)}/n$th slot of each plaintext we would obtain the E values which represent $\overline{\mathsf{rep}}_{j,i}(\mathbf{x})$. Let $\lambda = \lceil \log q / \log p \rceil$. We add to the public key of the SHE scheme the encryption of $\overline{\mathsf{rep}}(p^k \cdot \mathfrak{sk}^{(S)}, \ldots, p^k \cdot \mathfrak{sk}^{(S)})$ for $k = 0, \ldots, \lambda$ (where each component is copied $\ell^{(R)}/n$ times). For a given k this is a set of E ciphertexts, such that if we took the $i + (j-1) \cdot \ell^{(R)}/n$th slot of each plaintext we would obtain the E values which represent $\overline{\mathsf{rep}}_{j,i}(p^k \cdot \mathfrak{sk}^{(S)})$. Let the resulting vector of ciphertexts be denoted $\mathfrak{ct}_k$, for $k = 1, \ldots, \lambda$, where $\mathfrak{ct}_k$ is a vector of length E.

Let $\mathbf{M}_{c_1^{(j)}}$ be the matrix representation of the second ciphertext component $c_1^{(j)}$ of the j-th ciphertext that we want to bootstrap. We write

$$\mathbf{M}_{c_1^{(j)}} = \sum_{k=0}^{\lambda} p^k \cdot \mathbf{M}_1^{(j,k)}$$

where $\mathbf{M}_1^{(j,k)}$ is a matrix with coefficients in $\{0, \ldots, p-1\}$. We then have that

$$\begin{aligned} c_0^{(j)} + \mathfrak{sk}^{(S)} \cdot c_1^{(j)} &= c_0^{(j)} + \sum_{k=0}^{\lambda} \left(p^k \cdot \mathbf{M}_1^{(j,k)} \cdot \underline{\mathfrak{sk}}^{(S)}\right) \\ &= c_0^{(j)} + \sum_{k=0}^{\lambda} \left(\mathbf{M}_1^{(j,k)} \cdot (p^k \cdot \underline{\mathfrak{sk}}^{(S)})\right) \end{aligned}$$

where $\underline{\mathfrak{sk}}^{(S)}$ is the vector of coefficients of the secret key $\mathfrak{sk}^{(S)}$.

We let $\mathbf{M}_1^{(k)} = \bigoplus_{j=1}^{\ell^{(R)}/n} \mathbf{M}_1^{(j,k)} = \mathbf{diag}(\mathbf{M}_1^{(1,k)}, \ldots, \mathbf{M}_1^{(\ell^{(R)}/n,k)})$. We now apply $\overline{\mathsf{rep}}$ to both sides, which means we need to compute homomorphically the ciphertext which represents

$$\overline{\mathsf{rep}}\left(c_0^{(1)}, \ldots, c_0^{(\ell^{(R)}/n)}\right) \cdot \prod_{k=0}^{\lambda} \overline{\mathsf{rep}}\left(p^k \cdot \underline{\mathfrak{sk}}^{(S)}, \ldots, p^k \cdot \underline{\mathfrak{sk}}^{(S)}\right)^{\mathbf{M}_1^{(k)}}.$$

We are thus in the situation described in Section 5. Thus the homomorphic evaluation of $\overline{\mathsf{rep}}(\mathbf{x})$ requires a depth of

$$\frac{1}{2} + \lceil \log_2 \ell^{(R)} \rceil + \lceil \log_2 p \rceil + \lceil \log_2 \lceil \log_2 p \rceil \rceil + \lceil \log_2 \lambda \rceil$$

and

$$\lambda \cdot \left(1 + \ell^{(R)} \cdot (1 + E + 2 \cdot M \cdot \lceil \log_2 p \rceil)\right)$$

multiplications.

6.2 Repacking

At this point in the bootstrapping procedure (assuming for simplicity that a ring switch has not occured) we have a single ciphertext $\mathfrak{ct}$ whose $\ell^{(R)}$ slots encode the coefficients (over the small ring) of the $\ell^{(R)}/n$ ciphertexts that we are bootstrapping. Our task is now to extract these coefficients to produce a ciphertext (or set of ciphertexts) which encode the same data. Effectively this is the task of performing $\ell^{(R)}/n$ inverse Fourier transforms (a.k.a interpolations) over S in parallel, and then encoding the result as elements in R via the embedding $\iota : S \longrightarrow R$.

