Practical Bootstrapping in Quasilinear Time

Jacob Alperin-Sheriff* Chris Peikert†

July 3, 2013

Abstract

Gentry's "bootstrapping" technique (STOC 2009) constructs a fully homomorphic encryption (FHE) scheme from a "somewhat homomorphic" one that is powerful enough to evaluate its own decryption function. To date, it remains the only known way of obtaining unbounded FHE. Unfortunately, bootstrapping is computationally very expensive, despite the great deal of effort that has been spent on improving its efficiency. The current state of the art, due to Gentry, Halevi, and Smart (PKC 2012), is able to bootstrap "packed" ciphertexts (which encrypt up to a linear number of bits) in time only *quasilinear* $\tilde{O}(\lambda) = \lambda \cdot \log^{O(1)} \lambda$ in the security parameter. While this performance is *asymptotically* optimal up to logarithmic factors, the practical import is less clear: the procedure composes multiple layers of expensive and complex operations, to the point where it appears very difficult to implement, and its concrete runtime appears worse than those of prior methods (all of which have quadratic or larger asymptotic runtimes).

In this work we give *simple*, *practical*, and entirely *algebraic* algorithms for bootstrapping in quasilinear time, for both "packed" and "non-packed" ciphertexts. Our methods are easy to implement (especially in the non-packed case), and we believe that they will be substantially more efficient in practice than all prior realizations of bootstrapping. One of our main techniques is a substantial enhancement of the "ring-switching" procedure of Gentry et al. (SCN 2012), which we extend to support switching between two rings where neither is a subring of the other. Using this procedure, we give a natural method for homomorphically evaluating a broad class of structured linear transformations, including one that lets us evaluate the decryption function efficiently.

*School of Computer Science, College of Computing, Georgia Institute of Technology. Email: jmas6@cc.gatech.edu

†School of Computer Science, Georgia Institute of Technology. Email: cpeikert@cc.gatech.edu. This material is based upon work supported by the National Science Foundation under CAREER Award CCF-1054495, by the Alfred P. Sloan Foundation, and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the National Science Foundation, the Sloan Foundation, DARPA or the U.S. Government.

1 Introduction

Bootstrapping, a central technique from the breakthrough work of Gentry [Gen09b, Gen09a] on fully homomorphic encryption (FHE), converts a sufficiently powerful "somewhat homomorphic" encryption (SHE) scheme into a fully homomorphic one. (An SHE scheme can support a bounded number of homomorphic operations on freshly generated ciphertexts, whereas an FHE scheme has no such bound.) In short, bootstrapping works by *homomorphically* evaluating the SHE scheme's decryption function on a ciphertext that cannot support any further homomorphic operations. This has the effect of "refreshing" the ciphertext, i.e., it produces a new one that encrypts the same message and can handle more homomorphic operations. Bootstrapping remains the only known way to achieve *unbounded* FHE, i.e., a scheme that can homomorphically evaluate any efficient function using keys and ciphertexts of a fixed size.[1]

In order to be "bootstrappable," an SHE scheme must be powerful enough to homomorphically evaluate its own decryption function, using whatever homomorphic operations it supports. For security reasons, the key and ciphertext sizes of all known SHE schemes grow with the *depth* and, to a lesser extent, the *size* of the functions that they can homomorphically evaluate. For instance, under plausible hardness conjectures, the key and ciphertext sizes of the most efficient SHE scheme to date [BGV12] grow quasilinearly in both the supported multiplicative depth d and the security parameter λ, i.e., as $\tilde{O}(d \cdot \lambda)$. Clearly, the runtime of bootstrapping must also grow with the sizes of the keys, ciphertexts, and decryption function. This runtime is perhaps the most important measure of efficiency for FHE, because bootstrapping is currently the biggest bottleneck by far in instantiations, both in theory and in practice.

The past few years have seen an intensive study of different forms of decryption procedures for SHE schemes, and their associated bootstrapping operations [Gen09b, Gen09a, vDGHV10, GH11b, BV11a, GH11a, BGV12, GHS12b]. The first few bootstrapping methods had moderate polynomial runtimes in the security parameter λ, e.g., $\tilde{O}(\lambda^4)$. Brakerski, Gentry, and Vaikuntanathan [BGV12] gave a major efficiency improvement, reducing the runtime to $\tilde{O}(\lambda^2)$. They also gave an amortized method that bootstraps $\tilde{\Omega}(\lambda)$ ciphertexts at once in $\tilde{O}(\lambda^2)$ time, i.e., quasilinear runtime per ciphertext. However, these results apply only to "non-packed" ciphertexts, i.e., ones that encrypt essentially just one bit each, which combined with the somewhat large runtimes makes these methods too inefficient to be used very much in practice. Most recently, Gentry, Halevi, and Smart [GHS12a] achieved bootstrapping for "packed" ciphertexts (i.e., ones that encrypt up to $\tilde{\Omega}(\lambda)$ bits each) in *quasilinear* $\tilde{O}(\lambda)$ runtime, which is asymptotically optimal in space and time, up to polylogarithmic factors. For this they relied on a general "compiler" from another work of theirs [GHS12b], which achieved SHE/FHE for sufficiently wide circuits with polylogarithmic multiplicative "overhead," i.e., cost relative to evaluating the circuit "in the clear."

Bootstrapping and FHE in quasi-optimal time and space is a very attractive and powerful theoretical result. However, the authors of [GHS12b, GHS12a] caution that their constructions may have limited potential for use in practice, for two main reasons: first, the runtimes, while asymptotically quasilinear, include very large polylogarithmic factors. For realistic values of the security parameter, these polylogarithmic terms exceed the rather small (but asymptotically worse) quasilinear overhead obtained in [BGV12]. The second reason is that their bootstrapping operation is algorithmically very complex and difficult to implement (see the next paragraphs for details). Indeed, while there are now a few working implementations of bootstrapping (e.g., [GH11b, CCK+13]) that follow the templates from [Gen09b, Gen09a, vDGHV10, BGV12], we are not aware of any attempt to implement any method having subquadratic runtime.

[1] This stands in contrast with *leveled* FHE schemes, which can homomorphically evaluate a function of any *a priori* bounded depth, but using keys and ciphertexts whose sizes depend on the bound. Leveled FHE can be constructed without resorting to bootstrapping [BGV12].

Is quasilinear efficient? The complexity and large practical overhead of the constructions in [GHS12b, GHS12a] arise from two kinds of operations. First, the main technique from [GHS12b] is a way of homomorphically evaluating any sufficiently shallow and wide arithmetic circuit on a "packed" ciphertext that encrypts a high-dimensional vector of plaintexts in multiple "slots." It works by first using ring automorphisms and key-switching operations [BV11a, BGV12] to obtain a small, fixed set of "primitive" homomorphic permutations on the slots. It then composes those permutations (along with other homomorphic operations) in a log-depth permutation network, to obtain any permutation. Finally, it homomorphically evaluates the desired circuit by combining appropriate permutations with relatively simple homomorphic slot-selection and ring operations.

In the context of bootstrapping, one of the key observations from [GHS12a] is that a main step of the decryption procedure can be evaluated using the above technique. Specifically, they need an operation that moves the coefficients of an encrypted plaintext polynomial, reduced modulo a cyclotomic polynomial $\Phi_m(X)$, into the slots of a packed ciphertext (and back again). Once the coefficients are in the slots, they can be rounded in a batched (SIMD) fashion, and then mapped back to coefficients of the plaintext. The operations that move the coefficients into slots and vice-versa can be expressed as $O(\log \lambda)$-depth arithmetic circuits of size $O(\lambda \log \lambda)$, roughly akin to the classic FFT butterfly network. Hence they can be evaluated homomorphically with polylogarithmic overhead, using [GHS12b]. However, as the authors of [GHS12a] point out, the decryption circuit is quite large and complex – especially the part that moves the slots back to the coefficients, because it involves reduction modulo $\Phi_m(X)$ for an m having several prime divisors. This modular reduction is the most expensive part of the decryption circuit, and avoiding it is one of the main open problems given in [GHS12a]. However, even a very efficient decryption circuit would still incur the large polylogarithmic overhead factors from the techniques of [GHS12b].

1.1 Our Contributions

We give a new bootstrapping algorithm that runs in *quasilinear* $\tilde{O}(\lambda)$ time per ciphertext with *small* polylogarithmic factors, and is algorithmically much simpler than previous methods. It is easy to implement, and we believe that it will be substantially more efficient in practice than all prior methods. We provide a unified bootstrapping procedure that works for both "non-packed" ciphertexts (which encrypt integers modulo some p, e.g., bits) and "packed" ciphertexts (which encrypt elements of a high-dimensional ring), and also interpolates between the two cases to handle an intermediate concept we call "semi-packed" ciphertexts.

Our procedure for non-packed ciphertexts is especially simple and efficient. In particular, it can work very naturally using only cyclotomic rings having power-of-two index, i.e., rings of the form $\mathbb{Z}[X]/(1 + X^{2^k})$, which admit very fast implementations. This improves upon the method of [BGV12], which achieves quasilinear *amortized* runtime when bootstrapping $\tilde{\Omega}(\lambda)$ non-packed ciphertexts at once. Also, while that method can also use power-of-two cyclotomics, it can only do so by emulating $\mathbb{Z}_2$ (bit) arithmetic within $\mathbb{Z}_p$ for some moderately large prime p, which translates additions in $\mathbb{Z}_2$ into much more costly multiplications in $\mathbb{Z}_p$. By contrast, our method works "natively" with any plaintext modulus.

For packed ciphertexts, our procedure draws upon high-level ideas from [GHS12b, GHS12a], but our approach is conceptually and technically very different. Most importantly, it completely avoids the two main inefficiencies from those works: first, unlike [GHS12b], we do not use permutation networks or any explicit permutations of the plaintext slots, nor do we rely on a general-purpose compiler for homomorphically evaluating arithmetic circuits. Instead, we give direct, practically efficient procedures for homomorphically mapping the coefficients of an encrypted plaintext element into slots and vice-versa. In particular, our procedure does not incur the large cost or algorithmic complexity of homomorphically reducing modulo $\Phi_m(X)$, which was the main bottleneck in the decryption circuit of [GHS12a].

At a higher level, our bootstrapping method has two other attractive and novel features: first, it is entirely "algebraic," by which we mean that the full procedure (including generation of all auxiliary data it uses) can be described as a short sequence of elementary operations from the "native instruction set" of the SHE scheme. By contrast, all previous methods at some point invoke rather generic arithmetic circuits, e.g., for modular addition of values represented as bit strings, or reduction modulo a cyclotomic polynomial $\Phi_m(X)$. Of course, arithmetic circuits can be evaluated using the SHE scheme's native operations, but we believe that the distinction between "algebraic" and "non-algebraic" is an important qualitative one, and it certainly affects the simplicity and concrete efficiency of the bootstrapping procedure.

The second nice feature of our method is that it completely decouples the algebraic structure of the SHE plaintext ring from that which is needed by the bootstrapping procedure. In previous methods that use amortization (or "batching") for efficiency (e.g., [SV11, BGV12, GHS12a]), the ring and plaintext modulus of the SHE scheme must be chosen so as to provide many plaintext slots. However, this structure may not always be a natural match for the SHE application's efficiency or functionality requirements. For example, the lattice-based pseudorandom function of [BPR12] works very well with a ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ where both q and n are powers of two, but for such parameters R_q has only *one* slot. Our method can bootstrap even for this kind of plaintext ring (and many others), while still using batching to achieve quasilinear runtime.

1.2 Techniques

At the heart of our bootstrapping procedure are two novel homomorphic operations for SHE schemes over cyclotomic rings: for non-packed (or semi-packed) ciphertexts, we give an operation that *isolates the message-carrying coefficient(s)* of a high-dimensional ring element; and for (semi-)packed ciphertexts, we give an operation that *maps coefficients to slots* and vice-versa.

Isolating coefficients. Our first homomorphic operation is most easily explained in the context of non-packed ciphertexts, which encrypt single elements of the quotient ring $\mathbb{Z}_p$ for some small modulus p, using ciphertexts over some cyclotomic quotient ring $R_q = R/qR$ of moderately large degree $d = \deg(R/\mathbb{Z}) = \tilde{O}(\lambda)$. We first observe that a ciphertext to be bootstrapped can be reinterpreted as an encryption of an R_q-element, one of whose $\mathbb{Z}_q$-coefficients (with respect to an appropriate basis of the ring) "noisily" encodes the message, and whose other coefficients are just meaningless noise terms. We give an simple and efficient homomorphic operation that preserves the meaningful coefficient, and maps all the others to zero. Having isolated the message-encoding coefficient, we can then homomorphically apply an efficient integer "rounding" function (see [GHS12a] and Appendix B) to recover the message from its noisy encoding, which completes the bootstrapping procedure. (Note that it is necessary to remove the meaningless noise coefficients first, otherwise they would interfere with the correct operation of the rounding function.)

Our coefficient-isolating procedure works essentially by applying the *trace function* $\mathrm{Tr}_{R/\mathbb{Z}}\colon R \to \mathbb{Z}$ to the plaintext. The trace is the "canonical" $\mathbb{Z}$-linear function from R to $\mathbb{Z}$, and it turns out that for the appropriate choice of $\mathbb{Z}$-basis of R used in decryption, the trace simply outputs (up to some scaling factor) the message-carrying coefficient we wish to isolate. One simple and very efficient way of applying the trace homomorphically is to use the "ring-switching" technique of [GHPS12], but unfortunately, this requires the ring-LWE problem [LPR10] to be hard over the target ring $\mathbb{Z}$, which is clearly not the case. Another way follows from the fact that $\mathrm{Tr}_{R/\mathbb{Z}}$ equals the sum of all d automorphisms of R; therefore, it can be computed by homomorphically applying each automorphism and summing the results. Unfortunately, this method takes at least *quadratic* $\Omega(\lambda^2)$ time, because applying each automorphism homomorphically takes $\Omega(\lambda)$ time, and there are $d = \Omega(\lambda)$ automorphisms.

