TOWARDS PRACTICAL FULLY HOMOMORPHIC ENCRYPTION

A Thesis
Presented to
The Academic Faculty

by

Jacob Alperin-Sheriff

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Computer Science

Georgia Institute of Technology
December, 2015

TOWARDS PRACTICAL FULLY HOMOMORPHIC ENCRYPTION

Approved by:

Professor Chris Peikert, Advisor
Computer Science
Georgia Institute of Technology

Professor Zvika Brakerski
Department of Computer Science and
Applied Mathematics
Weizmann Institute of Science

Professor Sasha Boldyreva
Computer Science
Georgia Institute of Technology

Professor Matt Baker
School of Mathematics
Georgia Institute of Technology

Professor Richard Lipton
Computer Science
Georgia Institute of Technology

Date Approved: July 21, 2015

ACKNOWLEDGEMENTS

I want to thank my advisor Chris Peikert for being an invaluable resource, inspiration and collaborator. I would also like to thank Craig Gentry, for getting the ball rolling on FHE; a drawing of his "fully homomorphic box dragon" [45] that I made in early 2011 has been hanging next to me in the lab ever since as inspiration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Fully homomorphic encryption (FHE) allows for computation of arbitrary functions on encrypted data by a third party, while keeping the contents of the encrypted data secure. This area of research has exploded in recent years following Gentry's seminal work. However, the early realizations of FHE, while very interesting from a theoretical and proof-of-concept perspective, are unfortunately far too inefficient to provide any use in practice.

The *bootstrapping* step is the main bottleneck in current FHE schemes. This step refreshes the noise level present in the ciphertexts by homomorphically evaluating the scheme's decryption function over encryptions of the secret key. Bootstrapping is necessary in all known FHE schemes in order to allow an unlimited amount of computation, as without bootstrapping, the noise in the ciphertexts eventually grows to a point where decryption is no longer guaranteed to be correct.

In this work, we present two new bootstrapping algorithms for FHE schemes. The first works on packed ciphertexts, which encrypt many bits at a time, while the second works on unpacked ciphertexts, which encrypt a single bit at a time. Our algorithms lie at the heart of the fastest currently existing implementations of fully homomorphic encryption for packed ciphertexts and for single-bit encryptions, respectively, running hundreds of times as fast for practical parameters as the previous best implementations.

CHAPTER I

INTRODUCTION

1.1 Motivations for Fully Homomorphic Encryption

Cloud computing allows for the outsourcing of computation to an external shared pool of configurable computing resources. [70] This saves users from having to acquire hardware infrastructure, and instead lets them "pay to play," spending money on the service only when using it. In the last decade, it has become ubiquitous, with many large technology companies such as Amazon, Google, Oracle, and Microsoft entering the area, and many new companies forming to take advantage of it.

There are a number of security concerns for cloud computing beyond those present for in-house IT concerns. Some of these concerns, such as the need to securely transfer information to and from the cloud, can easily be resolved with the proper implementation of various practical well-known public and private-key cryptographic solutions. [6] However, once the data has made it to the cloud securely, one needs to find a way to perform the computation securely as well.

Allowing the cloud server the ability to decrypt the encrypted information sent to it is a potential solution. In fact, this is what currently exists in all implemented cloud computing solutions. The problem with this solution is that it requires the client to trust the cloud server implementation to protect their data from misuse. This level of trust has been an acceptable risk to many when it comes to less sensitive data. However, for more sensitive data (such as trade secrets), many companies have had the attitude that "my sensitive corporate data will never be in the cloud," [6] as they are unwilling to trust the cloud provider to properly secure and not misuse their data.

An alternative to the above less than optimal solution is for the cloud to be able to perform

the desired computations on the data while it remains encrypted. Under this paradigm, there is no need for any level of trust in the cloud server, as the data remains encrypted even in the cloud, and even while computations are being performed. This option is known as fully homomorphic encryption (FHE) [43].

Often referred to as "the holy grail of cryptography," FHE has numerous uses beyond the given example concerning outsourced computation. FHE can also be used to achieve secret program execution [43, 26] and private information retrieval [66, 43, 87]. More recently, it has been used as a building block to achieve various cryptographic primitives such as indistinguishability obfuscation (IO) for all circuits [42].

1.2 Early Attempts at FHE

Fully homomorphic encryption was first considered in the early days of public key cryptography by Rivest, Adleman and Dertouzos in 1978, under the term "privacy homomorphism" [76]. The application they considered was for time-sharing computing services, which was essentially analogous to today's cloud computing reality. While they gave several examples of potential privacy homomorphisms, they acknowledged that they might be cryptographically insecure. As it turned out, a number of years later, all the given example homomorphisms in the paper were indeed shown to be insecure [27].

While numerous applications were discussed in the literature, over the next 30 years, essentially no progress was made. Only a handful of candidate fully homomorphic encryption schemes were proposed [37, 36]. Of these handful, all were shown in a short period of time to be insecure [32, 86]. Then, in 2009, Craig Gentry shocked the cryptographic community with a candidate fully homomorphic encryption scheme based on plausible hardness assumptions [43]. Since then, the area has exploded, with many, many papers in the area.

1.3 An Introduction to Fully Homomorphic Encryption

The basic idea behind FHE is fairly simple. FHE allows for the computation of arbitrary functions on encrypted data by a third party, without the other party having access to the secret key. By arbitrary functions, we mean any function that can be expressed as a (polynomial-sized) *circuits* in terms of the input. This definition is chosen because these functions are (when expressed as circuits) those generally considered to be computationally tractable. As all circuits can be built using only NAND gates, it is easily seen that is sufficient (and necessary) to be able to evaluate a NAND gate over two encrypted bits.

1.3.1 Fully Versus Partially Homomorphic

In currently known fully homomorphic encryption schemes, there is no "natural" way to directly evaluate a NAND gate. Instead, the basic operations in these encryption schemes are multiplication and addition over a commutative ring, typically one that embeds $\mathbb{Z}_2$. To see that the ability to evaluate multiplication and addition in such a ring is sufficient for FHE, note that for $x_1, x_2 \in \{0, 1\}$,

$$x_1 \text{ NAND } x_2 = 1 - x_1 x_2.$$

If an encryption scheme allows only homomorphic multiplication, or only homomorphic addition, the scheme is referred to as partially homomorphic. Many public-key encryption schemes are naturally partially homomorphic, such as (unpadded) RSA and El-Gamal (multiplication), and Paillier, Goldwasser-Micali, and Benaloh (addition). [40] Partially homomorphic encryption is itself an interesting topic with several applications. In particular, partially homomorphic encryption has been used extensively in secure voting protocols. [61, 64, 34, 13]. Nevertheless, while useful, partially homomorphic encryption has significantly less applications than fully homomorphic encryption, and for the remainder of this thesis, we will be referring only to fully homomorphic encryption.