There are a multitude of ways of doing this step (bar performing directly an inverse FFT algorithm), for example the general method of Alperin-Sheriff and Peikert [1] could be applied. This makes the observation that the FFT to a vector of Fourier coefficients $\mathbf{x}$ is essentially applying a linear operation, and hence we can compute it by taking the trace of a value $\alpha \cdot \mathbf{x}$ for some fixed constant α.

We select a more naive, and simplistic approach. Suppose $\mathbf{x}$ is the vector which is encoded by the input ciphertext. We first homomorphically compute

$$\mathbf{b}_1, \ldots, \mathbf{b}_{\ell^{(R)}} = \mathsf{replicate}(\mathbf{x}).$$

Where $\mathsf{replicate}(\mathbf{x})$ is the Full Replication algorithm from [17]. This produces $\ell^{(R)}$ ciphertexts, the ith of which encodes the constant polynomial over R_p equal to the i slot in $\mathbf{x}$. In [17] this is explained for the case where $\ell^{(R)} = N$, but the method clearly works when $\ell^{(R)} < N$. The method requires time $O(\ell^{(R)})$ and depth $O(\log \log \ell^{(R)})$.

Given the output $\mathbf{b}_1, \ldots, \mathbf{b}_{\ell^{(R)}}$, which encode the coefficients of the $\ell^{(R)}/n$ original plaintext vectors, we can now apply ι (which recall is a linear map) to obtain *any* linear function of the underlying plaintexts. For example we could produce $\ell^{(R)}/n$ ciphertexts each of which encodes one of the original plaintexts, or indeed a single ciphertext which encodes all of them.

So putting all of the sub-procedures for bootstrapping together, we find that we can bootstrap $\ell^{(R)}/n$ ciphertexts in parallel using a procedure of depth of

$$\lceil \log_2 d \rceil + \frac{1}{2} + \lceil \log_2 \ell^{(R)} \rceil + \lceil \log_2 p \rceil + \lceil \log_2 \lceil \log_2 p \rceil \rceil + \lceil \log_2 \lambda \rceil + O(\log_2 \log_2 \ell^{(R)})$$

and

$$D(E,d) + \lambda \cdot \Big(1 + \ell^{(R)} \cdot (1 + E + 2 \cdot M \cdot \lceil \log_2 p \rceil)\Big) + O(\ell^{(R)})$$

multiplications, where $d \approx \log q / \log E$, $E = \sum_{i=1}^{t} k_i$ and $M = 8 \cdot \sum_{i=1}^{t} k_i \cdot (k_i - 1)$.

7 Example Parameters

In Appendix A we present a calculation of suitable parameters for our scheme, and the polynomial representation of red. We target $\kappa = 128$-bits of security, and set the Hamming weight h of the secret key $\mathfrak{sk}$ to be 64 as in [15, 9].

On input N and n the to the formulae in Appendix A we obtain an upper bounds on $\log(Q_{L-1})$ and $\log(Q_{L_2})$. We now use equations (3)-(8) from the Appendix for different values of the plaintext modulus p to obtain a lower bound on $\log(Q_{L-1})$ and $\log(Q_{L_2})$. Then, we increase N and n until the lower bound on Q_{L-1} and Q_{L_2} from the functionality is below the upper bound from the security analysis. In this way we obtain lower bounds for N and n.

Table 1. Lower bounds on N and n

p	κ	$c = \ell^{(R)}/n$	$n \approx$	$N \approx$	$q \approx$
2	128	1	860	22600	11637
		2		23600	
$\approx 2^8$	128	1	1040	49600	1635087
		2, 3		51100	
		4, 5		52000	
		$[6, \ldots, 10]$		54100	
$\approx 2^{16}$	128	1	1300	81460	467989106
		2, 3		83460	
		$[4, \ldots, 10]$		87460	
$\approx 2^{32}$	128	1	1750	156000	$3.558467651 \cdot 10^{13}$
		2		158000	
		$[3, \ldots, 10]$		163000	

In Table 1 we consider four different values of p, and assume $c_{m'} \leq 2$; for simplicity we also set $t = 1$ in (1), i.e. $\mathbb{G} = \mathbb{F}^*_{p^k}$, for a suitable choice of k. After finding approximate values for N, n and q we can then search for exact values of N, n and q. More precisely, we are looking for cyclotomic rings R and S such that the degree $N = \phi(m)$ of $F(X) = \Phi_m(X)$ and $n = \phi(m')$ of $f(x) = \Phi_{m'}(X)$ are larger than the bounds above and n divides both N and $\ell^{(R)}$ (the number of plaintext slots associated with R). In addition we require that q divides $p^k - 1$. See Table 2 for some values.