So, instead of inefficiently computing the trace by summing all the automorphisms at once, we consider a *tower* of cyclotomic rings $\mathbb{Z} = R^{(0)} \subseteq R^{(1)} \subseteq \cdots \subseteq R^{(r)} = R$, usually written as $R^{(r)}/\cdots/R^{(1)}/R^{(0)}$. Then $\mathrm{Tr}_{R/\mathbb{Z}}$ is the composition of the individual trace functions $\mathrm{Tr}_{R^{(i)}/R^{(i-1)}} \colon R^{(i)} \to R^{(i-1)}$, and these traces are equal to the sums of all automorphisms of $R^{(i)}$ that fix $R^{(i-1)}$ pointwise, of which there are exactly $d_i = \deg(R^{(i)}/R^{(i-1)}) = \deg(R^{(i)}/\mathbb{Z})/\deg(R^{(i-1)}/\mathbb{Z})$. We can therefore compute each $\mathrm{Tr}_{R^{(i)}/R^{(i-1)}}$ in time linear in λ and in d_i; moreover, the number of trace functions to apply is at most logarithmic in $d = \deg(R/\mathbb{Z}) = \tilde{O}(\lambda)$, because each one reduces the degree by a factor of at least two. Therefore, by ensuring that the degrees of $R^{(r)}, R^{(r-1)}, \ldots, R^{(0)}$ decrease gradually enough, we can homomorphically apply the full $\mathrm{Tr}_{R/\mathbb{Z}}$ in quasilinear time. For example, a particularly convenient choice is to let $R^{(i)}$ be the 2^{i+1}st cyclotomic ring $\mathbb{Z}[X]/(1 + X^{2^i})$ of degree 2^i, so that every $d_i = 2$, and there are exactly $\log_2(d) = O(\log \lambda)$ trace functions to apply.

More generally, when bootstrapping a *semi-packed* ciphertext we start with a plaintext value in R_q that noisily encodes a message in S_p, for some subring $S \subseteq R$. (The case $S = \mathbb{Z}$ corresponds to a non-packed ciphertext.) We show that applying the trace function $\mathrm{Tr}_{R/S}$ to the R_q-plaintext yields a new plaintext *in* S_q that noisily encodes the message, thus isolating the meaningful part of the noisy encoding and vanishing the rest. We then homomorphically apply a rounding function to recover the S_p message from its noisy S_q encoding, which uses the technique described next.

Mapping coefficients to slots. Our second technique, and main technical innovation, is in bootstrapping (semi-)packed ciphertexts. We enhance the recent "ring-switching" procedure of [GHPS12], and use it to efficiently move "noisy" plaintext coefficients (with respect to an appropriate decryption basis) into slots for batch-rounding, and finally move the rounded slot values back to coefficients. We note that all previous methods for loading plaintext data into slots used the *same* ring for the source and destination, and so required the plaintext to come from a ring designed to have many slots. In this work, we use ring-switching to go from the SHE plaintext ring to a *different* ring having many slots, which is used only temporarily for batch-rounding. This is what allows the SHE plaintext ring to be decoupled from the rings used in bootstrapping, as mentioned above.

To summarize our technique, we first recall the ring-switching procedure of [GHPS12]. It was originally devised to provide moderate efficiency gains for SHE/FHE schemes, by allowing them to switch ciphertexts from high-degree cyclotomic rings to *subrings* of smaller degree (once enough homomorphic operations have been performed to make this secure). We generalize the procedure, showing how to switch between two rings where neither ring need be a subring of the other. The procedure has a very simple implementation, and as long as the two rings have a large *common subring*, it is also very efficient (i.e., quasilinear in the dimension). Moreover, it supports, as a side effect, the homomorphic evaluation of any function that is *linear over the common subring*. However, the larger the common subring is, the more restrictive this condition on the function becomes.

We show how our enhanced ring-switching can move the plaintext coefficients into the slots of the target ring (and back), which can be seen as just evaluating a certain $\mathbb{Z}$-linear function. Here we are faced with the main technical challenge: for efficiency, the common subring of the source and destination rings must be large, but then the supported class of linear functions is very restrictive, and certainly does not include the $\mathbb{Z}$-linear one we want to evaluate. We solve this problem by switching through a short sequence of "hybrid" rings, where adjacent rings have a large common subring, but the initial and final rings have only the integers $\mathbb{Z}$ in common. Moreover, we show that for an appropriately chosen sequence of hybrid rings, the $\mathbb{Z}$-linear function we want to evaluate *is* realizable by a sequence of allowed linear functions between adjacent hybrid rings. Very critically, this decomposition requires the SHE scheme to use a *highly structured*

basis of the ring for decryption. The usual representation of a cyclotomic ring as $\mathbb{Z}[X]/\Phi_m(X)$ typically does not correspond to such a basis, so we instead rely on the *tensorial decomposition* of the ring and its corresponding bases, as recently explored in [LPR13]. At heart, this is what allows us to avoid the expensive homomorphic reduction modulo $\Phi_m(X)$, which is one of the main bottlenecks in previous work [GHS12a].[2]

Stepping back a bit, the technique of switching through hybrid rings and bases is reminiscent of standard "sparse decompositions" for linear transformations like the FFT, in that both decompose a complicated high-dimensional transform into a short sequence of simpler, structured transforms. (Here, the simple transforms are computed merely as a side-effect of passing through the hybrid rings.) Because of these similarities, we believe that the enhanced ring-switching procedure will be applicable in other domain-specific applications of homomorphic encryption, e.g., signal-processing transforms or statistical analysis.

Organization. Section 2.1 recalls the extensive algebraic background required for our constructions, and Section 2.2 recalls a standard ring-based SHE scheme and some of its natural homomorphic operations. Section 3 defines the general bootstrapping procedure. Sections 4 and 5 respectively fill in the details of the two novel homomorphic operations used in the bootstrapping procedure. Appendix A documents a folklore transformation between two essentially equivalent ways of encoding messages in SHE schemes. Appendix B describes an integer rounding procedure that simplifies the one given in [GHS12a], and Appendix C gives some concrete choices of rings that our method can use in practice.

Acknowledgments. We thank Oded Regev for helpful discussions during the early stages of this research, and the anonymous CRYPTO'13 reviewers for their thoughtful comments.

2 Preliminaries

For a positive integer k, we let $[k] = \{0, \ldots, k-1\}$. For an integer modulus q, we let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the quotient ring of integers modulo q. For integers q, q', we define the integer "rounding" function $\lfloor \cdot \rceil_{q'} \colon \mathbb{Z}_q \to \mathbb{Z}_{q'}$ as $\lfloor x \rceil_{q'} = \lfloor (q'/q) \cdot x \rceil \bmod q'$.

2.1 Algebraic Background

Throughout this work, by "ring" we mean a commutative ring with identity. For two rings $R \subseteq R'$, an R-basis of R' is a set $B \subset R'$ such that every $r \in R'$ can be written uniquely as an R-linear combination of elements of B. For two rings R, S with a common subring E, an E-linear function $L \colon R \to S$ is one for which $L(r + r') = L(r) + L(r')$ for all $r, r' \in R$, and $L(e \cdot r) = e \cdot L(r)$ for all $e \in E, r \in R$. It is immediate that such a function is defined uniquely by its values on any E-basis of R.

2.1.1 Cyclotomic Rings

For a positive integer m called the *index*, let $\mathcal{O}_m = \mathbb{Z}[\zeta_m]$ denote the mth *cyclotomic ring*, where ζ_m is an abstract element of order m over $\mathbb{Q}$. (In particular, we do not view ζ_m as any particular complex root of unity.) The minimal polynomial of ζ_m over $\mathbb{Q}$ is the mth *cyclotomic polynomial* $\Phi_m(X) = \prod_{i \in \mathbb{Z}_m^*} (X - \omega_m^i) \in \mathbb{Z}[X]$, where $\omega_m = \exp(2\pi\sqrt{-1}/m) \in \mathbb{C}$ is the principal mth complex root of unity, and the roots $\omega_m^i \in \mathbb{C}$ range

[2] The use of more structured representations of cyclotomic rings in [LPR13] was initially motivated by the desire for simpler and more efficient algorithms for cryptographic operations. Interestingly, these representations yield moderate efficiency improvements for computations "in the clear," but dramatic benefits for their homomorphic counterparts!

over all the *primitive* complex mth roots of unity. Therefore, $\mathcal{O}_m$ is a ring extension of degree $n = \varphi(m)$ over $\mathbb{Z}$. (In particular, $\mathcal{O}_1 = \mathcal{O}_2 = \mathbb{Z}$.) Clearly, $\mathcal{O}_m$ is isomorphic to the polynomial ring $\mathbb{Z}[X]/\Phi_m(X)$ by identifying ζ_m with X, and has the "power basis" $\{1, \zeta_m, \ldots, \zeta_m^{n-1}\}$ as a $\mathbb{Z}$-basis. However, for non-prime-power m the power basis can be somewhat cumbersome and inefficient to work with. In Section 2.1.4 we consider other, more structured bases that are essential to our techniques.

If $m|m'$, we can view the mth cyclotomic ring $\mathcal{O}_m$ as a subring of $\mathcal{O}_{m'} = \mathbb{Z}[\zeta_{m'}]$, via the ring embedding (i.e., injective ring homomorphism) that maps ζ_m to $\zeta_{m'}^{m'/m}$. The ring extension $\mathcal{O}_{m'}/\mathcal{O}_m$ has degree $d = \varphi(m')/\varphi(m)$, and also d automorphisms τ_i (i.e., automorphisms of $\mathcal{O}_{m'}$ that fix $\mathcal{O}_m$ pointwise), which are defined by $\tau_i(\zeta_{m'}) = \zeta_{m'}^i$ for each $i \in \mathbb{Z}_{m'}^*$ such that $i = 1 \pmod m$. The *trace* function $\mathrm{Tr} = \mathrm{Tr}_{\mathcal{O}_{m'}/\mathcal{O}_m} : \mathcal{O}_{m'} \to \mathcal{O}_m$ can be defined as the sum of these automorphisms:

$$\mathrm{Tr}_{\mathcal{O}_{m'}/\mathcal{O}_m}(a) = \sum_i \tau_i(a) \in \mathcal{O}_m.$$

Notice that Tr is $\mathcal{O}_m$-linear by definition. If $\mathcal{O}_{m''}/\mathcal{O}_{m'}/\mathcal{O}_m$ is a tower of ring extensions, then the trace satisfies the composition property $\mathrm{Tr}_{\mathcal{O}_{m''}/\mathcal{O}_m} = \mathrm{Tr}_{\mathcal{O}_{m'}/\mathcal{O}_m} \circ \mathrm{Tr}_{\mathcal{O}_{m''}/\mathcal{O}_{m'}}$.

An important element in the mth cyclotomic ring is

$$g := \prod_{\text{odd prime } p|m} (1 - \zeta_p) \in \mathcal{O}_m. \tag{2.1}$$

Also define $\hat{m} = m/2$ if m is even, otherwise $\hat{m} = m$, for any cyclotomic index m. It is known that $g|\hat{m}$ (see, e.g., [LPR13, Section 2.5.4]). The following lemma shows how the elements g in different cyclotomic rings, and the ideals they generate, are related by the trace function.

Lemma 2.1. *Let $m|m'$ be positive integers and let $g \in R = \mathcal{O}_m, g' \in R' = \mathcal{O}_{m'}$ and $\hat{m}, \hat{m}'$ be as defined above. Then $\mathrm{Tr}_{R'/R}(g'R') = (\hat{m}'/\hat{m}) \cdot gR$, and in particular, $\mathrm{Tr}_{R'/R}(g') = (\hat{m}'/\hat{m}) \cdot g$.*

Later on we use the *scaled* trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$, which by the above lemma maps the ideal $g'R$ to gR, and g' to g.

Proof. Let $\mathrm{Tr} = \mathrm{Tr}_{R'/R}$. To prove the first claim, we briefly recall certain properties of $R^\vee$, the fractional ideal "dual" to R; see [LPR13, Section 2.5.4] for further details. First, $R^\vee = (g/\hat{m})R$, and similarly $(R')^\vee = (g/\hat{m}')R'$. It also follows directly from the definition of the dual ideal that $\mathrm{Tr}((R')^\vee) = R^\vee$; see for example [GHPS12, Equation 2.2]. Therefore, $\mathrm{Tr}(g'R') = (\hat{m}'/\hat{m}) \cdot gR$.

For the second claim, we first show the effect of the trace on g' when $m' = m \cdot p$ for some prime p. If p divides m, then $\hat{m}'/\hat{m} = m'/m = p$, the degree of R'/R is $\varphi(m')/\varphi(m) = p$, and $g' = g \in R$, so $\mathrm{Tr}(g') = \mathrm{Tr}(g) = p \cdot g$. Now suppose p does not divide m. If $p = 2$, then m is even and m' is odd, so $\hat{m}'/\hat{m} = (m'/2)/m = 1$, the degree of R'/R is 1, and $g' = g \in R$, so $\mathrm{Tr}(g') = g$. Otherwise p is odd, so $\hat{m}'/\hat{m} = m'/m = p$ and $g' = (1 - \zeta_p)g$. Therefore $\mathrm{Tr}(g') = \mathrm{Tr}(1 - \zeta_p) \cdot g = p \cdot g$, where the final equality follows from $\mathrm{Tr}(1) = p - 1$ and $\mathrm{Tr}(\zeta_p) = \zeta_p^1 + \zeta_p^2 + \cdots \zeta_p^{p-1} = -1$.