1.3.2 Security and Efficiency Requirements

In addition to the basic ability to evaluate circuits over encrypted messages, there are several other requirements that must be satisfied in order for a scheme to be fully homomorphic. First of all, the encryption scheme must still satisfy basic notions of cryptographic security (i.e. chosen-plaintext security). If not, the trivial "encryption" scheme that simply outputs the plaintext itself as the "encryption" would still satisfy the definition.

Another trivial solution that must be excluded is the solution that involves simply outputting the ciphertexts and the description of the circuit. To exclude this trivial solution, we require that the size of the output ciphertext be a fixed polynomial size in terms of the security parameter, independent of the size or depth of the circuit being evaluated. As a stronger requirement, we can follow Gentry and require that *decryption* of the ciphertext after the homomorphic evaluation be expressible by a circuit of fixed polynomial size in the security parameter (still independent of the complexity of the circuit being homomorphically evaluated). [43] Given that a major focus of this work is on expressing the decryption circuit as simply and efficiently as possible, this stronger requirement is particularly suitable for our purposes.

1.3.3 Noisy Ciphertexts

At a high level, all currently known FHE schemes have followed Craig Gentry's original paradigm. Gentry's most significant idea was to use a scheme with inherent additive and multiplicative homomorphisms. That is to say, to perform a homomorphic addition, one simply adds the ciphertexts together in a "natural manner", and to perform a homomorphic multiplication, one multiplies the ciphertexts together in a suitable "natural" manner. He found such a solution by using ideal lattices, which can be represented by elements of a polynomial ring, and as a result, have "natural" addition and multiplication.

The ciphertexts in Gentry's original scheme and all subsequent peer-reviewed schemes that are not currently broken have an associated noise level. Without this noise, they would

be insecure. This insecurity follows from the fact that if the noise were to be removed from any of these schemes, the decryption function would become linear. Concretely, there would be a function linear in the secret key that, when applied to ciphertexts, output the original messages. As a linear function, this decryption function would be learnable, which Kearns and Valiant have shown is inherently insecure [63, 19].

With the noise, the decryption functions of these schemes are no longer linear. Instead, computing the proper linear (in the secret key) function of the secret key and a ciphertext results in some form of "noisy" encoding of the message. As long as the noise remains below a certain threshold, we can recover the message. However, if the noise becomes too high, then we are no longer guaranteed to be able to recover the message.

For those schemes based on lattices, the hardness of breaking the underlying hard problems is proportional to the "noise level" in a ciphertext, where the "noise level" is the ratio of the absolute value of the noise in the ciphertext modulus we are working over to the absolute value of the noise in the ciphertext [75]. In particular, when this ratio becomes as small as 2^{-n} (where n is the dimension of the lattice used in the scheme), the underlying lattice problems can be efficiently solved with the LLL algorithm, leaving the schemes insecure.

1.3.4 Noise Growth and Homomorphic Operations

The importance of this trade-off between noise level and security becomes apparent when considering the homomorphic properties of the scheme. The noise level in these ciphertexts grows as a result of performing homomorphic operations. In currently known schemes, this noise growth limits the depth of computation that can be performed to an a priori bound. Such a scheme is known in the literature as SHE (somewhat homomorphic encryption) if the depth is relatively small (e.g. at most logarithmic in the security parameter) or as leveled FHE if the maximum depth of computation can be set a priori (via a suitable choice of initial parameters) to be any fixed polynomial in the security parameter.

The maximum depth of computation possible in these schemes is a direct consequence of the rate of noise growth for various homomorphic operations. Consequently, reducing this rate of growth has been a major focus of research. In Gentry's initial scheme, as well as in all subsequent schemes, performing a homomorphic addition causes the noise in the resulting ciphertext to be the sum of the noises in the two input ciphertexts.

Multiplication, however, has seen significant improvements in the rate of growth. In Gentry's initial candidate scheme, performing a homomorphic multiplication causes the noise in the resulting ciphertext to be the *product* of the noises in the two input ciphertexts. As a consequence, Gentry's initial scheme only allowed the evaluation of circuits of very limited depth (see Section 1.4 for further details). In more recent schemes [53, 20, 18], multiplication causes the noise level to grow by a multiplicative factor quasilinear in the security parameter λ. This turns out to be sufficient to give leveled FHE under plausible assumptions.

In order to accomodate noise growth while remaining secure, 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 [20] grow quasilinearly in both the supported multiplicative depth d and the security parameter λ, i.e., as $\tilde{O}(d \cdot \lambda)$.

1.4 Bootstrapping for FHE

To move beyond leveled FHE and enable evaluation of circuits of arbitrary depth, we need to use bootstrapping, Gentry's main innovation. Bootstrapping is a procedure that allows us to homomorphically refresh the noise level in the ciphertext.

1.4.1 Basics of Bootstrapping

To understand bootstrapping, one must first consider what happens when we decrypt a ciphertext in the clear. Viewing the decryption function for a given ciphertext as a known

function of the (unknown) secret key, we can see that evaluating the decryption function gives us the original message back. The crucial idea behind bootstrapping is noticing that since decryption can be represented as a function of the secret key, it can be evaluated homomorphically over an *encryption* of the secret key. Following this idea, we encrypt the secret key (under itself or its own public key) and put the resulting ciphertext (or ciphertexts) into a publicly available key that has been termed the evaluation key in the literature. Using this evaluation key, homomorphic evaluation of the decryption function, which we call bootstrapping, can be performed by a third party without knowledge of the secret key, simply by utilizing the homomorphic properties of the underlying scheme. Importantly, the encryption of the secret key creates a "circular security" situation; we discuss this in more detail in Section 1.6.1.

The result of this homomorphic evaluation of the decryption function is an *encryption* of the original message. Since we began with an encryption of the original message, it is initially unclear how this bootstrapping operation does anything useful. To understand its utility, we need to consider the noise levels. Since no homomorphic operations have been done on the encryption of the secret key, those ciphertext (or ciphertexts) have relatively low noise levels. As a result, if the bootstrapping procedure can be done with sufficiently small noise growth, then we can take a ciphertext with a very high noise level and bootstrap to obtain a ciphertext that encrypts the same message with less noise. This lower noise means we can perform at least one more homomorphic operation on the ciphertext before needing to bootstrap again. By repeating this process, we obtain unbounded fully homomorphic encryption.

As things currently stand, bootstrapping is the main bottleneck in FHE. For a concrete example, in Gentry's initial implementation of bootstrapping it took 30 minutes perform a single bootstrapping operation on a single CPU with an underlying lattice while achieving 72 bits of security under the best known lattice-reduction algorithms [47]. While still a big breakthrough, this scheme was ludicrously inefficient. Since then, there has been intensive

study of different forms of decryption procedures for SHE schemes, and their associated bootstrapping operations.