Notice that the value of q is strongly influenced by the ring constant $c_{m'}$. In Table 1 we set $c_{m'} = 1.28$ (i.e. we assume the best case of m' being prime), whereas in Table 2 we compute the actual value of the ring constant for each cyclotomic ring we consider. For example for $p = 2$, in Table 1 we obtain an approximate value $q \approx 11637$, but in Table 2 we need a larger value due to the additional condition that q divides $p^k - 1$, and the ring constant, which is bigger than 1.27 for $m' = 1271$ and $m' = 1057$.

8 Elliptic Curves Based Variant

We now extend our algorithm from representations in finite fields to representations in elliptic curve groups. Recall we need to embed $\mathbb{Z}_q^+$ into a group defined over $\mathbb{F}_p$ whose operations can be expressed in terms of

Table 2. A concrete set of cyclotomic rings with an estimation of the number of multiplications and the depth required to perform our bootstrapping step

p	m	$N = \phi(m)$	m'	$n = \phi(m')$	$(d^{(R)}, \ell^{(R)})$	$c_{m'}$	$\ell^{(R)}/n$	k	L	d	# Mults	q
2	31775	24000	1271	1200	$(20, 1200)$	3.93	1	16	24	4	4940446	65535
	32767	27000	1057	900	$(15, 1800)$	2.69	2	15	24	4	6107675	32767
$2^8 + 1$	62419	51840	1687	1440	$(36, 1440)$	2.72	1	3	36	14	226905	4243648
	81651	51200	1601	1600	$(16, 3200)$	1.28	2	3	37	14	503403	2121824
	70655	52128	1087	1086	$(12, 4344)$	1.28	4	3	38	14	683126	2121824
$2^{16} + 1$	104797	81600	1361	1360	$(60, 1360)$	1.28	1	2	45	29	97025	536887296
	156891	86400	1687	1440	$(30, 2880)$	2.72	2	2	46	30	204976	1073774592
	98087	86400	2651	2400	$(6, 14400)$	2.9	6	2	48	30	1022929	2147549184
$2^{32} + 15$	262251	158760	2649	1764	$(90, 1764)$	1.81	1	2	61	46	239267	50637664608480
	164557	162396	2083	2082	$(39, 4164)$	1.28	2	2	63	46	563267	414161297767368
	175933	162240	2227	2080	$(26, 6240)$	3.56	3	2	63	47	843575	115044804935380

the functionality of the homomorphic encryption scheme. This means that the range of the representation should be an algebraic group. We have already seen linear algebraic groups (a.k.a. matrix representations) used in this context in work of Alperin-Sherriff and Peikert, thus as it is natural (to anyone who has studied algebraic groups) to consider algebraic varieties. The finite field case discussed in the previous sections corresponds to the genus zero case, thus the next natural extension would be to examine the genus one case (a.k.a. elliptic curves).

The reason for doing this is the value of q from Table 2 compared to the estimated values from Table 1 are far from optimal. This is because we have few possible group orders of $\mathbb{F}^*_{p^{k_i}}$. The standard trick in this context (used for example in the ECM factorization method, the ECPP primality prover, or even indeed in all of elliptic curve cryptography) is to replace the multiplicative group of a finite field by an elliptic curve group.

Just as before we select a coprime factorization $q = \prod_{i=1}^{t} e_i$ (with the e_i not necessarily prime, but pairwise coprime). But now we require that e_i divides the order of an elliptic curve E_i defined over p^{k_i}. Since the group orders of elliptic curves are distributed roughly uniformly within the Hasse interval it is highly likely that there are such elliptic curves. Determining such curves may however be a hard problem for a fixed value of q; a problem which arose previously in cryptography in [3]. However, since we have some freedom in selecting q in our scheme we can select q and the E_i simultaneously, and hence finding the elliptic curves will not be a problem.