The general case follows from the composition property of the trace, by iteratively applying the above case to any cyclotomic tower $R^{(r)}/R^{(r-1)}/\cdots/R^{(0)}$, where $R^{(r)} = R'$ and $R^{(0)} = R$, and the ratio of the indices of $R^{(i)}, R^{(i-1)}$ is prime for every $i = 1, \ldots, r$. □

2.1.2 Tensorial Decomposition of Cyclotomics

An important fact from algebraic number theory, used centrally in this work (and in [LPR13]), is the *tensorial decomposition* of cyclotomic rings (and their bases) in terms of subrings. Let $\mathcal{O}_{m_1}, \mathcal{O}_{m_2}$ be cyclotomic rings. Then their largest common subring is $\mathcal{O}_{m_1} \cap \mathcal{O}_{m_2} = \mathcal{O}_g$ where $g = \gcd(m_1, m_2)$, and their smallest common extension ring, called the *compositum*, is $\mathcal{O}_{m_1} + \mathcal{O}_{m_2} = \mathcal{O}_l$ where $l = \mathrm{lcm}(m_1, m_2)$. When considered as extensions of $\mathcal{O}_g$, the ring $\mathcal{O}_l$ is isomorphic to the *ring tensor product* of $\mathcal{O}_{m_1}$ and $\mathcal{O}_{m_2}$, written as (sometimes suppressing $\mathcal{O}_g$ when it is clear from context)

$$\mathcal{O}_l/\mathcal{O}_g \cong (\mathcal{O}_{m_1}/\mathcal{O}_g) \otimes (\mathcal{O}_{m_2}/\mathcal{O}_g).$$

On the right, the ring tensor product is defined as the set of all $\mathcal{O}_g$-linear combinations of *pure tensors* $a_1 \otimes a_2$, with ring operations defined by $\mathcal{O}_g$-bilinearity:

$$\begin{aligned}(a_1 \otimes a_2) + (b_1 \otimes a_2) &= (a_1 + b_1) \otimes a_2,\\ (a_1 \otimes a_2) + (a_1 \otimes b_2) &= a_1 \otimes (a_2 + b_2),\\ c(a_1 \otimes a_2) &= (ca_1) \otimes a_2 = a_1 \otimes (ca_2)\end{aligned}$$

for any $c \in \mathcal{O}_g$, and the mixed-product property $(a_1 \otimes a_2) \cdot (b_1 \otimes b_2) = (a_1 b_1) \otimes (a_2 b_2)$. The isomorphism with $\mathcal{O}_l/\mathcal{O}_g$ then simply identifies $a_1 \otimes a_2$ with $a_1 \cdot a_2 \in \mathcal{O}_l$. Note that any $a_1 \in \mathcal{O}_{m_1}$ corresponds to the pure tensor $a_1 \otimes 1$, and similarly for any $a_2 \in \mathcal{O}_{m_2}$.

The following simple lemma will be central to our techniques.

Lemma 2.2. *Let m_1, m_2 be positive integers and $g = \gcd(m_1, m_2)$, $l = \mathrm{lcm}(m_1, m_2)$. Then for any $\mathcal{O}_g$-linear function $\bar{L}\colon \mathcal{O}_{m_1} \to \mathcal{O}_{m_2}$, there is an (efficiently computable) $\mathcal{O}_{m_2}$-linear function $L\colon \mathcal{O}_l \to \mathcal{O}_{m_2}$ that coincides with $\bar{L}$ on the subring $\mathcal{O}_{m_1} \subseteq \mathcal{O}_l$.*

Proof. Write $\mathcal{O}_l \cong \mathcal{O}_{m_1} \otimes \mathcal{O}_{m_2}$, where the common base ring $\mathcal{O}_g$ is implicit. Let $L\colon (\mathcal{O}_{m_1} \otimes \mathcal{O}_{m_2}) \to \mathcal{O}_{m_2}$ be the $\mathcal{O}_g$-linear function uniquely defined by $L(a_1 \otimes a_2) = \bar{L}(a_1) \cdot a_2 \in \mathcal{O}_{m_2}$ for all pure tensors $a_1 \otimes a_2$. Then because $(a_1 \otimes a_2) \cdot b_2 = a_1 \otimes (a_2 b_2)$ for any $b_2 \in \mathcal{O}_{m_2}$ by the mixed-product property, L is also $\mathcal{O}_{m_2}$-linear. Finally, for any $a_1 \in \mathcal{O}_{m_1}$ we have $L(a_1 \otimes 1) = \bar{L}(a_1)$ by construction. □

2.1.3 Ideal Factorization and Plaintext Slots

Here we recall the unique factorization of prime integers into prime ideals in cyclotomic rings, and, following [SV11], how the Chinese remainder theorem can yield several plaintext "slots" that embed $\mathbb{Z}_q$ as a subring, even for composite q. Similar facts for composite moduli are presented in [GHS12a], but in terms of p-adic approximations and Hensel lifting. Here we give an ideal-theoretic interpretation using the Chinese remainder theorem, which we believe is more elementary, and is a direct extension of the case of prime moduli.

Let $p \in \mathbb{Z}$ be a prime integer. In the mth cyclotomic ring $R = \mathcal{O}_m = \mathbb{Z}[\zeta_m]$ (which has degree $n = \varphi(m)$ over $\mathbb{Z}$), the ideal pR factors into prime ideals as follows. First write $m = \bar{m} \cdot p^k$ where $p \nmid \bar{m}$. Let $e = \varphi(p^k)$, and let d be the multiplicative order of p modulo in $\mathbb{Z}^*_{\bar{m}}$, and note that d divides $\varphi(\bar{m}) = n/e$. The ideal pR then factors into the product of eth powers of $\varphi(\bar{m})/d = n/(de)$ distinct prime ideals $\mathfrak{p}_i$, i.e.,

$$pR = \prod \mathfrak{p}_i^e.$$

Each prime ideal $\mathfrak{p}_i$ has norm $|R/\mathfrak{p}_i| = p^d$, so each quotient ring $R/\mathfrak{p}_i$ is isomorphic to the finite field $\mathbb{F}_{p^d}$. In particular, it embeds $\mathbb{Z}_p$ as a subfield. (Although we will not need this, the prime ideals are concretely given by $\mathfrak{p}_i = pR + F_i(\zeta_m)R$, where $\Phi_{\bar{m}}(X) = \prod_i F_i(X) \pmod p$ is the mod-p factorization of the $\bar{m}$th cyclotomic polynomial into $\varphi(\bar{m})/d$ distinct irreducible polynomials of degree d.)

We now see how to obtain quotient rings of R that embed the ring $\mathbb{Z}_q$, where $q = p^r$ for some integer $r \geq 1$. (The case of arbitrary integer modulus q follows immediately from the Chinese remainder theorem.) Here we have the factorization $qR = \prod_i \mathfrak{p}_i^{re}$, and it turns out that each quotient ring $R/\mathfrak{p}_i^{re}$ embeds $\mathbb{Z}_q$ as a subring. One easy way to see this is to notice that q is the smallest power of p in $\mathfrak{p}_i^{re}$, so the integers $\{0, \ldots, q-1\}$ representing $\mathbb{Z}_q$ are distinct modulo $\mathfrak{p}_i^{re}$.

By the Chinese Remainder Theorem (CRT), for $q = p^r$ the natural ring homomorphism from R_q to the product ring $\bigoplus_i (R/\mathfrak{p}_i^{re})$ is an isomorphism. When the natural plaintext space of a cryptosystem is R_q, we refer to the $\varphi(\bar{m})/d$ quotient rings $R/\mathfrak{p}_i^{re}$ as the plaintext "$\mathbb{Z}_q$-slots" (or just "slots"), and use them to store vectors of $\mathbb{Z}_q$-elements via the CRT isomorphism. With this encoding, ring operations in R_q induce "batch" (or "SIMD") component-wise operations on the corresponding vectors of $\mathbb{Z}_q$ elements. We note that the CRT isomorphism is easy to compute in both directions. In particular, to map from a vector of $\mathbb{Z}_q$-elements to R_q just requires knowing a fixed mod-q CRT set $C = \{c_i\} \subset R$ for which $c_i = 1 \pmod{\mathfrak{p}_i^{re}}$ and $c_i = 0 \pmod{\mathfrak{p}_j^{re}}$ for all $j \neq i$. Such a set can be precomputed using, e.g., a generalization of the extended Euclidean algorithm.

Splitting in cyclotomic extension rings. Now consider a cyclotomic extension R'/R where $R' = \mathcal{O}_{m'} = \mathbb{Z}[\zeta_{m'}]$ for some m' divisible by m. Then for each prime ideal $\mathfrak{p}_i \subset R$ dividing pR, the ideal $\mathfrak{p}_i R'$ factors into equal powers of the same number of prime ideals $\mathfrak{p}'_{i,i'} \subset R'$, where all the $\mathfrak{p}'_{i,i'}$ are distinct. The ideal $\mathfrak{p}'_{i,i'}$ is said to "lie over" $\mathfrak{p}_i$ (and $\mathfrak{p}_i$ in turns lies over p). Since $\mathfrak{p}'_{i,i'}$ are also the prime ideals appearing in the factorization pR', we can determine their number and multiplicity exactly as above. Letting $\bar{m}'$, e' and d' be defined as above for R', we known that $pR' = \prod_{i,i'} (\mathfrak{p}'_{i,i'})^{e'}$, where there are a total of $\varphi(\bar{m}')/d'$ distinct prime ideals $\mathfrak{p}'_{i,i'}$. Therefore, each $\mathfrak{p}_i$ splits into exactly $(\varphi(\bar{m}') \cdot d)/(\varphi(\bar{m}) \cdot d')$ ideals each; this number is sometimes called the "relative splitting number" of p in R'/R.

2.1.4 Product Bases

Our bootstrapping technique relies crucially on certain highly structured bases and CRT sets, which we call "product bases (sets)," that arise from towers of cyclotomic rings. Let $\mathcal{O}_{m''}/\mathcal{O}_{m'}/\mathcal{O}_m$ be such a tower, let $B'' = \{b''_{j''}\} \subset \mathcal{O}_{m''}$ be any $\mathcal{O}_{m'}$-basis of $\mathcal{O}_{m''}$, and let $B' = \{b'_{j'}\} \subset \mathcal{O}_{m'}$ be any $\mathcal{O}_m$-basis of $\mathcal{O}_{m'}$. Then it follows immediately that the product set $B'' \cdot B' := \{b''_{j''} \cdot b'_{j'}\} \subset \mathcal{O}_{m''}$ is an $\mathcal{O}_m$-basis of $\mathcal{O}_{m''}$.[3] Of course, for a tower of several cyclotomic extensions and relative bases, we can obtain product bases that factor with a corresponding degree of granularity.

Factorization of the powerful and decoding bases. An important structured $\mathbb{Z}$-basis of $\mathcal{O}_m$, called the "powerful" basis in [LPR13], was defined in that work as the product of all the power $\mathbb{Z}$-bases $\{\zeta^0, \zeta^1, \ldots, \zeta^{\varphi(p^e)-1}\}$ of $\mathcal{O}_{p^e}$ (where $\zeta = \zeta_{p^e}$), taken over all the maximal prime-power divisors p^e of m. In turn, it is straightforward to verify that the power $\mathbb{Z}$-basis of $\mathcal{O}_{p^e}$ can be obtained from the tower $\mathcal{O}_{p^e}/\mathcal{O}_{p^{e-1}}/\cdots/\mathbb{Z}$, as the product of all the power $\mathcal{O}_{p^{i-1}}$-bases $\{\zeta_{p^i}^0, \ldots, \zeta_{p^i}^{d_i-1}\}$ of $\mathcal{O}_{p^i}$ for $i = 1, \ldots, e$, where $d_i = \varphi(p^i)/\varphi(p^{i-1}) \in \{p-1, p\}$ is the degree of $\mathcal{O}_{p^i}/\mathcal{O}_{p^{i-1}}$. Therefore, the powerful basis has a

[3] Formally, this basis is a *Kronecker* product of the bases B'' and B', which is typically written using the $\otimes$ operator. We instead use $\cdot$ to avoid confusion with pure tensors in a ring tensor product, which the elements of $B'' \cdot B'$ may not necessarily be.

"finest possible" product structure. (This is not the case for other commonly used bases of $\mathcal{O}_m$, such as the power $\mathbb{Z}$-basis, unless m is a prime power.)

Similarly, [LPR13] defines the "decoding" $\mathbb{Z}$-basis D of a certain fractional ideal $\mathcal{O}_m^\vee = (g/\hat{m})\mathcal{O}_m$, which is the "dual ideal" of $\mathcal{O}_m$, to be the dual basis of the conjugate powerful basis. Unlike the powerful basis, the decoding basis has optimal noise tolerance (see [LPR13, Section 6.2]) and is therefore a best choice to use in decryption, when using the dual ideal $\mathcal{O}_m^\vee$ appropriately in a cryptosystem. For simplicity, our formulation of the cryptosystem (see Section 2.2) avoids using $\mathcal{O}_m^\vee$ by "scaling up" to $(\hat{m}/g)\mathcal{O}_m^\vee = \mathcal{O}_m$, and so we are interested in factorizations of the scaled-up $\mathbb{Z}$-basis $(\hat{m}/g)D$ of $\mathcal{O}_m$. As shown in [LPR13, Lemma 6.3], this basis is very closely related to the powerful basis, and has a nearly identical product structure arising from the towers $\mathcal{O}_{p^e}/\mathcal{O}_{p^{e-1}}/\cdots/\mathbb{Z}$ for the maximal prime-power divisors p^e of m. The only difference is in the choice of the lowest-level $\mathbb{Z}$-bases of each $\mathcal{O}_p/\mathbb{Z}$, which are taken to be $\{\zeta_p^j + \zeta_p^{j+1} + \cdots + \zeta_p^{p-2}\}_{j\in\{0,\ldots,p-2\}}$ instead of the power basis. In summary, the preferred $\mathbb{Z}$-basis of $\mathcal{O}_m$ used for decryption also has a finest-possible product structure.

Factorization of CRT sets. Using the splitting behavior of primes and prime ideals, we can also define CRT sets having a finest-possible product structure. First consider any cyclotomic extension $\mathcal{O}_{m'}/\mathcal{O}_m$, and suppose that prime integer p splits in $\mathcal{O}_m$ into distinct prime ideals $\mathfrak{p}_i$. In turn, each $\mathfrak{p}_i$ splits in $\mathcal{O}_{m'}$ into the same number k of prime ideals $\mathfrak{p}'_{i,i'}$, which are all distinct. For simplicity, assume for now that p does not divide m or m', so none of the ideals occur with multiplicity.