1.4.2 Survey of Bootstrapping Algorithms

There have been two primary methods by which the time required for bootstrapping has been improved. The first method has been through improving the underlying homomorphic encryption scheme. Improvements in this category have included reducing the growth rate of the noise from homomorphic multiplications, allowing "packed" ciphertexts (encrypting many bits at once), and giving FHE schemes provably secure under better-understood underlying assumptions [44, 53, 84, 47, 23, 20].

Our work focuses on the second method, which involves improving the algorithm for bootstrapping for a given underlying SHE scheme. Here we give a survey of previous works falling into this category.

Before discussing these methods, we recall a high-level form of the decryption function that holds for all lattice-based SHE schemes. The schemes are parameterized by a security parameter λ, R, q, and k, where R is a commutative ring and q and k are integers. In the general lattice setting $R = \mathbb{Z}$ and $k = n = \tilde{O}(\lambda)$ is quasilinear in the security parameter, while in the ideal lattice setting R is a cyclotomic ring of size $n = \tilde{O}(\lambda)$ and $k = 2$, while in both settings q is polynomial in the security parameter. Ciphertexts are vectors in R_q^k and secret keys are *short* vectors in R^k. Messages can be viewed as elements of R_p, where p is an integer, $p \ll q$; in some schemes, the actual message may lie in some subring R'_p of R_p. Decryption of a ciphertext $\mathbf{c} \in R_q^k$ can then be expressed as the following function of the secret key $\mathbf{s} \in R^k$:

$$\mathsf{Dec}_{\mathbf{c}}(\mathbf{s}) = \lfloor \langle \mathbf{c}, \mathbf{s} \rangle \rceil_p,$$

where $\lfloor w \rceil_p := \lfloor w \cdot (p/q) \rceil$ denotes rounding $w := \langle \mathbf{c}, \mathbf{s} \rangle \in R_q$ to the nearest multiple of $\frac{q}{p}$.[1]

[1] Although most early schemes used an lsb encoding rather than the msb encoding described above, there is

1.4.2.1 Gentry's Initial Algorithm

In Gentry's initial bootstrapping algorithm, all encryptions were of single bits, due to problems of overly high-depth circuits he encountered when considering decryption circuits for larger elements. Under these restrictions, evaluating the decryption circuit essentially reduced to adding up n integers, each represented in binary with $O(\log(n))$ bits of precision. To evaluate this, he used elementary symmetric polynomials (all of degree at most n) to compute the Hamming weight, represented in binary, of each *column* of bits (i.e. bits with the same place value) of the original n integers, ending up with $\log n$ integers represented in binary with $O(\log(n)$ bits of precision. As the polynomials evaluated in this step are of degree at most n, it can be computed in depth $\log n$. He then used the "3-for-2" trick, which allows one to replace 3 numbers represented in binary with 2 numbers having the same sum, and can be computed in constant depth. Applying this recursively to end up with 2 numbers requires depth $O(\log_{3/2}(\log n))$, and the final sum requires depth $O(\log \log n)$. This allows the entire bootstrapping algorithm to be evaluated depth $O(\log n)$ [43], which is easily seen to be within a constant factor of the optimal depth for fan-in two circuits.

However, in this initial scheme, the error growth rate from multiplication was too high to allow evaluation of this entire depth $O(\log n)$ circuit. In order to get around this issue and make the scheme bootstrappable, Gentry introduced the now deprecated "squashing" technique. This technique involves adding to the public key a set of vectors with a secret sparse subset summing to the secret key, which makes knowledge of this sparse subset sufficient to decrypt. This extra information allows the encrypter to form a modified ciphertext $\mathbf{c}'$ by computing the inner product of each of these vectors with the original ciphertext $\mathbf{c}$ and concatenating the results. To decrypt, we only need to sum as many integers as there are elements in the secret sparse subset. By making the subset sufficiently sparse (while staying large enough to ensure a superpolynomially large number of possible subsets),

---

an easy transformation from lsb (where p and q must be coprime for security) to msb; see Section 2.8. As the reverse transformation is not possible when p and q are not coprime, we have written it in the msb form.

the depth can be sufficiently reduced, allowing bootstrapping to proceed.

1.4.2.2 Subsequent Works

This need for squashing was first overcome by Gentry and Halevi. To do so, they expressed decryption as a depth-3 arithmetic circuit ($\sum\prod\sum$). To handle the multiplication step, they were able to switch to the multiplicatively homomorphic El-Gamal scheme after the initial addition without any homomorphic evaluation. By setting the parameters of the lattice-based SHE scheme to be sufficiently large, they were then able to handle decryption of the El-Gamal scheme and the final addition step without the need for squashing. However, these techniques did not improve on efficiency [46].

Soon after, the need for squashing was entirely obviated in a work by Brakerski and Vaikuntanathan [23]. This work introduced key-switching as well as dimension and modulus reduction. By using these techniques, the ciphertext can be shrunk down to a size small enough to allow the scheme to evaluate its decryption circuit homomorphically. These techniques are discussed in somewhat more detail in Sections 2.4 and 3.5.1.

There have been several other works focusing on bootstrapping since then. Two of these algorithms are the starting point for the works making up this thesis, and we discuss them in the next section [50, 51, 25]. The remainder of the works that have been published build on the preliminary versions of the works making up this thesis, and we defer discussion of these works to the beginning of the relevant chapters [60, 73, 59, 38].

1.5 Our Contributions

In this thesis, we present two bootstrapping algorithms, both of which were joint work with Chris Peikert. These algorithms significantly improve on the previous work in terms of practical efficiency. As a result, we bring fully homomorphic encryption much closer to practicality, although there is still a long way to go.

1.5.1 Efficient Bootstrapping-General

Chapter 2 contains our first bootstrapping algorithm. A preliminary version of this work appeared in CRYPTO 2013 [3].

In the earliest fully homomorphic encryption schemes [43, 44, 84, 23], encryption was limited to a bit (or more precisely, to an element mod p for a small integer p). Improvements by Smart and Vercauteren [82] as well as by Brakerski, Gentry and Vaikuntanathan [20] led to schemes working over "packed" ciphertexts, which encrypt large numbers of plaintext elements, achieving $\tilde{\Omega}(\lambda)$ encrypted bits per ciphertext. Moreover, when encrypting modulo qR for judicious choices of modulus q and cyclotomic ring R, these ciphertexts also allow the encryption of individual (scalar) plaintext elements in multiple "slots" via the Chinese Remainder Theorem (CRT). This then allows for batch (SIMD) homomorphic operations on the plaintexts in the various slots. These advances initially allowed for the construction of a scheme running bootstrapping in time $\tilde{O}(\lambda^2)$ with quasilinear overhead. Subsequently, Gentry, Halevi, and Smart [51, 50] showed how to bootstrap these "packed" ciphertexts in quasilinear $\tilde{O}(\lambda)$ runtime using a technique involving the composition of permutations in a log-depth permutation network.