Again, we fix a polynomial representation of $\mathbb{F}_{p^{k_i}}$, i.e. an irreducible polynomial $f_i(x)$ of degree k_i such that $\mathbb{F}_{p^{k_i}} = \mathbb{F}_p[x]/f_i(x)$, and now we let $G_i \in E_i(\mathbb{F}_{p^{k_i}})$ denote a fixed point on the elliptic curve of order e_i. We now can translate our method into this new setting. For example Equation (1) translates to

$$\mathsf{rep} : \begin{cases} \mathbb{Z}_q^+ & \longrightarrow \mathbb{G} = \prod_{i=1}^{t} E_i(\mathbb{F}_{p^{k_i}}) \\ a & \longmapsto ([a_1]G_1, \ldots, [a_t]G_t) \end{cases} \tag{2}$$

where $a_i = a \pmod{e_i}$.

Homomorphic calculations in $\mathbb{G}$ are then performed using Jacobian Projective coordinates. This means that general point addition can be performed with multiplicative depth five and $M' = 16 \cdot M$ homomorphic multiplications. Our method then proceeds as before, except we replace homomorphic multiplication in $\mathbb{F}^*_{p^{k_i}}$ with Jacobian projective point addition in $E_i(\mathbb{F}_{p^{k_i}})$.

The computation of red is then performed as follows. We first homomorphically map the projective points in $\mathbb{G}$ into an affine point. Each such conversion, in component i, requires an $\mathbb{F}_{p^{k_i}}$-field inversion and

three $\mathbb{F}_{p^{k_i}}$-field multiplications. If we let DInv_i (resp. MInv_i) denote the depth (resp. number of multiplications in $\mathbb{F}_p$) of the circuit to invert in the field $\mathbb{F}_{p^{k_i}}$. This implies that the conversion of a set of projective points in $\mathbb{G}$ to set of affine points requires depth $\max_{i=1}^{t} \mathsf{DInv}_i$ and $\sum_{i=1}^{t}(\mathsf{MInv}_i + 3 \cdot M)$ homomorphic multiplications over $\mathbb{F}_p$.

Given this conversion we now have effectively $E' = E + t$ as opposed to E variables defining the element in G. The extra t variables coming from an the y-coordinate; it is clear we only need to store t such variables as opposed to E such variables as each x coordinate corresponds to at most two y-coordinates and hence a naive form of homomorphic point compression can be applied.

This means the map red (after the conversion to affine coordinates) can be expressed as a degree d' map; where we expect d' to be the smallest d' such that $\sum_{i=1}^{d'} {}^{E'}C_i > q$, which means we expect $d' \approx \log q / \log E'$. This means, as before, that the resulting depth will be $\lceil \log_2 d' \rceil$ and the number of multiplications will be $D(E', d')$.

So putting all of the sub-procedures for bootstrapping together, we find that we can use the elliptic curve variant of our bootstrapping method to bootstrap $\ell^{(R)}/n$ ciphertexts in parallel using a procedure of depth of

$$\lceil \log_2 d' \rceil + 5 \cdot \left(\frac{1}{2} + \lceil \log_2 \ell^{(R)} \rceil + \cdot \lceil \log_2 p \rceil + \cdot \lceil \log_2 \lceil \log_2 p \rceil \rceil + \lceil \log_2 \lambda \rceil \right) + \max_{i=1}^{t} \mathsf{DInv}_i + O(\log_2 \log_2 \ell^{(R)})$$

and

$$D(E', d') + \lambda \cdot \left(8 + \ell^{(R)} \cdot (8 + 3 \cdot E' + 2 \cdot M' \cdot \lceil \log_2 p \rceil) \right) + \sum_{i=1}^{t} (\mathsf{MInv}_i + 3 \cdot M) + O(\ell^{(R)})$$

multiplications, where $d' \approx \log q / \log E'$, $E' = \sum_{i=1}^{t}(k_i + 1)$, $M = \sum_{i=1}^{t} k_i \cdot (k_i - 1)/2$ and $M' = 16 \cdot M$.

However, the ability to use arbitrary q comes at a penalty; the depth required has dramatically increased due to the elliptic curve group operations. For example if we consider a prime p of size roughly 2^{16} and $k = 2$, then we need about 82 levels, as opposed to 46 with the finite field variant. This then strongly influences the required value of N, pushing it up from around $85,000$ to $220,000$. Thus in practice the elliptic curve variant is unlikely to be viable.

9 Acknowledgements

This work has been supported in part by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO, by EPSRC via grant EP/I03126X, and by Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number FA8750-11-2-0079[1].

References

1. J. Alperin-Sheriff and C. Peikert. Practical bootstrapping in quasilinear time. In *CRYPTO*, volume 8042 of *Lecture Notes in Computer Science*, pages 1–20, 2013.
2. J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. Cryptology ePrint Archive, Report 2014/094, 2014.