A mod-p CRT set $C = \{c_i\}$ for $\mathcal{O}_m$ satisfies $c_i = 1 \pmod{\mathfrak{p}_i}$ and $c_i = 0 \pmod{\mathfrak{p}_j}$ for $j \neq i$; therefore, $c_i = 1 \pmod{\mathfrak{p}'_{i,i'}}$ and $c_i = 0 \pmod{\mathfrak{p}'_{j,i'}}$ for all i' and all $j \neq i$. We can choose a set $S = \{s_{i'}\} \subset \mathcal{O}_{m'}$ of size k such that $C' = S \cdot C$ is a mod-p CRT set for $\mathcal{O}_{m'}$, as follows: partition the ideals $\mathfrak{p}'_{i,i'}$ arbitrarily according to i', and define $s_{i'} \in \mathcal{O}_{m'}$ to be congruent to 1 modulo all those ideals $\mathfrak{p}'_{i,i'}$ in the i'th component of the partition, and 0 modulo all the other ideals $\mathfrak{p}'_{j,j'}$. Then it is immediate that each product $c_i \cdot s_{i'}$ is 1 modulo $\mathfrak{p}'_{i,i'}$, and 0 modulo all other $\mathfrak{p}'_{j,j'}$. Therefore, $C' = S \cdot C$ is a mod-p CRT set for $\mathcal{O}_{m'}$. The generalization of this process to the case where p factors into *powers* of the ideals, and to moduli $q = p^r$, is immediate.

For an arbitrary cyclotomic index m, consider any cyclotomic tower $\mathcal{O}_m/\cdots/\mathbb{Z}$. Then a mod-$q$ CRT set with corresponding product structure can be obtained by iteratively applying the above procedure at each level of the tower. A finest-possible product structure is obtained by using tower of maximal length (i.e., one in which the ratio of indices at adjacent levels is always prime).

2.2 Ring-Based Homomorphic Cryptosystem

Here we recall a somewhat-homomorphic encryption scheme whose security is based on the ring-LWE problem [LPR10] in arbitrary cyclotomic rings. For our purposes we focus mainly on its decryption function, though below we also recall its support for "ring switching" [GHPS12]. For further details on its security guarantees, various homomorphic properties, and efficient implementation, see [LPR10, BV11b, BGV12, GHS12c, GHPS12, LPR13].

Let $R = \mathcal{O}_m \subseteq R' = \mathcal{O}_{m'}$ be respectively the mth and m'th cyclotomic rings, where $m|m'$. The plaintext ring is the quotient ring R_p for some integer p; ciphertexts are made up of elements of R'_q for some integer q, which for simplicity we assume is divisible by p; and the secret key is some $s \in R'$. The case $m = 1$ corresponds to "non-packed" ciphertexts, which encrypt elements of $\mathbb{Z}_p$ (e.g., single bits), whereas $m = m'$ corresponds to "packed" ciphertexts, and $1 < m < m'$ corresponds to what we call "semi-packed" ciphertexts. Note that without loss of generality we can treat any ciphertext as packed, since R'_p embeds R_p.

But the smaller m is, the simpler and more practically efficient our bootstrapping procedure can be. Since our focus is on refreshing ciphertexts that have large noise rate, we can think of m' as being somewhat small (e.g., in the several hundreds) via ring-switching [GHPS12], and q also as being somewhat small (e.g., in the several thousands) via modulus-switching. Our main focus in this work is on a plaintext modulus p that is a power of two, though for generality we present all our techniques in terms of arbitrary p.

A ciphertext encrypting a message $\mu \in R_p$ under secret key $s' \in R'$ is some pair $c' = (c'_0, c'_1) \in R'_q \times R'_q$ satisfying the relation

$$c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \pmod{qR'} \tag{2.2}$$

for some error (or "noise") term $e' \in R'$ such that $e' \cdot g' \in g'R'$ is sufficiently "short," where $g' \in R'$ is as defined in Equation (2.1).[4] Informally, the "noise rate" of the ciphertext is the ratio of the "size" of e' (or more precisely, the magnitude of its coefficients in a suitable basis) to q/p.

We note that Equation (2.2) corresponds to what is sometimes called the "most significant bit" (msb) message encoding, whereas somewhat-homomorphic schemes are often defined using "least significant bit" (lsb) encoding, in which p and q are coprime and $c'_0 + c'_1 s' = e' \pmod{qR'}$ for some error term $e' \in \mu + pR'$. For our purposes the msb encoding is more natural, and in any case the two encodings are essentially equivalent: when p and q are coprime, we can trivially switch between the two encodings simply by multiplying by p or p^{-1} modulo q (see Appendix A). When p divides q, we can use homomorphic operations for the msb encoding due to Brakerski [Bra12]; alternatively, we can switch to and from a different modulus q' that is coprime with p, allowing us to switch between lsb and msb encodings as just described. In practice, it may be preferable to use homomorphic operations for the lsb encoding, because they admit optimizations (e.g., the "double-CRT representation" [GHS12c]) that may not be possible for the msb operations (at least when p divides q).

2.2.1 Decryption

At a high level, the decryption algorithm works in two steps: the "linear" step simply computes $v' = c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \in R'_q$, and the "non-linear" step outputs $\lfloor v' \rceil_p \in R_p$ using a certain "ring rounding function" $\lfloor \cdot \rceil_p \colon R'_q \to R_p$. As long as the error term e' is within the tolerance of the rounding function, the output will be $\mu \in R_p$. This is all entirely analogous to decryption in LWE-based systems, but here the rounding is n-dimensional, rather than just from $\mathbb{Z}_q$ to $\mathbb{Z}_p$.

Concretely, the ring rounding function $\lfloor \cdot \rceil_p \colon R'_q \to R_p$ is defined in terms of the integer rounding function $\lfloor \cdot \rceil_p \colon \mathbb{Z}_q \to \mathbb{Z}_p$ and a certain "decryption" $\mathbb{Z}$-basis $B' = \{b_j\}$ of R', as follows.[5] Represent the input $v' \in R'_q$ in the decryption basis as $v' = \sum_j v'_j \cdot b'_j$ for some coefficients $v'_j \in \mathbb{Z}_q$, then independently round the coefficients, yielding an element $\sum \lfloor v'_j \rceil_p \cdot b'_j \in R'_p$ that corresponds to the message $\mu \in R_p$ (under the standard embedding of R_p into R'_p).

[4] Quantitatively, "short" is defined with respect to the *canonical embedding* of R', whose precise definition is not needed in this work. The above system is equivalent to the one from [LPR13] in which the message, error term, and ciphertext components are all taken over the "dual" fractional ideal $(R')^\vee = (g'/\hat{m}')R'$ in the m'th cyclotomic number field, and the error term has an essentially spherical distribution over $(R')^\vee$. In that system, decryption is best accomplished using a certain $\mathbb{Z}$-basis of $(R')^\vee$, called the *decoding basis*, which optimally decodes spherical errors. The above formulation is more convenient for our purposes, and simply corresponds with multiplying everything in the system of [LPR13] by an $\hat{m}'/g'$ factor. This makes $e' \cdot g' \in g'R' = \hat{m}'(R')^\vee)$ short and essentially spherical in our formulation. See [LPR10, LPR13] for further details.

[5] In our formulation, the basis B' is $(\hat{m}'/g')$ times the decoding basis of $(R')^\vee$. See Section 2.1.4 and Footnote 4.

2.2.2 Changing the Plaintext Modulus

We use two operations on ciphertexts that alter the plaintext modulus p and encrypted message $\mu \in R_p$. The first operation changes p to any multiple $p' = dp$, and produces an encryption of some $\mu' \in R'_{p'}$ such that $\mu' = \mu \pmod{pR'}$. To do this, it simply "lifts" the input ciphertext $c' = (c'_0, c'_1) \in (R'_q)^2$ to an arbitrary $c'' = (c''_0, c''_1) \in (R'_{q'})^2$ such that $c''_j = c'_j \pmod{qR'}$, where $q' = dq$. This works because

$$c''_0 + c''_1 \cdot s' \in c'_0 + c'_1 \cdot s' + qR' = \left(\frac{q}{p} \cdot \mu + e'\right) + qR' = \frac{q'}{p'}(\mu + pR') + e' \pmod{q'R'}.$$

Notice that this leaves the noise rate unchanged, because the noise term is still e', and $q'/p' = q/p$.

The second operation applies to an encryption of a message $\mu \in R_p$ that is known to be divisible by some divisor d of p, and produces an encryption of $\mu/d \in R_{p/d}$. The operation actually leaves the ciphertext c' unchanged; it just declares the associated plaintext modulus to be p/d (which affects how decryption is performed). This works because

$$c'_0 + c'_1 \cdot s' = \frac{q}{p}\mu + e' = \frac{q}{p/d} \cdot (\mu/d) + e' \pmod{qR'}.$$

Notice that the noise rate of the ciphertext has been divided by d, because the noise term is still e' but $q/p' = d(q/p)$.

2.2.3 Ring Switching

We rely heavily on the cryptosystem's support for switching ciphertexts to a cyclotomic subring S' of R', which as a side-effect homomorphically evaluates any desired S'-linear function on the plaintext. Notice that the linear function L is applied to the plaintext as embedded in R'_p; this obviously applies the induced function on the true plaintext space R_p.

Proposition 2.3 ([GHPS12], full version). *Let $S' \subseteq R'$ be cyclotomic rings. Then the above-described cryptosystem supports the following homomorphic operation: given any S'-linear function $L\colon R'_p \to S'_p$ and a ciphertext over R'_q encrypting (with sufficiently small error term) a message $\mu \in R'_p$, the output is a ciphertext over S'_q encrypting $L(\mu) \in S'_p$.*

The security of the procedure described in Proposition 2.3 is based on the hardness of the ring-LWE problem in S', so the dimension of S' must be sufficiently large. The procedure itself is quite simple and efficient: it first switches to a secret key that lies in the subring S', then it multiplies the resulting ciphertext by an appropriate fixed element of R' (which is determined solely by the function L). Finally, it applies to the ciphertext the trace function $\mathrm{Tr}_{R'/S'}\colon R' \to S'$. All of these operations are quasi-linear time in the dimension of $R'/\mathbb{Z}$, and very efficient in practice. In particular, the trace is a trivial linear-time operation when elements are represented in any of the bases we use. The ring-switching procedure increases the effective error rate of the ciphertext by a factor of about the square root of the dimension of R', which is comparable to that of a single homomorphic multiplication. See [GHPS12] for further details.

3 Overview of Bootstrapping Procedure

Here we give a high-level description of our bootstrapping procedure. We present a unified procedure for non-packed, packed, and semi-packed ciphertexts, but note that for non-packed ciphertexts, Steps 3a and 3c (and possibly 1c) are null operations, while for packed ciphertexts, Steps 1b, 1c, and 2 are null operations.

Recalling the cryptosystem from Section 2.2, the plaintext ring is R_p and the ciphertext ring is R'_q, where $R = \mathcal{O}_m \subseteq R' = \mathcal{O}_{m'}$ are cyclotomic rings (so $m|m'$), and q is a power of p. The procedure also uses a larger cyclotomic ring $R'' = \mathcal{O}_{m''} \supseteq R'$ (so $m'|m''$) to work with ciphertexts that encrypt elements of the original ciphertext ring R'_q. To obtain quasilinear runtimes and exponential hardness (under standard hardness assumptions), our procedure imposes some mild conditions on the indices m, m', and m'':

- The dimension $\varphi(m'')$ of R'' must be quasilinear, so we can represent elements of R'' efficiently.
- For Steps 2 and 3, all the prime divisors of m and m' must be small (i.e., polylogarithmic).
- For Step 3, m and m''/m must be coprime, which implies that m and m'/m must be coprime also. Note that the former condition is always satisfied for non-packed ciphertexts (where $m = 1$). For packed ciphertexts (where $m = m'$), the latter condition is always satisfied, which makes it easy to choose a valid m''. For semi-packed ciphertexts (where $1 < m < m'$), we can always satisfy the latter condition either by increasing m (at a small expense in practical efficiency in Step 3; see Section 5.1.3), or by effectively decreasing m slightly (at a possible improvement in practical efficiency; see Section 3.2).

For example, when $m = 1$, both m' and m'' can be powers of two.

The input to the procedure is a ciphertext $c' = (c'_0, c'_1) \in (R'_q)^2$ that encrypts some plaintext $\mu \in R_p$ under a secret key $s' \in R'$, i.e., it satisfies the relation

$$v' = c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \pmod{qR'}$$

for some small enough error term $e' \in R'$. The procedure computes a new encryption of $\lfloor v' \rceil_p = \mu$ (under some secret key, not necessarily s') that has substantially smaller noise rate than the input ciphertext. It proceeds as follows (explanatory remarks appear in italics):

1. Convert c' to a "noiseless" ciphertext c'' over a large ring R''_Q that encrypts a plaintext $(g'/g)u' \in R'_{q'}$, where $g' \in R'$, $g \in R$ and $\hat{m}, \hat{m}' \in \mathbb{Z}$ are as defined in (and following) Equation (2.1), $q' = (\hat{m}'/\hat{m})q$, and $u' = v' \pmod{qR'}$. This proceeds in the following sub-steps (see Section 3.1 for further details).

   Note that $g'/g \in R'$ by definition, and that it divides $\hat{m}'/\hat{m}$.

   (a) Reinterpret c' as a noiseless encryption of $v' = \frac{q}{p} \cdot \mu + e' \in R'_q$ as a *plaintext*, noting that both the plaintext and ciphertext rings are now taken to be R'_q.

   This is purely a conceptual change in perspective, and does not involve any computation.

   (b) Using the procedure described in Section 2.2.2, change the plaintext (and ciphertext) modulus to $q' = (\hat{m}'/\hat{m})q$, yielding a noiseless encryption of some $u' \in R'_{q'}$ such that $u' = v' \pmod{qR'}$.

   Note that this step is a null operation if the original ciphertext was packed, i.e., if $m = m'$.