However, while the result of Gentry, Halevi and Smart is a powerful theoretical result and is asymptotically within a polylogarithmic factor of being optimal, it is far from the end of the discussion in terms of practicality. Part of the reason for this is that the additional polylogarithmic factor is very very large. As a result, for practical values of the security parameter λ, the runtime appears to be longer than for the asymptotically worse $\tilde{O}(\lambda^2)$ of Brakerski, Gentry and Vaikuntanathan. Indeed, a recent implementation of parts of this scheme by Halevi and Shoup shows that the permutation/shift-networks upon which the quasilinear time bootstrapping result is based are not that efficient, with each individual permutation in the log-depth permutation network taking several seconds to compute [58].

We give a bootstrapping algorithm which improves significantly on that of Gentry et al [3]. The main observation made in [50] is that one can use the log-depth permutation

network to move the coefficients of an encrypted plaintext polynomial reduced modulo the cyclotomic ring $\Phi_m(X)$ into the individual plaintext "slots." The coefficients can then be rounded in polylogarithmic time in a SIMD fashion (using a result from [50]) and then mapped back to the coefficients of the plaintext.

Our work has the same high-level form of moving the plaintext coefficients into slots, rounding, and then moving from the slots back to the coefficients. However, we avoid many of the inefficiencies of the previous work such as the use of permutation networks or reductions modulo the full cyclotomic ring (the mth cyclotomic polynomial generally has a very ugly form for the values of m we need), thus getting a practically more efficient scheme.

Several implementations at least partially based on this work now exist. An implementation of FHE by Rohloff and Cousins performs bootstrapping based on our techniques for *non-packed* ciphertexts, achieving bootstrapping of ciphertexts encrypting a single bit with 64 of security in 275 seconds on 20 cores [77]. A more recent work by Halevi and Shoup, which uses the tensorial decomposition (but not ring-switching) central to our work, takes 320 seconds on a single core to bootstrap a ciphertext with 1024 slots of elements in GF(2^{16}), achieving an amortized rate of 51 bits/second [59].

1.5.2 Efficient Bootstrapping with Polynomial Error Growth

Chapter 3 contains our second bootstrapping algorithm, which focuses on achieving efficiency while keeping the noise growth to a level polynomial in the security parameters. A preliminary version of this work appeared in CRYPTO 2014 [4].

From a security perspective, the work described in the previous section suffers from a significant drawback: the scheme must be based on assuming the hardness of *superpolynomial* approximation of worst-case lattice problems on ideal lattices. This also indirectly affects the efficiency, as the size of the parameters required to achieve real-world security go up with the size of the approximation factor in the reduction.

Brakerski and Vaikuntanathan [25] took an important step by giving a method for bootstrapping a somewhat homomorphic encryption scheme that only incurs *polynomial* error growth in the security parameter λ. This immediately gives (leveled) FHE with security based on worst-case lattice problems with polynomial approximation factors, via the reductions to the learning with errors (LWE) problem given in [75, 74, 22] (to achive unbounded FHE, they require the standard additional circular security assumption). The starting point for their scheme is the homomorphic cryptosystem of Gentry, Sahai and Waters (GSW) [53], and the "quasi-additive" nature of its noise growth under homomorphic multiplication when using $\{0, 1\}$ messages, and more generally when working over permutation matrices. They then combine this with the "circuit sequentialization" property of Barrington's Theorem [10], which converts a depth-d circuit into a length-4^d "branching program," which can be realized by a fixed sequence of conditional multiplications of permutation matrices. Since the decryption circuit for the GSW scheme can be realized as a depth $c \log(\lambda)$ circuit for a constant c, Barrington's theorem gives a polynomial-length program which can be evaluated with only polynomial growth in the noise. On the downside, evaluating the branching program requires $O(\lambda^{2c})$ homomorphic multiplications, rendering it very inefficient.

We improve on this result by giving a bootstrapping method that requires only a quasi-linear $\tilde{O}(\lambda)$ of homomorphic operations on GSW ciphertexts. If we instantiate our boostrapping method with a GSW scheme based on ring-LWE, our algorithm requires $\tilde{O}(\lambda^2)$ *bit operations*. Moreover, letting $n = \tilde{O}(\lambda)$, our scheme can be based directly on lattice approximation factors of $\tilde{O}(n^3)$, which is not that much larger than the factors required for just plain public-key encryption.

The techniques in this work are hamstrung by the fact that the polynomial error growth depends on the messages being roots of unity instead of larger ring elements, preventing us from including messages of size proportional to the security parameter. However, it performs quite well in practice. In a recent work which optimizes our bootstrapping algorithm in the ring-setting, they are able to evaluate bootstrapping in less than a second (for an encryption

of a single bit) [38].

1.6 Open Problems

Here we discuss two major open problems in the area of bootstrapping and homomorphic cryptography.

1.6.1 Circular Security

If we wish to achieve only leveled FHE (providing a bootstrapping key for each "level"), then our bootstrapping algorithms are secure under the Ring-LWE and LWE assumptions, respectively, as one can use a hybrid argument to show IND-CPA security given the IND-CPA security of the underlying scheme. However, as we have mentioned, in order to achieve unbounded FHE with our bootstrapping algorithms, as well as to avoid absurdly large public key sizes, it is necessary to make an ad hoc assumption of "circular security."

1.6.1.1 Definitions

Classical definitions of security for encryption (such as semantic or IND-CPA security) [54] assume that the plaintext messages are chosen independently of the secret key. However, in bootstrapping, providing encryptions of the secret key under its own public key is an inherent requirement. Several works have shown separations between traditional notions of security and circular security [30, 79, 65], making it potentially risky to trust that the encryption schemes we are using are in fact circular secure unless we manage to find a proof of circular security.