3. D. Boneh and R.J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 283–297, 1996.
4. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325. ACM, 2012.
5. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE, 2011.
6. Z. Brakerski and V. Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12, 2014.
7. Y. Chen and P.Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20, 2011.
8. A. Choudhury, J. Loftus, E. Orsini, A. Patra, and N.P. Smart. Between a rock and a hard place: Interpolating between MPC and FHE. In *ASIACRYPT*, volume 8270 of *Lecture Notes in Computer Science*, pages 221–240, 2013.
9. I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure MPC for dishonest majority – or: Breaking the SPDZ limits. In *ESORICS*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18, 2013.
10. I. Damgård, V. Pastro, N.P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, 2012.
11. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, 2009.
12. C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148, 2011.
13. C. Gentry, S. Halevi, C. Peikert, and N.P. Smart. Field switching in BGV-style homomorphic encryption. *Journal of Computer Security*, 21(5):663–684, 2013.
14. C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16, 2012.
15. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867, 2012.
16. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, 2013.
17. S. Halevi and V. Shoup. Algorithms in HElib. Cryptology ePrint Archive, Report 2014/106, 2014.
18. T. Lepoint and M. Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. Cryptology ePrint Archive, Report 2014/62, 2014.
19. R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339, 2011.
20. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, 2010.
21. D. Micciancio and O. Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.
22. K. Rohloff and D.B. Cousins. A scalable implementation of fully homomorphic encryption built on NTRU, 2014.
23. N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *PKC*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443, 2010.
24. N.P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71:57–81, 2014.
25. J. van de Pol and N.P. Smart. Estimating key sizes for high dimensional lattice-based systems. In *IMA Int. Conf.*, volume 8308 of *Lecture Notes in Computer Science*, pages 290–303, 2013.

A Parameter Calculation

In [15] a concrete set of parameters for the BGV SHE scheme was given for the case of binary message spaces, and arbitrary L. In [9] this was adapted to the case of message space R_p for 2-power cyclotomic rings, but only for the schemes which could support one level of multiplication gates (i.e. for $L = 1$). In [8] these two approaches were combined, for arbitrary L and p, and the analysis was (slightly) modified to remove the need for a modulus switching upon encryption. In this section we modify again the analysis of [8] to present an analysis which includes a step of field switching from [13]. We assume in this section that the reader is familiar with the analysis and algorithms from [15, 8, 13].

Our analysis will make extensive use of the following fact: If $a \in R$ be chosen from a distribution such that the coefficients are distributed with mean zero and standard deviation σ, then if ζ_m is a primitive mth root of unity, we can use $6 \cdot \sigma$ to bound $a(\zeta_m)$ and hence the canonical embedding norm of a. If we have two elements with variances σ_1^2 and σ_2^2, then we can bound the canonical norm of their product with $16 \cdot \sigma_1 \cdot \sigma_2$.

Ensuring We Can Evaluate the Required Depth: Recall we have two rings R and S of degree N and n respectively. The ring S is a subring of R and hence n divides N. We require a chain of moduli $q_0 < q_1 \ldots < q_L$ corresponding to each level of the scheme. We assume (for sake of simplicity) that $q_i/q_{i-1} = p_i$ are primes. Thus $q_L = q_0 \cdot \prod_{i=1}^{i=L} p_i$. Also note, that as in [8], we apply a $\mathsf{SHE.LowerLevel}$ (a.k.a. modulus switch) algorithm *before* a multiplication operation. This often leads to lower noise values in practice (which a practical instantiation can make use of). In addition it eliminates the need to perform a modulus switch after encryption, which happened in [15].

We utilize the following constants described in [9], which are worked out for the case of message space defined modulo p (the constants in [9] make use of an additional parameter, arising from the key generation procedure. In our case we can take this constant equal to one). In the following h is the Hamming weight of the secret keys $\mathfrak{sk}^{(R)}$ and $\mathfrak{sk}^{(S)}$.