   We need to increase the plaintext modulus because homomorphically computing $\mathrm{Tr}_{R'/R}$ in Step 2 below introduces an $\hat{m}'/\hat{m}$ factor into the plaintext, which we will undo by scaling the plaintext modulus back down to q. (See Section 3.2 for an alternative choice of q'.)

   (c) Multiply the ciphertext from the previous step by $g'/g \in R'$, yielding a noiseless encryption of plaintext $(g'/g)u' \in R'_{q'}$.

   The factor $(g'/g) \in R'$ is needed when we homomorphically compute $\mathrm{Tr}_{R'/R}$ in Step 2 below. Note that $g'/g = 1$ if and only if every odd prime divisor of m' also divides m, e.g., if $m = m'$.

(d) Convert to a noiseless ciphertext c'' that still encrypts $(g'/g)u' \in R'_{q'}$, but using a large enough ciphertext ring R''_Q for some $R'' = \mathcal{O}_{m''} \supseteq R'$ and modulus $Q \gg q'$.

A larger ciphertext ring R''_Q is needed for security in the upcoming homomorphic operations, to compensate for the low noise rates that will need to be used. These operations will expand the initial noise rate by a quasipolynomial $\lambda^{O(\log \lambda)}$ factor in total, so the dimension of R'' and the bit length of Q can be $\tilde{O}(\lambda)$ and $\tilde{O}(1)$, respectively.

The remaining steps are described here only in terms of their effect on the *plaintext* value and ring. Using ring- and modulus-switching, the ciphertext ring R'' and modulus Q may be made smaller as is convenient, subject to the security and functionality requirements. (Also, the ciphertext ring implicitly changes during Steps 3a and 3c.)

2. Homomorphically apply the scaled trace function $(\hat{m}/\hat{m}') \operatorname{Tr}_{R'/R}$ to the encryption of $(g'/g)u' \in R'_{q'}$, to obtain an encryption of plaintext

$$u = \frac{\hat{m}}{\hat{m}'} \cdot \operatorname{Tr}_{R'/R}\Big(\frac{g'}{g} \cdot u'\Big) = \frac{q}{p} \cdot \mu + e \in R_q$$

for some suitably small error term $e \in R$. See Section 4 for further details.

This step changes the plaintext ring from $R'_{q'}$ to R_q, and homomorphically isolates the noisy R_q-encoding of μ. It is a null operation if the original ciphertext was packed, i.e., if $m = m'$.

3. Homomorphically apply the ring rounding function $\lfloor \cdot \rceil_p \colon R_q \to R_p$, yielding an output ciphertext that encrypts $\lfloor u \rceil_p = \mu \in R_p$. This proceeds in three sub-steps, all of which are applied homomorphically (see Section 5 for details):

   (a) Map the coefficients u_j of $u \in R_q$ (with respect to the decryption basis B of R) to the $\mathbb{Z}_q$-slots of a ring S_q, where S is a suitably chosen cyclotomic.

   This step changes the plaintext ring from R_q to S_q. It is a null operation if the original ciphertext was non-packed (i.e., if $m = 1$), because we can let $S = R = \mathbb{Z}$.

   (b) Batch-apply the integer rounding function $\lfloor \cdot \rceil \colon \mathbb{Z}_q \to \mathbb{Z}_p$ to the $\mathbb{Z}_q$-slots of S_q, yielding a ciphertext that encrypts the values $\mu_j = \lfloor u_j \rceil_p \in \mathbb{Z}_p$ in its $\mathbb{Z}_p$-slots.

   This step changes the plaintext ring from S_q to S_p. It constitutes the only non-linear operation on the plaintext, with multiplicative depth $\lceil \lg p \rceil \cdot (\log_p(q) - 1) \approx \log(q)$, and as such is the most expensive in terms of runtime, noise expansion, etc.

   (c) Reverse the map from the step 3a, sending the values μ_j from the $\mathbb{Z}_p$-slots of S_p to coefficients with respect to the decryption basis B of R_p, yielding an encryption of $\mu = \sum_j \mu_j b_j \in R_p$.

   This step changes the plaintext ring from S_p to R_p. Just like step 3a, it is a null operation for non-packed ciphertexts.

3.1 Obtaining a Noiseless Ciphertext

Step 1 of our bootstrapping procedure is given as input a ciphertext $c' = (c'_0, c'_1)$ over R'_q that encrypts (typically with a high noise rate) a message $\mu \in R_p$ under key $s' \in R'$, i.e., $v' = c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \in R'_q$ for some error term e'. We first change our perspective and view c' as a "noiseless" encryption (still under s')

of the plaintext value $v' \in R'_q$, taking both the plaintext and ciphertext rings to be R'_q. This view is indeed formally correct, because

$$c'_0 + c'_1 \cdot s' = \frac{q}{q} \cdot v' + 0 \pmod{qR'}.$$

Next, in preparation for the upcoming homomorphic operations we increase the plaintext (and ciphertext) modulus to q', and multiply the resulting ciphertext by g'/g. These operations clearly preserve noiselessness. Finally, we convert the ciphertext ring to R''_Q for a sufficiently large cyclotomic $R'' \supseteq R'$ and modulus $Q \gg q$ that is divisible by q. This is done by simply embedding R' into R'' and introducing extra precision, i.e., scaling the ciphertext up by a Q/q factor. It is easy to verify that these operations also preserve noiselessness.

3.2 Variants and Optimizations

Our basic procedure admits a few minor variants and practical optimizations, which we discuss here.

Smaller temporary modulus q'. In Step 1b we increase the plaintext modulus from q to $q' = rq$ where $r = \hat{m}'/\hat{m}$, and at the end of Step 2 we reduce the modulus back to q because the plaintext is divisible by r. The net effect of this, versus using a modulus q throughout, is that the modulus Q is larger by an r factor, as are the error rates used for key-switching in Step 2. This does not affect the asymptotic cost of bootstrapping, but it may have a small impact in practice. Instead, we can increase the modulus to only $q' = (r/d)q$, where d is the largest divisor of r coprime with q. Then in Step 2 we can remove an (r/d) factor from the plaintext by scaling the modulus back down to q, and keep track of the remaining d factor and remove it upon decryption. (We could also remove the d factor by multiplying the ciphertext by $d^{-1} \bmod q$, but this would increase the noise rate by up to a $q/2$ factor, which is typically much larger than the $\hat{m}'/\hat{m}$ factor we were trying to avoid in the first place.)

Using a smaller index m in Steps 2 and 3. Steps 3a and 3c can be much more costly in practice than Step 2, because they require working with rings that have at least $\varphi(m)$ $\mathbb{Z}_q$-slots. As the number of needed slots increases, the indices of such rings tend to grow quickly, and involve more prime divisors of larger size (though asymptotically the indices remain quasilinear); see Appendix C for some examples. So, in practice it may be faster to invoke Step 3 *a few times* to evaluate the rounding function over a *smaller* ring $\tilde{R} = \mathcal{O}_{\tilde{m}} \subset R$, for some proper divisor $\tilde{m}$ of m. Our procedure can be adapted to work in this way, even if the original plaintext μ is an arbitrary element of the plaintext space R_p.

The main facts we use are that the decryption basis B of R factors as $B = B' \cdot \tilde{B}$, where $\tilde{B}$ is the decryption basis of $\tilde{R}$, and in particular B' is an optimally short $\tilde{R}$-basis of R. (See Section 2.1.4.) Moreover, applying the ring rounding function on any $u \in R_q$ is equivalent to independently applying the ring rounding function on each of u's $\tilde{R}_q$-coefficients with respect to B'. Lastly, the $\tilde{R}_q$-coefficients of u can be individually extracted using the trace function $\mathrm{Tr}_{R/\tilde{R}}$ on certain fixed (short) multiples of u. (This all just generalizes the case $\tilde{R} = \mathbb{Z}$ in the natural way.) Using these facts, in Step 2 we can homomorphically apply $\mathrm{Tr}_{R/\tilde{R}}$ several times to obtain encryptions of the $\tilde{R}_q$-coefficients of the noisy encoding $u \approx (q/p) \cdot \mu$, then use Step 3 to homomorphically round those coefficients to get the $\tilde{R}_p$-coefficients of $\mu \in R_p$, and finally reassemble the pieces by homomorphically multiplying by the short basis elements in B', and summing the results.

Note that the above method requires evaluating $\mathrm{Tr}_{R/\tilde{R}}$ a total of $\varphi(m)/\varphi(\tilde{m})$ times in Step 2, and the same goes for the $\tilde{R}_q$ rounding function in Step 3. Because each evaluation takes quasilinear time no matter what $\tilde{m}$ is, the asymptotic performance can only worsen as $\tilde{m}$ decreases. However, in practice there may be benefits in choosing $\tilde{m}$ to be slightly smaller than m.

4 Homomorphic Trace

Here we show how to perform Step 2 of our bootstrapping procedure, which homomorphically evaluates the scaled trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$ on an encryption of $(g'/g)u' \in R'_{q'}$, where recall that: $g' \in R', g \in R$ are as defined in Equation (2.1), and (g'/g) divides $(\hat{m}'/\hat{m})$; the plaintext modulus is $q' = (\hat{m}'/\hat{m})q$; and

$$u' = v' = \frac{q}{p} \cdot \mu + e' \pmod{qR'},$$

where $e' \cdot g' \in g'R'$ is sufficiently short. Our goal is to show that:

1. the scaled trace of the plaintext $(g'/g)u'$ is some $u = \frac{q}{p} \cdot \mu + e \in R_q$, where $e \cdot g \in gR$ is short, and
2. we can efficiently homomorphically apply the scaled trace on a ciphertext c'' over some larger ring $R'' = \mathcal{O}_{m''} \supseteq R'$.

4.1 Trace of the Plaintext

We first show the effect of the scaled trace on the plaintext $(g'/g)u' \in R'_{q'}$. By the above description of $u' \in R'_{q'}$ and the fact that $(g'/g)q$ divides $q' = (\hat{m}'/\hat{m})q$, we have

$$(g'/g)u' = (g'/g)v' = (g'/g)\left(\frac{q}{p} \cdot \mu + e'\right) \pmod{(g'/g)qR'}.$$

Therefore, letting $\mathrm{Tr} = \mathrm{Tr}_{R'/R}$, by R-linearity of the trace and Lemma 2.1, we have

$$\begin{aligned}\mathrm{Tr}((g'/g)u') &= \mathrm{Tr}(g'/g) \cdot \frac{q}{p} \cdot \mu + \mathrm{Tr}(e' \cdot g')/g \\ &= \frac{\hat{m}'}{\hat{m}}\left(\frac{q}{p} \cdot \mu + e\right) \pmod{q'R},\end{aligned}$$

where $e = (\hat{m}/\hat{m}')\,\mathrm{Tr}(e' \cdot g')/g \in R$. Therefore, after scaling down the plaintext modulus q' by an $\hat{m}'/\hat{m}$ factor (see Section 2.2.2), the plaintext is $\frac{q}{p} \cdot \mu + e \in R_q$.

Moreover, $e \cdot g = (\hat{m}/\hat{m}')\,\mathrm{Tr}(e' \cdot g') \in gR$ is short because $e' \cdot g' \in g'R'$ is short; see, e.g., [GHPS12, Corollary 2.2]. In fact, by basic properties of the decoding/decryption basis (as defined in [LPR13]) under the trace, the coefficient vector of e with respect to the decryption basis of R is merely a subvector of the coefficient vector of e' with respect to the decryption basis of R'. Therefore, e is within the error tolerance of the rounding function on R_q, assuming e' is within the error tolerance of the rounding function on R'_q.

4.2 Applying the Trace

Now we show how to efficiently homomorphically apply the scaled trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$ to an encryption of any plaintext in $R'_{q'}$ that is divisible by (g'/g). Note that this condition ensures that the output of the trace is a multiple of $\hat{m}/\hat{m}'$ in $R_{q'}$ (see Lemma 2.1), making the scaling a well-defined operation that results in an element of R_q.

First recall that $\mathrm{Tr}_{R'/R}$ is the sum of all $\varphi(m')/\varphi(m)$ automorphisms of R'/R, i.e., automorphisms of R' that fix R pointwise. Therefore, one way of homomorphically computing the scaled trace is to homomorphically apply the proper automorphisms, sum the results, and scale down the plaintext and its modulus. While this "sum-automorphisms" procedure yields the correct result, computing the trace in this way does not run in quasilinear time, unless the number $\varphi(m')/\varphi(m)$ of automorphisms is only polylogarithmic.

Instead, we consider a sufficiently fine-grained tower of cyclotomic rings

$$R^{(r)}/\cdots/R^{(1)}/R^{(0)},$$

where $R' = R^{(r)}$, $R = R^{(0)}$, and each $R^{(i)} = \mathcal{O}_{m_i}$, where m_i is divisible by m_{i-1} for $i > 0$; for the finest granularity we would choose the tower so that every m_i/m_{i-1} is prime. Notice that the scaled trace function $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$ is the composition of the scaled trace functions $(\hat{m}_{i-1}/\hat{m}_i)\,\mathrm{Tr}_{R^{(i)}/R^{(i-1)}}$, and that g'/g is the product of all $g^{(i)}/g^{(i-1)}$ for $i = 1, \ldots, r$, where $g^{(i)} \in R^{(i)}$ is as defined in Equation (2.1). So, another way of homomorphically applying the full scaled trace is to apply the corresponding scaled trace in sequence for each level of the tower, "climbing down" from $R' = R^{(r)}$ to $R = R^{(0)}$. In particular, if we use the above sum-automorphisms procedure with a tower of finest granularity, then there are at most $\log_2(m'/m) = O(\log \lambda)$ levels, and since we have assumed that every prime divisor of m'/m is bounded by polylogarithmic in the security parameter λ, the full procedure will run in quasilinear $\tilde{O}(\lambda)$ time.