Circular security is a special case of key-dependent message security. Key dependent message-security [14] has generally been defined with respect to a family of functions $\mathcal{F} \subset \{f : \mathcal{K}^\ell \to \mathcal{M}\}$, where $\mathcal{K}$ is the keyspace for secret keys, ℓ is the number of users in the system and $\mathcal{M}$ is the message space of the underlying encryption scheme (technically, $\mathcal{F}$ is a family of sets of functions parameterized by a security parameter λ). The adversary can query encryptions of arbitrary functions from this family, with the goal of distinguishing

between honest encryptions and encryptions of some fixed message value in $\mathcal{M}$ (e.g., 0). There is an important distinction between KDM security and traditional IND-CPA security. For traditional IND-CPA security, one can show via a generic hybrid argument that allowing only a single encryption query by the adversary is equivalent to allowing an unbounded number of encryption queries. However, due to the dependence of the messages on the secret key, this argument does not work for KDM security, and in general, security against a bounded number of queries does not imply security against an unbounded number of queries. The literature has also generally considered the case of multiple distinct users (and hence multiple secret keys), and has looked at *key cycles* (circular security), where (functions from a specific class of) key 1 is encrypted under key 2, key 2 under 3, . . ., key n under key 1, as well as *key-cliques* (clique security), where key i may be encrypted under key j for any i, j [21, 16, 5, 2].

1.6.1.2 Challenges

It appears at first glance that our task should be tractable. First, we only need to consider the case of providing encryptions of functions of the secret key for a single user under its own public key as part of the public evaluation key, rather than requiring broader forms of KDM security. Secondly, instead of needing to allow an unbounded number of encryptions, we only need sufficient encryptions of the proper functions to be able to bootstrap.

However, due to the homomorphic properties of the scheme, this is essentially just as powerful as being able to make an unbounded number of queries of *any* efficiently computable function of the secret keys. To see this, first note that in both schemes, the public evaluation key contains an encryption of the full secret key, meaning that given oracle decryption access to the ciphertexts in the public evaluation key, one can recover the full secret key. From an encryption of the secret key, one can then exploit the fact that it is an unbounded FHE scheme to homomorphically compute an encryption of any efficiently computable function of the secret key. Via proper re-randomization during the

procedure, it should be possible to make sure that the ciphertexts are distributed according to an instantiation of the real encryption scheme with sufficiently larger error terms than the actual ciphertexts in the public evaluation key.

As of yet, the most expressive results for KDM security for LWE and ring-LWE-based schemes is for a function space consisting of *a priori bounded* degree d polynomials, in a work by Brakerski and Vaikuntanathan. However, this is achieved at the cost of the ciphertexts becoming larger in dimension by a factor of d. As the only way that we know to reduce the dimension of ciphertexts itself requires an "encryption" of the secret key 2.4.6, this technique is entirely unusable for achieving security for unbounded degree functions [24].

In fact, there are some reasons to believe the task might not be possible. In particular, if the efficiency requirement on the function is relaxed, and the reduction is only given input/output access to the arbitrary challenge function, then there are *black-box impossibility* results known for achieving a proof of security [57]. As a result, it is perhaps not surprising that so far, a proof of security has eluded us.

1.6.1.3 Beyond Circular Security

Until very recently, achieving unbounded FHE without resorting to an ad hoc circular security assumption was probably the main outstanding theoretical problem relating to fully homomorphic encryption. However, a recent breakthrough work achieved a pure FHE construction from a generic leveled FHE scheme using indistinguishability obfuscation (IO) and puncturable pseudorandom functions (PRFs) without requiring any encryptions of the secret key under itself (circular security) [29]. Although candidate constructions for IO, following the initial work of Garg et al. [42], require pure FHE (which, in turn, prior to this work, required circular security) as a building block to achieve IO for *all* circuits, it appears that the amount of IO required to achieve their result can be based on other concrete hardness assumptions relating to cryptographic constructions of multilinear maps. However, the multilinear map hardness assumptions have not been well-studied and in

fact, have recently resulted in meaningful attacks, although none of the attacks have yet been "fatal." [48, 17, 31, 35] The construction also inherently requires the rather strong assumptions of subexponentially hard IO and subexponentially hard one way functions, rather than requiring only the standard polynomial hardness. Moreover, from a practical perspective, FHE based on this construction would be very very very inefficient and have very very large keys. As a result, this construction is far from the end-all in the area, and it is still very worthwhile to consider resolving the circular security issues in our bootstrapping constructions or their variants.

1.6.2 Bootstrappable Homomorphic Trapdoors

A dual to fully homomorphic encryption is the new primitive of fully homomorphic trapdoor functions (HTDFs). Here, each trapdoor function has an associated *tag* t, which corresponds to a message in FHE. Performing computations on the trapdoor functions induces operations on the underlying tags. Among other useful properties, the trapdoor remains functional as long as the tag t is invertible, while it becomes "punctured" (allowing the embedding of a challenge instance of a hard problem) when the tag t is 0. As an unwanted but heretofore necessary side effect, these computations also degrade the quality of the trapdoor in a manner corresponding to noise growth in FHE schemes. The primitive was implicitly introduced by Boneh et al. [15] in the form *key*-homomorphic encryption with the goal of applying this technique to construct attribute-based encryption (ABE) for arbitrary arithmetic circuits. It was then explicitly described by Wichs et al and used to create, as a first-class object, *leveled* fully homomorphic trapdoor functions HTDFs [55] in order to achieve *leveled* fully homomorphic signatures.

We have shown in an independent work that a slight extension of homomorphic trapdoor functions can lead to compact public key sizes for lattice-based signatures [1]. Still to be resolved is the question of bootstrapping these homomorphic trapdoor functions. At present, one can only homomorphically compute functions of an a priori bounded degree.

If we could *bootstrap* these trapdoor functions, we could then homomorphically compute functions of unbounded degree over the tags. While the form of these functions is similar to that of the GSW cryptosystem [53], there are a number of challenges to be overcome in order to adapt these trapdoor functions to make them bootstrappable.

1.7 Notation

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, and $(\mathbb{Z}_q, +)$ its additive group. For integers q, q', we define the integer "rounding" function $\lfloor \cdot \rceil_{q'} \colon \mathbb{Z}_q \to \mathbb{Z}_{q'}$ as $\lfloor x \rceil_{q'} = \lfloor (q'/q) \cdot x \rceil \bmod q'$. We write vectors $\mathbf{v}$ in bold lowercase and we write matrices $\mathbf{M}$ in bold uppercase. By $\mathrm{ord}_p n$ we mean the multiplicative order of n modulo p (i.e. the least k just that $n^k = 1 \bmod p$).