$$
\begin{aligned}
B_{\mathsf{Clean}} =& N \cdot p/2 + p \cdot \sigma \cdot \left(\frac{16 \cdot N}{\sqrt{2}} + 6 \cdot \sqrt{N} + 16 \cdot \sqrt{h \cdot N} \right) \\
B^{(R)}_{\mathsf{Scale}} =& p \cdot \sqrt{3 \cdot N} \cdot \left(1 + \frac{8}{3} \cdot \sqrt{h} \right) \\
B^{(S)}_{\mathsf{Scale}} =& p \cdot \sqrt{3 \cdot n} \cdot \left(1 + \frac{8}{3} \cdot \sqrt{h} \right) \\
B^{(R)}_{\mathsf{Ks}} =& p \cdot \sigma \cdot N \cdot \left(1.49 \cdot \sqrt{h \cdot N} + 2.11 \cdot h + 5.54 \cdot \sqrt{h} + 1.96\sqrt{N} + 4.62 \right) \\
B^{(S)}_{\mathsf{Ks}} =& p \cdot \sigma \cdot n \cdot \left(1.49 \cdot \sqrt{h \cdot n} + 2.11 \cdot h + 5.54 \cdot \sqrt{h} + 1.96\sqrt{n} + 4.62 \right)
\end{aligned}
$$

As in [15] we define a small "wiggle room" ξ which we set to be equal to eight; this is set to enable a number of additions to be performed without needing to individually account for them in our analysis. These constants arise in the following way:

- A freshly encrypted ciphertext at level L has noise bounded by B_{Clean}.
- In the worst case, when applying $\mathsf{SHE.LowerLevel}$ to a (big ring) ciphertext at level $\mathfrak{l} > L_2 + 1$ with noise bounded by B' one obtains a new ciphertext at level $\mathfrak{l} - 1$ with noise bounded by

$$\frac{B'}{p_{\mathfrak{l}}} + B^{(R)}_{\mathsf{Scale}}.$$

- In the worst case, when applying $\mathsf{SHE.LowerLevel}$ to a (small ring) ciphertext at level $\mathfrak{l} \leq L_2 + 1$ with noise bounded by B' one obtains a new ciphertext at level $\mathfrak{l} - 1$ with noise bounded by

$$\frac{B'}{p_{\mathfrak{l}}} + B^{(S)}_{\mathsf{Scale}}.$$

- When applying the tensor product multiplication operation to (big ring) ciphertexts of a given level $\mathfrak{l} > L_2 + 1$ of noise B_1 and B_2 one obtains a new ciphertext with noise given by

$$B_1 \cdot B_2 + \frac{B^{(R)}_{\mathsf{Ks}} \cdot q_{\mathfrak{l}}}{P_R} + B^{(R)}_{\mathsf{Scale}},$$

  where P_R is a value to be determined later.

– When applying the tensor product multiplication operation to (small ring) ciphertexts of a given level $\mathfrak{l} \leq L_2$ of noise B_1 and B_2 one obtains a new ciphertext with noise given by

$$B_1 \cdot B_2 + \frac{B_{\mathsf{Ks}}^{(S)} \cdot q_{\mathfrak{l}}}{P_S} + B_{\mathsf{Scale}}^{(S)},$$

where again P_S is a value to be determined later.

A general evaluation procedure begins with a freshly encrypted ciphertext at level L with noise B_{Clean}. When entering the first multiplication operation we first apply a SHE.LowerLevel operation to reduce the noise to a universal bounds. $B^{(R)}$, whose value will be determined later. We therefore require

$$\frac{\xi \cdot B_{\mathsf{Clean}}}{p_L} + B_{\mathsf{Scale}}^{(R)} \leq B^{(R)},$$

i.e.

$$p_L \geq \frac{8 \cdot B_{\mathsf{Clean}}}{B^{(R)} - B_{\mathsf{Scale}}^{(R)}}. \tag{3}$$

We now turn to dealing with the SHE.LowerLevel operations which occurs before a multiplication gate at level $\mathfrak{l} \in \{1, \ldots, L-1\} \setminus \{L_2 + 1\}$. In what follows we assume $\mathfrak{l} > L_2 + 1$, to obtain the equations for $\mathfrak{l} \leq L_2$ one simply replaces the R-constants by their equivalent S-constants. We perform a worst case analysis and assume that the input ciphertexts are at level $\mathfrak{l}$. We can then assume that the input to the tensoring operation in the previous multiplication gate (just after the previous SHE.LowerLevel) was bounded by $B^{(R)}$, and so the output noise from the previous multiplication gate for each input ciphertext is bounded by $(B^{(R)})^2 + B_{\mathsf{Ks}}^{(R)} \cdot q_{\mathfrak{l}}/P_R + B_{\mathsf{Scale}}^{(R)}$. This means the noise on entering the SHE.LowerLevel operation is bounded by ξ times this value, and so to maintain our invariant we require

$$\frac{\xi \cdot (B^{(R)})^2 + \xi \cdot B_{\mathsf{Scale}}^{(R)}}{p_{\mathfrak{l}}} + \frac{\xi \cdot B_{\mathsf{Ks}}^{(R)} \cdot q_{\mathfrak{l}}}{P_R \cdot p_{\mathfrak{l}}} + B_{\mathsf{Scale}}^{(R)} \leq B^{(R)}.$$