For technical reasons related to the analysis of noise terms under automorphisms, we actually use the sum-automorphisms procedure only on levels $R^{(i)}/R^{(i-1)} = \mathcal{O}_{m_i}/\mathcal{O}_{m_{i-1}}$ of the tower where every odd prime dividing m_i also divides m_{i-1}. Otherwise, we instead apply the scaled trace via an alternative procedure using ring-switching, which has essentially the same runtime (see Section 4.2.2 below for details). In fact, the alternative procedure can actually be used for *any* level of the tower, but it has the slight disadvantage of requiring the index of the ciphertext ring to be divisible by at least one prime that does not divide m_i; this is why we prefer not to use it when, e.g., m_i is a power of two.

4.2.1 Details of the Sum-Automorphisms Procedure

Here we specify the procedure for homomorphically applying the scaled trace by summing automorphisms, as sketched above. Let $R'/R = \mathcal{O}_{m'}/\mathcal{O}_m$ be a cyclotomic extension, where here m, m' are just dummy indices, not necessarily the ones from above. As already mentioned, we require that every odd prime dividing m' also divides m. The procedure takes as input a ciphertext c'' over some $R'' \supseteq R'$ that encrypts a plaintext $w' \in R'_{q'}$ under secret key $s'' \in R''$, where $q' = (\hat{m}'/\hat{m})q$ and w' is divisible by (g'/g). It proceeds as follows:

1. Compute ciphertexts $\tau_i(c'')$ over R'' for a certain set of automorphisms τ_i of R''/R that induce the automorphisms of R'/R. These ciphertexts will respectively encrypt $\tau_i(w') \in R'_{q'}$ under secret key $\tau_i(s'')$. Then key-switch [BV11a, BGV12] these to ciphertexts $c^{(i)}$ encrypting $\tau_i(w')$ under a common secret key $\tilde{s}$. See below for further details.

2. Sum the ciphertexts $c^{(i)}$ (component-wise) to get a new ciphertext $\tilde{c}$ that encrypts (under secret key $\tilde{s}$) the plaintext $\mathrm{Tr}_{R'/R}(w') = \sum_i \tau_i(w') \in R_{q'}$, which is divisible by $\hat{m}'/\hat{m}$.

3. Using the procedure from Section 2.2.2, reduce the plaintext modulus to q, resulting in a ciphertext that encrypts the scaled trace $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}(w') \in R_q$ under $\tilde{s}$.

The correctness of Steps 2 and 3 is immediate, so we just need to give the details of Step 1. We need to choose automorphisms τ_i of R''/R that induce the automorphisms of R'/R. Recall that the latter are defined by $\tau_j(\zeta_{m'}) = \zeta_{m'}^j$ for all $j \in \mathbb{Z}^*_{m'}$ such that $j = 1 \pmod m$. For each such j, we choose an $i \in \mathbb{Z}^*_{m''}$ such that $i = j \pmod{m'}$ and such that i is 1 modulo every prime p that divides m'' but not m'; this is possible by the Chinese Remainder Theorem. Then $\tau_i(\zeta_{m''}) = \zeta_{m''}^i$ is an automorphism of R''/R that induces τ_j, because $i = 1 \pmod m$ and

$$\tau_i(\zeta_{m'}) = \zeta_{m''}^{(m''/m')i} = \zeta_{m'}^j.$$

Also, by our assumption on m, m', each i we use is 1 modulo every prime that divides m'', because every such prime either divides m, or does not divide m', or is 2.

To complete the details of Step 1, we need to show why the ciphertext $\tau_i(c'')$ encrypts $\tau_i(w') \in R'_{q'}$ under secret key $\tau_i(s'')$. This follows from the decryption relation for c'', and the fact that τ_i is a ring homomorphism that induces an automorphism of R' and fixes $\mathbb{Z} \subseteq R$ pointwise:

$$\tau_i(c''_0) + \tau_i(c''_1) \cdot \tau_i(s'') = \frac{q}{p} \cdot \tau_i(\mu) + \tau_i(e''),$$

where the error term $e'' \in R''$ of c'' is such that $e'' \cdot g''$ is short (under the canonical embedding of R'').

The only subtlety is that we need $\tau_i(e'') \cdot g''$ to be short. We show below that $g'' = \tau_i(g'')$, from which it follows that $\tau_i(e'') \cdot g'' = \tau_i(e'' \cdot g'')$, which is short because the automorphisms of R'' simply permute the coordinates of the canonical embedding, and hence preserve norms (see, e.g., [LPR10, Lemma 5.6]). To see that $g'' = \tau_i(g'')$, recall that $i \in \mathbb{Z}^*_{m''}$ is 1 modulo every prime p that divides m''. Therefore, τ_i fixes every ζ_p and hence also fixes g''.[6]

Lastly, we briefly analyze the efficiency of the procedure. Applying automorphisms to the ciphertext ring elements is a trivial linear-time operation in the dimension, when the element is represented in any of the structured bases we consider (and also in the so-called "Chinese remainder" basis). Similarly, key-switching is quasilinear time in the bit length of the ciphertext, which itself is quasilinear in our context.

4.2.2 Applying the Trace via Ring-Switching

Here we describe the alternative procedure for applying the scaled trace, which uses the ring-switching technique from [GHPS12] (see Proposition 2.3). Let $R'/R = \mathcal{O}_{m'}/\mathcal{O}_m$ be an arbitrary cyclotomic extension, where m, m' are again dummy variables. For this procedure, we require that the ciphertext ring $R'' = \mathcal{O}_{m''} \supseteq R'$ be such that m''/m' is coprime with m', but otherwise we can choose m'' however we like. As before, the input is a ciphertext c'' over R'' that encrypts a plaintext $w' \in R'_{q'}$, where w' is divisible by (g'/g).

The main idea is that since m' and m''/m' are coprime, we can write $R'' \cong R' \otimes U$ where $U = \mathcal{O}_{m''/m'}$ and the tensor product is over the largest common base ring $\mathbb{Z}$. Then the R-linear function $\mathrm{Tr}_{R'/R}$ is induced by the $(R \otimes U)$-linear function $L\colon (R' \otimes U) \to (R \otimes U)$ defined by $L(a' \otimes u) = \mathrm{Tr}_{R'/R}(a') \otimes u$ for all $a' \in R', u \in U$. So, using the ring-switching procedure from Proposition 2.3, we can homomorphically evaluate L on ciphertext c'', yielding an encryption of $\mathrm{Tr}_{R'/R}(w')$, and then scale down the plaintext and its modulus as usual. One nice fact we highlight is that using ring-switching to evaluate the function $\mathrm{Tr}_{R'/R}$ does not incur *any* multiplicative increase in the noise rate, only a small additive one from the key-switching step. This is because the factor associated with the function $\mathrm{Tr}_{R'/R}$ that is applied to the ciphertext in the ring-switching procedure is simply 1.

One very important point is that ring-switching requires ring-LWE to be hard over the target ring $\mathcal{O}_{m'' \cdot m/m'} \cong R \otimes U$, so its dimension must be sufficiently large, but at the same time we cannot make the dimension of $R'' = \mathcal{O}_{m''}$ too large, for efficiency reasons. Therefore, we only use the procedure when m'/m is small, and for sufficiently large m''. Note that if the m'' associated with a given input ciphertext is too small, we can trivially increase it by embedding into a larger cyclotomic ring.

[6] If, contrary to our assumption, m' was divisible by one or more primes that did not divide m, then the error term $\tau_i(e'' \cdot g'')$ appearing in the ciphertext would be accompanied by a factor of $g''/\tau_i(g'')$. The expansion associated with this term can be bounded and is not excessive, but it depends on the number and sizes of the primes dividing m' and not m. By contrast, the alternative procedure described in Section 4.2.2 incurs no multiplicative increase in the noise rate.

5 Homomorphic Ring Rounding

In this section we describe how to efficiently homomorphically evaluate the "ring rounding function" $\lfloor \cdot \rceil_p \colon R_q \to R_p$, where $R = \mathcal{O}_m$ is the mth cyclotomic ring. Conceptually, we follow the high-level strategy from [GHS12a], but instantiate it with very different technical components. Recall from Section 2.2.1 that the rounding function expresses its input u in the "decryption" $\mathbb{Z}$-basis $B = \{b_j\}$ of R, as $u = \sum_j u_j \cdot b_j$ for $u_j \in \mathbb{Z}_q$, and outputs $\lfloor u \rceil_p := \sum_j \lfloor u_j \rceil_p \cdot b_j \in R_p$. Unlike with integer rounding from $\mathbb{Z}_q$ to $\mathbb{Z}_p$, it is not clear whether this rounding function has a low-depth arithmetic formula using just the ring operations of R. One difficulty is that there are an *exponentially* large number of values in R_q that map to a given value in R_p, which might be seen as evidence that a corresponding arithmetic formula must have large depth. Fortunately, we show how to circumvent this issue by using an additional homomorphic operation, namely, an enhancement of ring-switching. In short, we reduce the homomorphic evaluation of the ring rounding function (from R_q to R_p) very simply and efficiently to that of several parallel (batched) evaluations of the integer rounding function (from $\mathbb{Z}_q$ to $\mathbb{Z}_p$).

5.1 Overview

Suppose we choose some cyclotomic ring $S = \mathcal{O}_\ell$ having a mod-q CRT set $C = \{c_j\} \subset S$ of cardinality exactly $|B|$. That is, we have a ring embedding from the product ring $\mathbb{Z}_q^{|B|}$ into S_q, given by $\mathbf{u} \mapsto \sum_j u_j \cdot c_j$. Note that the choice of the ring S is at our convenience, and need not have any relationship to the plaintext ring R_q. We express the rounding function $R_q \to R_p$ as a sequence of three steps:

1. Map $u = \sum_j u_j \cdot b_j \in R_q$ to $\sum_j u_j \cdot c_j \in S_q$, i.e., send the $\mathbb{Z}_q$-coefficients of u (with respect to the decryption basis B) to the $\mathbb{Z}_q$-slots of S_q.
2. Batch-apply the integer rounding function from $\mathbb{Z}_q$ to $\mathbb{Z}_p$ to the slot values u_j, to get $\sum_j \lfloor u_j \rceil_p \cdot c_j \in S_2$.
3. Invert the map from the first step to obtain $\lfloor u \rceil_p = \sum_j \lfloor u_j \rceil_2 \cdot b_j \in R_2$.

Using batch/SIMD operations [SV11], the second step is easily achieved using the fact that S_q embeds the product of several copies of the ring $\mathbb{Z}_q$, via the CRT elements c_j. That is, we can simultaneously round all the coefficients u_j to $\mathbb{Z}_p$, using just one evaluation of an arithmetic procedure over S corresponding to one for the integer rounding function from $\mathbb{Z}_q$ to $\mathbb{Z}_p$.

We now describe one way of expressing the first and third steps above, in terms of operations that can be evaluated homomorphically. The first simple observation is that the function mapping $u = \sum_j u_j \cdot b_j$ to $\sum_j u_j \cdot c_j$ is induced by a $\mathbb{Z}$-linear function $\bar{L} \colon R \to S$. Specifically, $\bar{L}$ simply maps each $\mathbb{Z}$-basis element b_j to c_j. Now suppose that we choose S so that its largest common subring with R is $\mathbb{Z}$, i.e., the indices m, ℓ are coprime. Then letting $T = R + S = \mathcal{O}_{m\ell} \cong R \otimes S$ be the compositum ring, Lemma 2.2 yields an S-linear function $L \colon T \to S$ that coincides with $\bar{L}$ on $R \subseteq T$, and in particular on u. The ring-switching procedure from Proposition 2.3 can homomorphically evaluate any S-linear function from T to S, and in particular, the function L. Therefore, by simply embedding R into T, we can homomorphically evaluate $\bar{L}(x) = L(x)$ by applying the ring-switching procedure with L.

Unfortunately, there is a major problem with the efficiency of the above approach: the *dimension* (over $\mathbb{Z}$) of the compositum ring T is the product of those of R and S, which are each at least linear in the security parameter. Therefore, representing and operating on arbitrary elements in T requires at least quadratic time.

5.1.1 Efficiently Mapping from B to C

In hindsight, the quadratic runtime of the above approach should not be a surprise, because we treated $\bar{L}\colon R \to S$ as an arbitrary $\mathbb{Z}$-linear transformation, and B, C as arbitrary sets. To do better, $\bar{L}$, B, and C must have some structure we can exploit. Fortunately, they can—*if* we choose them carefully. We now describe a way of expressing the first and third steps above in terms of simple operations that can be evaluated homomorphically in quasilinear time.

The main idea is as follows, and is summarized in Figure 1. Instead of mapping directly from R to S, we will express $\bar{L}$ as a *sequence* of linear transformations $\bar{L}_1, \ldots, \bar{L}_r$ through several "hybrid" cyclotomic rings $R = H^{(0)}, H^{(1)}, \ldots, H^{(r)} = S$. For sets B and C with an appropriate product structure, these transformations will respectively map $A_0 = B \subset H^{(0)}$ to some structured subset $A_1 \subset H^{(1)}$, then A_1 to some structured subset $A_2 \subset H^{(2)}$, and so on, finally mapping A_{r-1} to $A_r = C \subset H^{(r)}$. In contrast to the inefficient method described above, the hybrid rings will be chosen so that each compositum $T^{(i)} = H^{(i-1)} + H^{(i)}$ of adjacent rings has dimension just *slightly larger* (by only a polylogarithmic factor) than that of R. This is achieved by choosing the indices of $H^{(i-1)}, H^{(i)}$ to have large greatest common divisor, and hence small least common multiple. For example, the indices can share almost all the same prime divisors (with multiplicity), and have just one different prime divisor each. Of course, other tradeoffs between the number of hybrid rings and the dimensions of the compositums are also possible.