CHAPTER II

EFFICIENT BOOTSTRAPPING OF PACKED CIPHERTEXTS

2.1 Bootstrapping Efficiently

The past few years have seen an intensive study of different forms of decryption procedures for SHE schemes, and their associated bootstrapping operations [44, 43, 84, 47, 23, 46, 20, 51]. The first few bootstrapping methods had moderate polynomial runtimes in the security parameter λ, e.g., $\tilde{O}(\lambda^4)$. Brakerski, Gentry, and Vaikuntanathan [20] 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 [50] 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 [51], 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 [51, 50] 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 [20]. The second reason is that their

bootstrapping operation is algorithmically very complex and difficult to implement (see the next paragraphs for details). Indeed, perhaps due to this difficulty, no implementations the only implementation of bootstrapping with subquadratic runtime that we are aware of is based of a

Is quasilinear efficient? The complexity and large practical overhead of the constructions in [51, 50] arise from two kinds of operations. First, the main technique from [51] 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 [23, 20] 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 [50] 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 [51]. However, as the authors of [50] 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 [50]. However, even a very efficient decryption circuit would still incur the large polylogarithmic overhead factors from the techniques of [51].

2.2 Overview of Our Algorithms

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 has proven to be substantially more efficient in practice than all prior methods. [59] 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 (see, i.e. This improves upon the method of [20], 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. However, for practical parameters (although not asymptotically) it has now been superseded efficiency-wise by the extension of our algorithm in Chapter 3 by Ducas and Micciancio [38].

For packed ciphertexts, our procedure draws upon high-level ideas from [51, 50], but our approach is conceptually and technically very different. Most importantly, it completely avoids the two main inefficiencies from those works: first, unlike [51], 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 [50].

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., [82, 20, 50]), 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 [8] 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.

2.2.1 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 [50] and Appendix 2.9) 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 [49], but unfortunately, this requires the ring-LWE problem [68] 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 [49], 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 [49]. 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 [69]. 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 [50].[1]

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.

2.3 Algebraic Number Theory 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.

[1] The use of more structured representations of cyclotomic rings in [69] 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!

2.3.1 Cyclotomic Number Fields

For a positive integer m, called the *index*, let ζ_m denote a primitive mth root of unity. The mth *cyclotomic* number field is the field extension $K = Q(\zeta_m)$. 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 over all the *primitive* complex mth roots of unity. We have by definition that K is a field extension of degree $n = \varphi(m)$ over $\mathbb{Q}$, so that the first and second cyclotomic number fields are simple $\mathbb{Q}$.

2.3.2 Canonical Embedding

In this section we briefly describe the *canonical embedding* of a cyclotomic number field, which endow the number field with a 'canonical' geometry.

As a Galois extension, the mth cyclotomic number field $K = \mathbb{Q}(\zeta_m)$ is of degree n, and therefore has exactly n distinct ring homorphisms (embeddings) $\sigma_i : K \to \mathbb{C}$ that fix every element of the base field $\mathbb{Q}$. Letting $\omega_m^i \in \mathbb{C}$ be some fixed primitive mth root of unity, these embeddings can be defined for each $i \in \mathbb{Z}_m^*$ by $\sigma_i(\zeta_m) = \omega_m^i$. The *canonical embedding* $\sigma : K \to \mathbb{C}^{n=\phi(m)}$ can then be defined as

$$\sigma(a) = (\sigma_i(a))_{i \in \mathbb{Z}_m^*}.$$

We recall several important facts about the canonical embedding (for further details, see [67, 69]). Most importantly, it gives K a canonical geometry. We define the ℓ_2 norm for $a \in K$ to be $\|a\|_2 = \|\sigma(a)\|_2$, and the ℓ_∞ norm to be $\max_i|\sigma_i(a)|$. Multiplication in the embedding is component-wise, so that for $a, b \in K$ (letting $\|\cdot\|$ denote an arbitrary ℓ_p norm, we have that

$$\|a \cdot b\| \leq \|a\|_\infty \cdot \|b\|,$$

giving a bound on how much elements expand other elements via multiplication. This bound is critical in the context of FHE.

2.3.3 Ring of Integers and Its Ideals

In this work, we shall mainly be concerned with the *ring of integers* $\mathcal{O}_m = \mathbb{Z}[\zeta_m]$ of the mth cyclotomic number field, which we refer to as the mth *cyclotomic ring*. As a result, $\mathcal{O}_m$ 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 2.3.7 we consider other, more structured bases that are essential to our techniques.

For a ring $R \subset K$ of algebraic integers, a *fractional ideal* $\mathcal{I} \subseteq K$ of R is a nontrivial R-module, such that there exists an element $d \in R$ making $d\mathcal{I} \subseteq R$ into an (integral) ideal of R. As a result, all (integral) ideals can be seen to be fractional ideals by taking $d = 1$. A fractional ideal is *principal* if it is generated by a single element. In other words, there exists $u \in K$ such that $\mathcal{I} = uR$. Importantly, any fractional ideal $\mathcal{I}$ embeds under σ as a lattice $\sigma(\mathcal{I})$, which we refer to as an ideal lattice.

The sum $\mathcal{I} + \mathcal{J}$ of ideals is the set of all $a + b$ for $a \in \mathcal{I}, b \in \mathcal{J}$, while the product ideal $\mathcal{I}\mathcal{J}$ is the set of all *finite sums* of terms ab for $a \in \mathcal{I}, b \in \mathcal{J}$. As in any Dedekind domain, the set of fractional ideals of R form a group under multiplication, so that every fractional ideal $\mathcal{I}$ has a multiplicative inverse $\mathcal{I}^{-1}$.

Two ideals $\mathcal{I}, \mathcal{J} \subseteq R$ are *coprime* if $\mathcal{I} + \mathcal{J} = R$. An ideal $\mathfrak{p} \subsetneq R$ is *prime* if whenever $ab \in \mathfrak{p}$ for some $a, b \in R$ then $a \in \mathfrak{p}$ or $b \in \mathfrak{p}$ (or both). Since R is a Dedekind domain, every ideal $\mathcal{I}$ factors uniquely up to ordering as a product of powers of prime ideals [41].

2.3.4 Duality and the Trace Function

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 $\mathrm{Tr} = \mathrm{Tr}_{\mathcal{O}_{m'}/\mathcal{O}_m} \colon \mathcal{O}_{m'} \to \mathcal{O}_m$ 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 $\mathrm{Tr}_{\mathcal{O}_{m''}/\mathcal{O}_m} = \mathrm{Tr}_{\mathcal{O}_{m'}/\mathcal{O}_m} \circ \mathrm{Tr}_{\mathcal{O}_{m''}/\mathcal{O}_{m'}}$.

For any fractional ideal $\mathcal{I} \in K$, its *dual* is defined as

$$\mathcal{I}^\vee = \{a \in K : \mathrm{Tr}(a\mathcal{I}) \subseteq \mathbb{Z}\}$$

Important facts about the dual ideal are that $(\mathcal{I}^\vee)^\vee = \mathcal{I}$, that $\mathcal{I}^\vee$ is itself a fractional ideal, and that it embeds under σ. Additionally, we have that for any fractional ideal $\mathcal{I}$, its dual is $\mathcal{I}^\vee = \mathcal{I}^{-1} \cdot \mathcal{R}^\vee$. The factor $\mathcal{R}^\vee$ is generally called the *codifferent* in the literature.