Rearranging this into a quadratic equation in $B^{(R)}$ we have

$$\frac{\xi}{p_{\mathfrak{l}}} \cdot (B^{(R)})^2 - B^{(R)} + \left(\frac{\xi \cdot B_{\mathsf{Scale}}^{(R)}}{p_{\mathfrak{l}}} + \frac{\xi \cdot B_{\mathsf{Ks}}^{(R)} \cdot q_{\mathfrak{l}-1}}{P_R} + B_{\mathsf{Scale}}^{(R)} \right) \leq 0.$$

We denote the constant term in this equation by $R_{\mathfrak{l}-1}$. We now assume that all primes $p_{\mathfrak{l}}$ are of roughly the same size (for the ring R), and noting the we need to only satisfy the inequality for the largest modulus $\mathfrak{l} = L - 1$ (resp. $\mathfrak{l} = L_2$ for the ring S). We now fix R_{L-2} by trying to ensure that R_{L-2} is close to $B_{\mathsf{Scale}}^{(R)} \cdot (1 + \xi/p_{L-1}) \approx B_{\mathsf{Scale}}^{(R)}$, so we set $R_{L-2} = (1 - 2^{-3}) \cdot B_{\mathsf{Scale}}^{(R)} \cdot (1 + \xi/p_{L-1})$, and obtain

$$P_R \approx 8 \cdot \frac{\xi \cdot B_{\mathsf{Ks}}^{(R)} \cdot q_{L-2}}{B_{\mathsf{Scale}}^{(R)}}, \tag{4}$$

since $B_{\mathsf{Scale}}^{(R)} \cdot (1 + \xi/p_{L-1}) \approx B_{\mathsf{Scale}}^{(R)}$. Similarly for the small ring we find

$$P_S \approx 8 \cdot \frac{\xi \cdot B_{\mathsf{Ks}}^{(S)} \cdot q_{L_2-1}}{B_{\mathsf{Scale}}^{(S)}}, \tag{5}$$

To ensure we have a solution we require $1 - 4 \cdot \xi \cdot R_{L-2}/p_{L-1} \geq 0$, (resp. $1 - 4 \cdot \xi \cdot R_{L_2-1}/p_{L_2} \geq 0$) which implies we should take, for $i = 2, \ldots, L-1$,

$$p_i \approx \begin{cases} 4 \cdot \xi \cdot R_{L-2} \approx 32 \cdot B_{\mathsf{Scale}}^{(R)} = p_R & \text{For } i = L_2 + 2, \ldots, L-1, \\ 4 \cdot \xi \cdot R_{L_2-1} \approx 32 \cdot B_{\mathsf{Scale}}^{(S)} = p_S & \text{For } i = 1, \ldots, L_2. \end{cases} \tag{6}$$

We now examine what happens at level $L_2 + 1$ when we perform a ring switch operation. Following Lemma 3.2 of [13] we know the noise increases by a factor of $(p/2) \cdot \sqrt{N/n}$. The noise output from the previous multiplication gate is bounded by $(B^{(R)})^2 + B_{\mathsf{Ks}}^{(R)} \cdot q_{L_2+2}/P_R + B_{\mathsf{Scale}}^{(R)}$. Note that

$$\begin{aligned} \frac{B_{\mathsf{Ks}}^{(R)} \cdot q_{L_2+2}}{P_R} &\approx \frac{B_{\mathsf{Ks}}^{(R)} \cdot q_{L_2+2} \cdot B_{\mathsf{Scale}}^{(R)}}{8 \cdot \xi \cdot B_{\mathsf{Ks}}^{(R)} \cdot q_{L-2}} \\ &\approx \frac{B_{\mathsf{Scale}}^{(R)}}{8 \cdot \xi \cdot p_R^{L_1-4}} \end{aligned}$$