The flip side of this approach is that using ring-switching, we can homomorphically evaluate only $E^{(i)}$-*linear* functions $\bar{L}_i\colon H^{(i-1)} \to H^{(i)}$, where $E^{(i)} = H^{(i-1)} \cap H^{(i)}$ is the largest common subring of adjacent hybrid rings. Since each $E^{(i)}$ is large by design (to keep the compositum $T^{(i)}$ small), this requirement is quite strict, yet we still need to construct linear functions $\bar{L}_i$ that sequentially map $B = A_0$ to $C = A_r$. To achieve this, we construct all the sets A_i to have appropriate product structure. Specifically, we ensure that for each $i = 1, \ldots, r$, we have factorizations

$$A_{i-1} = A_{i-1}^{\text{out}} \cdot Z_i, \quad A_i = A_i^{\text{in}} \cdot Z_i \tag{5.1}$$

for some set $Z_i \subset E^{(i)}$, where both A_{i-1}^{out} and A_i^{in} are linearly independent over $E^{(i)}$. (Note that for $1 \le i < r$, each A_i needs to factor in two ways over two subrings $E^{(i-1)}$ and $E^{(i)}$, which is why we need two sets A_i^{in} and A_i^{out}.) Then, we simply define $\bar{L}_i$ to be an arbitrary $E^{(i)}$-linear function that bijectively maps A_{i-1}^{out} to A_i^{in}. (Note that A_{i-1}^{out} and A_i^{in} have the same cardinality, because A_{i-1} and A_i do.) It immediately follows that $\bar{L}_i$ bijectively maps A_{i-1} to A_i, because

$$\bar{L}_i(A_{i-1}) = \bar{L}_i(A_{i-1}^{\text{out}} \cdot Z_i) = \bar{L}_i(A_{i-1}^{\text{out}}) \cdot Z_i = A_i^{\text{in}} \cdot Z_i$$

by $E^{(i)}$-linearity and the fact that $Z_i \subset E^{(i)}$.

Summarizing the above discusion, we have the following theorem.

Theorem 5.1. *Suppose there exists a sequence of cyclotomic rings $R = H^{(0)}, H^{(1)}, \ldots, H^{(r)} = S$ and sets $A_i \subset H^{(i)}$ such that for all $i = 1, \ldots, r$, we have $A_{i-1} = A_{i-1}^{out} \cdot Z_i$ and $A_i = A_i^{in} \cdot Z_i$ for some sets $Z_i \subset E^{(i)} = H^{(i-1)} \cap H^{(i)}$ and A_{i-1}^{out}, A_i^{in} that are each $E^{(i)}$-linearly independent and of equal cardinality. Then there is a sequence of $E^{(i)}$-linear maps $\bar{L}_i\colon H^{(i-1)} \to H^{(i)}$, for $i = 1, \ldots, r$, whose composition $\bar{L}_r \circ \cdots \circ \bar{L}_1$ bijectively maps A_0 to A_r.*

5.1.2 Applying the Map Homomorphically

So far we have described how our desired map between *plaintext* rings R and S can be expressed as a sequence of linear maps through hybrid plaintext rings. In the context of bootstrapping, for security these

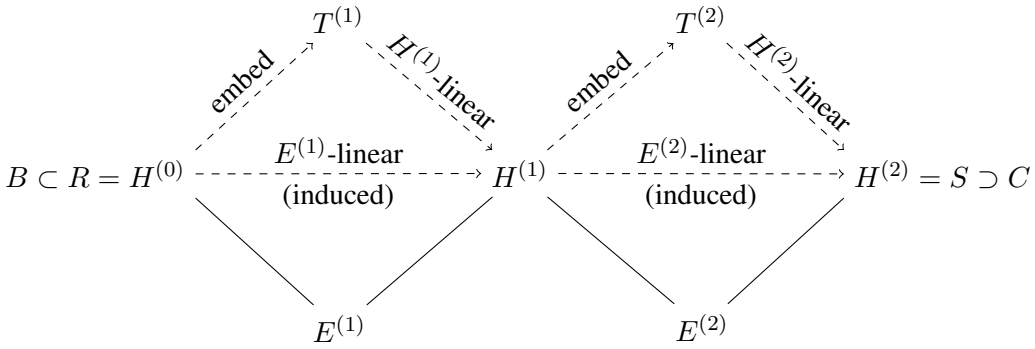

Figure 1: An example mapping from $B \subset R$ to $C \subset S$, via a sequence of hybrid rings. Each $E^{(i)} = H^{(i-1)} \cap H^{(i)}$ is a largest common subring, and each $T^{(i)} = H^{(i-1)} + H^{(i)}$ is a compositum, of adjacent hybrid rings. For any $E^{(i)}$-linear function from $H^{(i-1)}$ to $H^{(i)}$, there is a corresponding $H^{(i)}$-linear function from $T^{(i)}$ to $H^{(i)}$ that coincides with it on $H^{(i-1)}$ (see Lemma 2.2).

plaintext rings typically need to be embedded in some larger ciphertext rings, because the dimensions of R, S are not large enough to securely support the very small noise used in bootstrapping. For example, following Step 2 of our bootstrapping procedure (Section 3), we have a ciphertext over the ring R'' where $R'' = \mathcal{O}_{m''} \supseteq R$ for some m'' of our choice that is divisible by m. We need to choose the sequence of hybrid *ciphertext* rings so that they admit linear functions (over the respective largest common subrings) that induce the desired ones on the underlying *plaintext* rings. This turns out to be very easy to do, as we now explain.

Let H, H' be adjacent hybrid plaintext rings having largest common subring $E = H \cap H'$ and compositum $T = H + H'$. Then we want the corresponding ciphertext rings to be $\tilde{H} \cong H \otimes U$, $\tilde{H}' \cong H' \otimes U'$ for some cyclotomic rings U, U', and the largest common subring and compositum of $\tilde{H}, \tilde{H}'$ to be $\tilde{E} \cong E \otimes (U \cap U')$ and $\tilde{T} \cong T \otimes (U + U')$, respectively (where all the tensor products are over the common base ring $\mathbb{Z}$). Then any E-linear function $L\colon H \to H'$ is induced by any $\tilde{E}$-linear function $\tilde{L}\colon \tilde{H} \to \tilde{H}'$ satisfying $\tilde{L}(h \otimes 1) = L(h) \otimes 1$, which is the function we actually apply when switching between ciphertext rings.

To satisfy the above conditions, it is sufficient (and in fact necessary) to choose the respective indices u, u' of U, U' so that $\mathrm{lcm}(u, u')$ is coprime with $\mathrm{lcm}(h, h')$, where h, h' are the respective indices of H, H'. Then the ciphertext rings $\tilde{H}, \tilde{H}'$ have indices hu and $h'u'$, and their compositum has index $\mathrm{lcm}(h, h') \cdot \mathrm{lcm}(u, u')$, which must be quasilinear for asymptotic efficiency. In typical instantiations, in order to get enough additional slots in each successive ring, h'/h will be moderately large and $\mathrm{lcm}(h, h') \approx h'$. So to ensure that all the ciphertext rings are about the same size, we can choose $u/u' \approx h'/h$ and $\mathrm{lcm}(u, u') \approx u$.

5.1.3 Mapping Selected Coefficients

In some settings we may only need to map certain coefficients into slots, i.e., map a particular portion of B to a CRT set of appropriate size. For example, when bootstrapping a semi-packed ciphertext over $R' = \mathcal{O}_{m'}$ with plaintext over $\tilde{R} = \mathcal{O}_{\tilde{m}}$, we may need to artificially expand our view of the plaintext ring to some $R = \mathcal{O}_m$, so that m is coprime with m'/m (see the constraints listed at the start of Section 3). In such a case, the decryption basis B of R factors as $B = B' \cdot \tilde{B}$, where $\tilde{B}$ is the decryption basis of $\tilde{R}$ and $B' \subset R$ is a particular $\tilde{R}$-basis of R. Since the true message is really only over $\tilde{R}$, it can be shown that the only coefficients we need to recover the message are associated with the subset $b' \cdot \tilde{B} \subseteq B$ for a particular fixed

$b' \in B'$. Therefore, when designing the hybrid rings and CRT sets we only need $|\tilde{B}|$ slots in total. When initially switching from R through the hybrid rings, we do so in a way that maps b' to one entry of a CRT set and all the other elements of B' to zero, then continue by mapping all of $\tilde{B}$ to a CRT set as usual. Note that we still need to go through just as many hybrid rings to map from R to S, but the size of S can be significantly smaller because it needs fewer CRT slots.

5.2 Construction

By Theorem 5.1 and the ring-switching procedure, in order to map $B \subset R$ to a CRT set C of some ring S of our choice in a way that can be efficiently evaluated homomorphically, we just need to construct hybrid cyclotomic rings $R = H^{(0)}, H^{(1)}, \ldots, H^{(r)} = S$ and sets $A_i \subset H^{(i)}$ (where $A_0 = B$ and $A_r = C$) to satisfy the following two properties for each $i = 1, \ldots, r$:

1. Each compositum $T^{(i)} = H^{(i-1)} + H^{(i)}$ is not too large, i.e., its dimension is quasilinear.
2. The sets A_{i-1}, A_i factor as described in Equation (5.1).

The remainder of this subsection is dedicated to providing such a construction.

5.2.1 Decomposition of R and Basis $B \subset R$

For our given ring $R = \mathcal{O}_m$ and its $\mathbb{Z}$-basis B used in decryption, we consider a tower of cyclotomic rings

$$R^{(r)}/R^{(r-1)}/\cdots/R^{(1)}/R^{(0)},$$

where $R^{(r)} = R$ and $R^{(0)} = \mathcal{O}_1 = \mathbb{Z}$, and each $R^{(i)}$ has index m_i, which is divisible by m_{i-1} for $i > 0$. For example, in a finest-grained decomposition, r is the number of prime divisors (with multiplicity) of m, and the ratios m_i/m_{i-1} are all these prime divisors in some arbitrary order. A coarser-grained decomposition may be used as well, but will tend to make the compositum rings $T^{(i)}$ larger.

We need $\mathbb{Z}$-bases B_i of the rings $R^{(i)}$ that have a product structure induced by the tower. Specifically, for each $i = 1, \ldots, r$ we need to have the factorization

$$B_i = B'_i \cdot B_{i-1} \subset R^{(i)} \tag{5.2}$$

for some set $B'_i \subset R^{(i)}$ that is linearly independent over $R^{(i-1)}$. We also need the basis $B^{(r)}$ of $R = R^{(r)}$ to be the one used for decryption. As shown in Section 2.1.4, the scaled-up "decoding" basis of R has a finest-possible factorization, so we can use it as B for any choice of the tower.

We mention that the power basis $\{1, \zeta_m, \zeta_m^2, \ldots, \zeta_m^{\varphi(m)-1}\}$ of R, which is implicitly the one used when representing R as the polynomial ring $\mathbb{Z}[X]/\Phi_m(X)$, *does not* have the required product structure when m is divisible by two or more odd primes, but that it does coincide with the scaled-up decoding basis when m is a power of 2. (See [LPR13] for details.)

5.2.2 Ring S and CRT Set $C \subset S$.

We next design $S = \mathcal{O}_\ell$ so that it also yields a tower of cyclotomic rings $S^{(r)}/S^{(r-1)}/\cdots/S^{(1)}/S^{(0)}$, where $S^{(r)} = S$ and $S^{(0)} = \mathbb{Z}$, and each $S^{(i)}$ has index ℓ_i. As described in Sections 2.1.3 and 2.1.4, there are structured mod-q CRT sets $\tilde{C}_i$ of $S^{(i)}$ that factor as

$$\tilde{C}_i = \tilde{C}'_i \cdot \tilde{C}_{i-1},$$

where $\tilde{C}'_i \subset S^{(i)}$ is an $S^{(i-1)}$-linearly independent set whose cardinality is the "relative splitting number" of p in $S^{(i)}/S^{(i-1)}$, i.e., the number of distinct prime ideals in $S^{(i)}$ lying over any prime ideal divisor of p in $S^{(i-1)}$.

We need to choose the ring S and its tower so that for all $i = 1, \ldots, r$,

- the respective indices m_{r-i+1}, ℓ_i of $R^{(r-i+1)}, S^{(i)}$ are coprime (certainly it suffices for m and ℓ to be coprime, but this is not always necessary);
- the dimension $\varphi(m_{r-i+1} \cdot \ell_i)$ is not too large (specifically, it is quasi-linear in the security parameter);
- the relative splitting number $|\tilde{C}'_i| \geq |B'_{r-i+1}|$.

We can then easily define structured CRT sets $C_i \subset \tilde{C}_i \subset S^{(i)}$ of the appropriate cardinality, and in particular $C = C_r$, as follows. Define $C_0 = \{1\} \subset \mathbb{Z} = S^{(0)}$. Then for each $i = 1, \ldots, r$, let $C'_i \subseteq \tilde{C}'_i$ be an arbitrary subset having cardinality exactly $|B'_{r-i+1}|$, and define

$$C_i = C'_i \cdot C_{i-1} \subset \tilde{C}_i. \tag{5.3}$$

5.2.3 Hybrid Rings $H^{(i)}$ and Sets $A_i \subset H^{(i)}$

Informally, with each successive hybrid ring we remove another level from the R-tower and add on another level to the S-tower, and similarly with the corresponding components of the structured sets B and C. Formally, for $i = 0, 1, \ldots, r$ we define

$$\begin{aligned} H^{(i)} &= \mathcal{O}_{m_{r-i}\,\ell_i} \cong R^{(r-i)} \otimes S^{(i)}, \\ A_i &= B_{r-i} \cdot C_i \subset H^{(i)}, \end{aligned} \tag{5.4}$$

where the tensor product in Equation (5.4) applies to the rings as extensions of $\mathbb{Z}$, and the isomorphism holds because $\gcd(m_{r-i}, \ell_i) \leq \gcd(m_{r-i+1}, \ell_i) = 1$ by design. Note that $H^{(0)} = \mathcal{O}_{m_r} = R$, $H^{(r)} = \mathcal{O}_{\ell_r} = S$, and $A_0 = B_r = B$, $A_r = C_r = C$, as required.