To characterize the codifferent, we consider an important element in the mth cyclotomic ring, defined as

$$g := \prod_{\text{odd prime } p|m} (1 - \zeta_p) \in \mathcal{O}_m. \tag{2.3.1}$$

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., [69, 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 2.3.1. *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 $R^\vee$; see [69, 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 [49, 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 $i = 1, \ldots, r$. $\square$

2.3.5 Tensorial Decomposition of Cyclotomics

An important fact from algebraic number theory, used centrally in this work (and in [69]), 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)

$$\mathcal{O}_l/\mathcal{O}_g \cong (\mathcal{O}_{m_1}/\mathcal{O}_g) \otimes (\mathcal{O}_{m_2}/\mathcal{O}_g). \tag{2.3.2}$$

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 2.3.2. *Let m_1, m_2 be positive integers and $g = \gcd(m_1, m_2)$, $l = \operatorname{lcm}(m_1, m_2)$. 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. □

2.3.6 Ideal Factorization and Plaintext Slots

Here we recall the details of unique factorization of prime integers into prime ideals in cyclotomic rings, and, following [82], 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 [50], 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.

2.3.7 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 $\mathcal{O}_{m''}$.[2] 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 [69], 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 "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, [69] 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

[2] 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.

basis. Unlike the powerful basis, the decoding basis has optimal noise tolerance (see [69, 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 2.4) 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 [69, 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).

2.4 Ring-Based Homomorphic Cryptosystem

Here we recall a symmetric-key somewhat-homomorphic encryption scheme whose security is based on the ring-LWE problem [68] in arbitrary cyclotomic rings. For further details on its security guarantees, various homomorphic properties, and efficient implementation, see [68, 24, 20, 52, 49, 69].

2.4.1 The Ring-LWE Problem

We briefly recall the ring-LWE problem, and in particular its discretized version, as well as the Gaussian distributions used in defining the problem. The problem involves distinguishing between samples drawn from the ring-LWE distribution (for a secret $s \in R_q^\vee$ chosen uniformly at random) and samples chosen uniformly at random. Concretely, for a distribution ψ, a sample from the discretized ring-LWE (often called R-DLWE in the literature) distribution $A_{s,\psi}$ over $R_q \times qR^\vee$ is generated by choosing $a \in R_q$ uniformly at random, sampling $e \leftarrow \psi$, and outputting $(a, b = a \cdot s + e \bmod qR^\vee)$, where $R = \mathcal{O}_m$ is a cyclotomic ring.

To ease error growth management and implementation, it is most convenient to set ψ equal to a properly discretized version of the (spherical) continuous Gaussian distribution D_s, which is a normalized version of the standard Gaussian function $\rho_s(\mathbf{x}) = \exp(-\pi\langle \mathbf{x}, \mathbf{x}\rangle/s^2)$.

2.4.2 Cryptosystem Definition

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. 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 [49], 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.

Let ψ be a continuous LWE error distribution over $K_{\mathbb{R}}$, and let $\lfloor \cdot \rceil$ denote a suitable discretization to (cosets of) $R^\vee$ of a properly chosen Gaussian distribution D_r (these discretizations can be achieved via various rounding techniques. For details, see [69]). The scheme can then be formally defined as follows.

- Gen: Sample secret key $\hat{s}' \leftarrow \lfloor \psi \rceil_{R'^\vee}$ and output $s' = (\hat{m}'/g')\hat{s}' \in R'$ as the secret key.

- $\mathsf{Enc}_{s'}(\mu \in R_p)$: Choose error term $e' \leftarrow (\hat{m}'/g')\lfloor \psi \rceil_{R'^\vee} \in R'_q$. Sample $c'_1 \leftarrow R_q$ uniformly at random. Let $c'_0 = -c'_1 \cdot s' + e' + \frac{q}{p} \cdot mu$, and output the ciphertext $(c'_0, c'_1) \in (R'_q \times R'_q)$.

- $\mathsf{Dec}_s(c' = (c'_0, c'_1))$: Compute $v := c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \pmod{qR'}$. Then output $\mu := \lfloor v \rceil_{\frac{q}{p}R}$.

Importantly for this work, we have that 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

$$c'_0 + c'_1 \cdot s' = \frac{q}{p} \cdot \mu + e' \pmod{qR'} \tag{2.4.1}$$

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 (2.3.1). Quantitatively, "short" is defined with respect to the *canonical embedding* of R'. The above system is equivalent to the one from [69] 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 [69] 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 [68, 69] for further details. 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 (2.4.1) 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. When p divides q, we can use homomorphic operations for the msb encoding due to Brakerski [18]; 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" [52]) that may not be possible for the msb operations (at least when p divides q).

2.4.3 Decryption Algorithm

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. In our formulation, the basis B' is $(\hat{m}'/g')$ times the decoding basis of $(R')^\vee$. See Section 2.3.7 and the previous section. 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).

2.4.4 Homomorphic Operations

To ease presentation, we say that a ciphertext of degree $d \geq 1$ is a polynomial $c = c(S)$ of degree at most d in an indeterminate S, having coefficients in R_q^d. Fresh ciphertexts produced by the encryption algorithm have degree 1, and can be seen as $c(S) = c_0 + c_1 S$.

Viewing the ciphertexts in this manner, it becomes very easy to define homomorphic operations. Specifically, for ciphertexts c, c' of arbitrary degrees k, k' (respectively), their homomorphic product is the degree-$(k + k')$ ciphertext $c(S) \boxdot c'(S) = c(S) \cdot c'(S)$, where $\cdot$ represents standard polynomial multiplication. The noise in the result is the product of the noise terms of c and c'. One can control the noise growth from multiplication by modulus-switching appropriately (the second operation below in Section 2.4.5).

For ciphertexts c, c' of *equal* degree k, their homomorphic sum is defined as the degree-k ciphertext $c(S) \boxplus c'(S) = c(S) + c'(S)$ (polynomial addition). The noise in the result is

the sum of the noise terms of c and c'. In order to homomorphically add two ciphertexts of different degrees, we can either homomorphically multiply the one with the smaller degree by a fixed public encryption of 1 a sufficient number of times, or we can use the key-switching technique (Section 2.4.6) to reduce the degree of both polynomials to 1.

2.4.5 Changing the Plaintext Modulus

We use two operations on ciphertexts that alter the plaintext modulus p and encrypted message $\mu \in R_p$, often referred to in the literature as "modulus switching." 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 $q/p' = d(q/p)$.