Thus the we know that the noise after the ring switch operation is bounded by

$$B_{\mathsf{RingSwitch}} = \frac{p}{2} \cdot \sqrt{N/n} \cdot \left((B^{(R)})^2 + \frac{B_{\mathsf{Scale}}^{(R)}}{8 \cdot \xi \cdot p_R^{L_1-4}} + B_{\mathsf{Scale}}^{(R)} \right).$$

We now modulus switch down to level L_2, and obtain a ciphertext (over the ring S) with noise bounded by

$$\frac{B_{\mathsf{RingSwitch}}}{p_{L_2+1}} + B_{\mathsf{Scale}}^{(S)}.$$

We would like this to be less than the universal bound $B^{(S)}$, which implies

$$p_{L_2+1} \geq \frac{B_{\mathsf{RingSwitch}}}{B^{(S)} - B_{\mathsf{Scale}}^{(S)}}. \tag{7}$$

We now need to estimate the size of p_0. Due to the above choices the ciphertext to which we apply the bootstrapping has norm bound by $B^{(S)}$. This means that we require

$$q_0 = p_0 \geq 2 \cdot B^{(S)} \cdot c_{m'}, \tag{8}$$

to ensure a valid decryption/bootstrapping procedure. Recall $c_{m'}$ is the ring constant for the polynomial ring S and it depends only on m' (see [10] for details).

Ensuring We Have Security: The works before [25, 18], such as Lindner and Peikert [19], did not include the rank of the lattice into account when estimating the cost of the attacker. The reason is that the lattice rank appears to be only a second order term in the cost of the attack. However, for applications such as FHE, the dimension is usually very big, e.g. 2^{16}, and lattice algorithms are often polynomial in the rank. Therefore, even as a second order term it can contribute significantly to the cost of the attack. The largest modulus used in our big ring (resp. small ring) key switching matrices, i.e. the largest modulus used in an LWE instance, is given by $Q_{L-1} = P_R \cdot q_{L-1}$ (resp. $Q_{L_2} = P_S \cdot q_{L_2}$).

We recall the approach of [25, 18] here. First, fix some security level as measured in enumeration nodes, e.g. 2^{128}. Now, use estimates by Chen and Nguyen [7] are used to determine the cost of running BKZ 2.0

for various block sizes β. Combining this with the security level gives an upper bound on the rounds an attacker can perform, depending on β. Then, for various lattice dimensions r, the BKZ 2.0 simulator by Chen and Nguyen is used to determine the quality of the vector as measured by the root-Hermite factor $\delta(\beta, r) = (\|\mathbf{b}\|/\mathrm{vol}(L)^{1/r})^{1/r}$. Now, the best possible root-Hermite factor achievable by the attacker is given by $\delta(r) = \min_\beta \delta(\beta, r)$

In LWE, the relevant parameters for the security are the ring dimension n (resp. N), the modulus $Q = Q_{L_2}$ (resp. $Q = Q_{L-1}$) and the standard deviation σ. Note that in most scenarios, an adversary can choose how many LWE samples he uses in his attack. This number r is equal to the rank of the lattice. The distinguishing attack against LWE uses a short vector in the dual SIS lattice to distinguish the LWE distribution from the uniform distribution. More precisely, an adversary can distinguish between these two distributions with distinguishing advantage ε if the shortest vector he can obtain (in terms of its root-Hermite factor) satisfies

$$\delta(r)^r \cdot Q^{n/r-1} \cdot \sigma < \sqrt{-\log(\varepsilon)/\pi}.$$

It follows that in order for our system to be secure against the previously described adversary, we need that

$$\log_2(Q) \leq \min_{r>n} \frac{r^2 \cdot \log_2(\delta(r)) + r \cdot \log_2(\sigma/\alpha)}{r-n}, \tag{9}$$

where $\alpha = \sqrt{-\log(\varepsilon)/\pi}$. See also[21, 19, 18] for more information. For every n we can now compute an upper bound on $\log_2(q)$ by iterating the right hand side of Equation (9) over m and selecting the minimum.

Putting it all together As in [15, 9], we set $\sigma = 3.2$, $B^{(R)} = 2 \cdot B^{(R)}_{\mathsf{Scale}}$ and $B^{(S)} = 2 \cdot B^{(S)}_{\mathsf{Scale}}$. From our equations (3), (4), (5), (6), (7), and (8) we obtain equations for p_i for $i = 0, \ldots, L$, P_R and P_S in terms of n, N, L, h and the security level κ.