For each $i = 1, \ldots, r$, because m_{r-i+1} and ℓ_i are coprime, it is straightforward to verify that the largest common subring $E^{(i)} = H^{(i-1)} \cap H^{(i)}$ and compositum $T^{(i)} = H^{(i-1)} + H^{(i)}$ are

$$\begin{aligned} E^{(i)} &= \mathcal{O}_{m_{r-i}\,\ell_{i-1}} \cong R^{(r-i)} \otimes S^{(i-1)} \\ T^{(i)} &= \mathcal{O}_{m_{r-i+1}\,\ell_i} \cong R^{(r-i+1)} \otimes S^{(i)}, \end{aligned}$$

where the tensor products above are over the common base ring $\mathbb{Z}$. Note that the dimension of $T^{(i)}/\mathbb{Z}$ is $\varphi(m_{r-i+1} \cdot \ell_i)$, which is quasi-linear in the security parameter by construction.

Lemma 5.2. *The sets A_{i-1}, A_i factor as in Equation* (5.1)*, i.e., $A_{i-1} = A^{out}_{i-1} \cdot Z_i$ and $A_i = A^{in}_i \cdot Z_i$ for some sets $Z_i \subset E^{(i)}$ and A^{out}_{i-1}, A^{in}_i that are each $E^{(i)}$-linearly independent and of equal cardinality.*

Proof. Define $Z_i = B_{r-i} \cdot C_{i-1} \subset E^{(i)}$. Recall from Equation (5.2) that $B_{r-i+1} = B'_{r-i+1} \cdot B_{r-i}$, where $B'_{r-i+1} \subset R^{(r-i+1)}$ is linearly independent over $R^{(r-i)} \subset H^{(i-1)}$, and hence also over $E^{(i)} \cong R^{(r-i)} \otimes S^{(i-1)}$ (because it corresponds to the set of pure tensors $B'_{r-i+1} \otimes \{1\} \subset R^{(r-i+1)} \otimes S^{(i-1)}$). Then

$$A_{i-1} = (B'_{r-i+1} \cdot B_{r-i}) \cdot C_{i-1} = B'_{r-i+1} \cdot Z_i$$

is the desired factorization. Similarly, recall from Definition (5.3) that $C_i = C'_i \cdot C_{i-1}$, where $C'_i \subseteq \tilde{C}'_i \subset S^{(i)}$ is linearly independent over $S^{(i-1)}$, and hence also over $E^{(i)}$. Then we have the desired factorization

$$A_i = B_{r-i} \cdot (C'_i \cdot C_{i-1}) = C'_i \cdot Z_i.$$

Finally, we have $|A^{\text{out}}_{i-1}| = |B'_{r-i+1}| = |C'_i| = |A^{\text{in}}_i|$ by design of C'_i. □

References

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ICTS*, pages 309–325. 2012.

[BPR12] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737. 2012.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, pages 868–886. 2012.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. 2011.

[BV11b] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524. 2011.

[CCK+13] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 315–335. 2013.

[Gen09a] C. Gentry. *A fully homomorphic encryption scheme*. Ph.D. thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009.

[GH11a] C. Gentry and S. Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *FOCS*, pages 107–109. 2011.

[GH11b] C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, pages 129–148. 2011.

[GHPS12] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart. Ring switching in BGV-style homomorphic encryption. In *SCN*, pages 19–37. 2012. Full version at `http://eprint.iacr.org/2012/240`.

[GHS12a] C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography*, pages 1–16. 2012.

[GHS12b] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, pages 465–482. 2012.

[GHS12c] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, pages 850–867. 2012.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 2013. To appear. Preliminary version in Eurocrypt 2010.

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT*, pages 35–54. 2013.

[SV11] N. Smart and F. Vercauteren. Fully homomorphic SIMD operations. Cryptology ePrint Archive, Report 2011/133, 2011. `http://eprint.iacr.org/`.

[vDGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43. 2010.

A Transformation Between LSB and MSB Encodings

Here we describe a folklore transformation between the "least significant bit" and "most significant bit" message encodings for (ring-)LWE-based cryptosystems.

Let plaintext modulus p and ciphertext modulus q be coprime, fix integers c_p, c_q such that $c_p p + c_q q = 1$, and observe that $c_p = p^{-1} \pmod q$ and $c_q = q^{-1} \pmod p$.

- An lsb encoding of a value $\mu \in \mathbb{Z}_p$ is any $v \in \mathbb{Z}_q$ such that $v = e \pmod q$ for some integer $e \in [-q/2, q/2)$ where $e = \mu \pmod p$.
- An msb encoding of μ is any $w \in \mathbb{Z}_q$ such that $\lfloor w \rceil_p := \lfloor w \cdot (p/q) \rceil = \mu \pmod p$.

If $v \in \mathbb{Z}_q$ is an lsb encoding of $\mu \in \mathbb{Z}_p$, then we claim that $p^{-1} \cdot v \in \mathbb{Z}_q$ is an msb encoding of $-q^{-1} \cdot \mu \in \mathbb{Z}_p$. Indeed, since $v = e \pmod q$ for some $e \in (\mu + p\mathbb{Z}) \cap [-q/2, q/2)$, we have

$$\lfloor p^{-1} \cdot v \rceil_p = \left\lfloor \tfrac{1 - c_q q}{p} \cdot e \cdot \tfrac{p}{q} \right\rceil = \left\lfloor (\tfrac{1}{q} - c_q) \cdot e \right\rceil = -c_q \cdot e = -q^{-1} \cdot \mu \pmod p.$$

In the other direction, if $w \in \mathbb{Z}_q$ is an msb encoding of $\mu \in \mathbb{Z}_p$, then we claim that $p \cdot w$ is an lsb encoding of $-q \cdot \mu \in \mathbb{Z}_p$. Indeed, by assumption we have

$$\lfloor w \rceil_p = \lfloor w \cdot (p/q) \rceil = w \cdot (p/q) - f = \mu \pmod p$$

for some $f \in \frac{1}{q}\mathbb{Z} \cap [-1/2, 1/2)$. Multiplying by q and letting $e = q \cdot f \in \mathbb{Z} \cap [-q/2, q/2)$, we have

$$p \cdot w - e = q \cdot \mu \pmod{pq}.$$

Reducing this modulo q, we get $p \cdot w = e \pmod q$, and reducing it modulo p, we have $e = -q \cdot \mu \pmod p$.

The above facts make it possible to convert between lsb and msb representations of (ring-)LWE ciphertexts, simply by multiplying the ciphertext by p or p^{-1} modulo q. This works because decryption recovers a $\mathbb{Z}_q$-encoding of the message simply as a linear function of the ciphertext, so the p or p^{-1} factor simply "passes through" the ciphertext to the encoding. (In the ring setting, the encoding of plaintext ring elements is coefficient-wise in a certain basis, so the same reasoning applies.) If $q = -1 \pmod p$, then the above transformations preserve the message exactly. In other cases, we can just keep track of the factors of $-q$ or $-q^{-1}$ introduced by the conversions (which may be affected by other homomorphic operations), and remove them upon decryption.

B Integer Rounding Procedure

Here we recall (a close variant of) the efficient arithmetic procedure from [GHS12a] for computing the "most significant bit" function $\mathrm{msb}_q \colon \mathbb{Z}_q \to \mathbb{Z}_2$ for $q = 2^\ell \geq 4$, defined as $\mathrm{msb}_q(x) = \lfloor x/(q/2) \rfloor$. Note that the integer rounding function $\lfloor \cdot \rceil_2 \colon \mathbb{Z}_q \to \mathbb{Z}_2$ is simply $\lfloor x \rceil_2 = \mathrm{msb}_q(x + q/4)$. The multiplicative depth and cost (in number of operations) of the msb_q procedure are not precisely analyzed in [GHS12a], and the procedure as written turns out to be suboptimal in depth and number of operations by $\log_2(q)$ factors, because it (homomorphically) raises ciphertexts to large powers in an inner loop. So for completeness, here we

present a simplified and optimized version of the procedure, and an analysis of its depth and cost. It uses the standard ring operations of $\mathbb{Z}_{2^j}$, as well as division by 2 of values that are guaranteed to be even. All of these operations can be evaluated homomorphically for the cryptosystem described in Section 2.2, as explained in Section 2.2.2. The procedure also easily generalizes to any prime base.

Algorithm 1 Arithmetic procedure for computing $\mathrm{msb}_q \colon \mathbb{Z}_q \to \mathbb{Z}_2$ [GHS12a]

Input: Element $x \in \mathbb{Z}_q$, where $q = 2^\ell$ for some positive integer ℓ
Output: $\mathrm{msb}_q(x) \in \mathbb{Z}_2$

1: $w_0 \leftarrow x$ // $w_0 \in \mathbb{Z}_q$
2: **for** $i \leftarrow 1, \ldots, \ell - 1$ **do**
3: $\quad y \leftarrow x$ // $y \in \mathbb{Z}_q$, $y = x \pmod{2^{i+1}}$
4: $\quad$ **for** $j \leftarrow 0, \ldots, i-1$ **do**
5: $\quad\quad w_j \leftarrow w_j^2$ // now $w_j = \mathrm{lsb}(\lfloor x/2^j \rfloor) \pmod{2^{i-j+1}}$
6: $\quad\quad y \leftarrow (y - w_j)/2 \bmod (q/2^{j+1})$ // now $y \in \mathbb{Z}_{q/2^{j+1}}$, $y = \lfloor x/2^{j+1} \rfloor \pmod{2^{i-j}}$
7: $\quad w_i \leftarrow y$ // $w_i \in \mathbb{Z}_{q/2^i}$, $w_i = \lfloor x/2^i \rfloor \pmod 2$
8: **return** $w_{\ell-1} \in \mathbb{Z}_2$

Correctness follows from [GHS12a, Lemma 2]. The main idea is that when initially assigned, each w_j has the same least-significant bit as $\lfloor x/2^j \rfloor$, i.e., $w_j = \lfloor x/2^j \rfloor \pmod 2$ (but its other bits may not agree with x's). Each time w_j is squared in Step 5, its least-significant bit remains the same, but an additional more-significant bit is set to zero. That is, after t squarings, $w_j = \mathrm{lsb}(\lfloor x/2^j \rfloor) \pmod{2^{t+1}}$. Therefore, in iteration i, the inner loop "shifts away" the i least-significant bits of x, leaving the $(i+1)$st bit intact in the least significant position (but possibly changing the others), at which point we can assign w_i and maintain the invariant.

We now briefly analyze the *homomorphic* evaluation of the procedure, in terms of its induced noise growth and runtime cost. The most important observation is that although it is written using a doubly nested loop, the procedure actually has multiplicative depth exactly $\ell - 1 = \log_2(q/2)$. This is because in the inner loop, each w_j for $j = 0, \ldots, i-1$ can be squared in parallel (Step 5). Each squaring of the plaintext value $w_j \in \mathbb{Z}_{q/2^j}$ induces the usual small polynomial expansion $(q/2^j) \cdot n^c$ (where $c \approx 1$) in the noise rate of the associated ciphertext. The iterated subtractions and divisions by 2 (Step 6) cause no growth at all in the noise rate: each subtraction sums (at worst) the noise rates of the associated ciphertexts, and division by 2 halves the noise rate.

In the ith iteration, the procedure performs i homomorphic multiplications and i subtractions (and also i divisions by 2, but these are trivial as homomorphic operations). Therefore, the procedure uses a total of $\ell(\ell-1)/2$ homomorphic multiplications and subtractions each.

C Concrete Choices of Rings

Here, for $p = 2$ and several values of the original cyclotomic index m, we give some workable values for the target cyclotomic index ℓ, along with the indices of the intermediate "hybrid" rings, the dimensions of the compositum rings, etc. In the tables below, and following the notation in Section 5:

- m_{r-i+1} is the index of the ring R_{r-i+1} at step i;
- ℓ_i is the index of the ring S_i at step i;

- $\varphi(m_{r-i+1} \cdot \ell_i)$ is the dimension of the compositum ring at step i;
- $|B'_{r-i+1}|$ is the dimension of the intermediate ring extension $R^{(r-i+1)}/R^{(r-i)}$;
- $|\tilde{C}'_i|$ is the "relative splitting number" of $p = 2$ in the extension $S^{(i)}/S^{(i-1)}$.

All the indices are *lower bounds* needed to support the *functionality* of the ring-rounding technique on the plaintext space (Section 5). Larger ciphertext indices may be required to ensure adequate security for all the homomorphic operations; see Section 5.1.2.

Table 1: Concrete choices for $m_r = 512, \varphi(m_r) = 256$

Step i	m_{r-i+1}	ℓ_i	$\lvert B'_{r-i+1}\rvert$	$\lvert\tilde{C}'_i\rvert$	$\varphi(m_{r-i+1} \cdot \ell_i)$
1	512	73	8	8	18432
2	64	$1387 = 73 \cdot 19$	8	9	41472
3	8	$9709 = 1387 \cdot 7$	4	6	38836
4	2	$29127 = 9709 \cdot 3$	2	2	15552
5	1	29127			15552

Table 2: Concrete choices for $m_r = 256, \varphi(m_r) = 128$

Step i	m_{r-i+1}	ℓ_i	$\lvert B'_{r-i+1}\rvert$	$\lvert\tilde{C}'_i\rvert$	$\varphi(m_{r-i+1} \cdot \ell_i)$
1	256	17	2	2	2048
2	128	$221 = 17 \cdot 13$	4	4	12288
3	32	$1547 = 221 \cdot 7$	4	6	18432
4	8	$7735 = 1547 \cdot 5$	4	4	18432
5	2	$23205 = 7735 \cdot 3$	2	2	9216
6	1	23205			9216

Table 3: Concrete choices for $m_r = 128, \varphi(m_r) = 64$

Step i	m_{r-i+1}	ℓ_i	$\lvert B'_{r-i+1}\rvert$	$\lvert\tilde{C}'_i\rvert$	$\varphi(m_{r-i+1} \cdot \ell_i)$
1	128	17	2	2	1024
2	64	$221 = 17 \cdot 13$	4	4	6144
3	16	$1105 = 221 \cdot 5$	4	4	6144
4	4	$3315 = 1105 \cdot 3$	2	2	3072
5	2	$9945 = 3315 \cdot 3$	2	3	4608
6	1	9945			4608