2.4.6 Key Switching

An important transformation introduced by Brakerski and Vaikuntanathan [23] allows us to go from a degree-d ciphertext $c(S)$ (with coefficient vector $\vec{c} \in R_q^{d+1}$) encrypting a message

μ under secret key s to a degree-1 ciphertext c' encrypted the same message μ under the secret key t. This transformation is referred to as "key-switching," and is used both for reducing the degree of a ciphertext and as a crucial part of "ring-switching," described below.

To enable key-switching from a degree-d ciphertext $c(S)$ to a degree-1 ciphertext, a key-switching hint is added to the evaluation key. Let $\ell = \lceil \log_2 q \rceil$ and define

$$\mathbf{g} = (1, 2, 4, \ldots, 2^{\ell-1}) \in \mathbb{Z}_q^\ell \quad \text{and} \quad G = I_{[d]} \otimes \mathbf{g}^t \in \mathbb{Z}_q^{d \times d\ell}.$$

G has a very short basis [71], so we can sample a short $\vec{x}$ such that $G\vec{x} = \vec{c}$. Let $\vec{s} = (s^0, \ldots, s^d) \in R^{d+1}$. The key-switching hint will essentially consist of an encryption of $G^t\vec{s}$ (in lsb form) under the target secret key t. To key-switch, we compute the inner product of this encryption with $\vec{x}$, which gives us the required result. The noise incurred as a result of this transformation is a relatively small *additive* factor. For details, see [69].

2.4.7 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 2.4.1 ([49], 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 2.4.1 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'. Next, it multiplies the resulting ciphertext by an appropriate fixed element

of R' (which is determined solely by the function L being evaluated). Finally, it applies the trace function $\mathrm{Tr}_{R'/S'}: R' \to S'$ to the ciphertext. 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 [49] for further details.

2.5 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 2.4, 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 2.7.1.3), or by effectively decreasing m slightly (at a possible improvement in practical efficiency; see Section 2.5.2).

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 (2.3.1), $q' = (\hat{m}'/\hat{m})q$, and $u' = v' \pmod{qR'}$. This proceeds in the following sub-steps (see Section 2.5.1 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 2.4.5, 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 2.5.2 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 $(\hat{m}/\hat{m}')\,\mathrm{Tr}_{R'/R}$ to the encryption of $(g'/g)u' \in R'_{q'}$, to obtain an encryption of plaintext

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

for some suitably small error term $e \in R$. See Section 2.6 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 2.7 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 $\lfloor \cdot \rceil : \mathbb{Z}_q \to \mathbb{Z}_p$ 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.

2.5.1 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.

2.5.2 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 2.10 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 2.3.7.) 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.

2.6 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 (2.3.1), 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 $R'' = \mathcal{O}_{m''} \supseteq R'$.

2.6.1 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 2.3.1, 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 2.4.5), 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., [49, Corollary 2.2]. In fact, by basic properties of the decoding/decryption basis (as

defined in [69]) 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.

2.6.2 Applying the Trace

Now we show how to efficiently homomorphically apply the scaled trace function $(\hat{m}/\hat{m}') \operatorname{Tr}_{R'/R}$ 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 2.3.1), making the scaling a well-defined operation that results in an element of R_q.

First recall that $\operatorname{Tr}_{R'/R}$ 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 $(\hat{m}/\hat{m}') \operatorname{Tr}_{R'/R}$ is the composition of the scaled trace functions $(\hat{m}_{i-1}/\hat{m}_i) \operatorname{Tr}_{R^{(i)}/R^{(i-1)}}$, 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 (2.3.1). 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 2.6.2.2 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.

2.6.2.1 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 [23, 20] 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 2.4.5, 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., [68, 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''.[3]

[3] 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 2.6.2.2 incurs no multiplicative increase in the noise rate.

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.

2.6.2.2 Applying the Trace via Ring-Switching

Here we describe the alternative procedure for applying the scaled trace, which uses the ring-switching technique from [49] (see Proposition 2.4.1). 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 2.4.1, 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.

2.7 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 [50], but instantiate it with very different technical components. Recall from Section 2.4.3 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 $\mathbb{Z}_p$).

2.7.1 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 $\sum_j \lfloor u_j \rceil_p \cdot c_j \in S_p$.

3. Invert the map from the first step to obtain $\lfloor u \rceil_p = \sum_j \lfloor u_j \rceil_p \cdot b_j \in R_p$.

Using batch/SIMD operations [82], 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 2.3.2 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 2.4.1 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.

2.7.1.1 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

$$A_{i-1} = A_{i-1}^{\text{out}} \cdot Z_i, \quad A_i = A_i^{\text{in}} \cdot Z_i \tag{2.7.1}$$

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 $1 \leq i < r$, 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

$$\bar{L}_i(A_{i-1}) = \bar{L}_i(A_{i-1}^{\text{out}} \cdot Z_i) = \bar{L}_i(A_{i-1}^{\text{out}}) \cdot Z_i = A_i^{\text{in}} \cdot Z_i \tag{2.7.2}$$

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

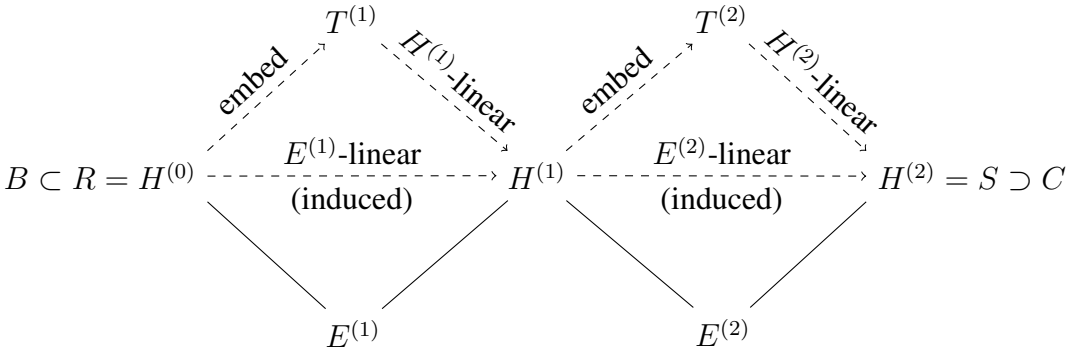

Figure 1: An example mapping from $B \subset R$ to $C \subset S$, via a sequence of hybrid rings. Each $E^{(i)} = H^{(i-1)} \cap H^{(i)}$ is a largest common subring, and each $T^{(i)} = H^{(i-1)} + H^{(i)}$ is a compositum, of adjacent hybrid rings. For any $E^{(i)}$-linear function from $H^{(i-1)}$ 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 2.3.2).

Summarizing the above discusion, we have the following theorem.

Theorem 2.7.1. *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.*

2.7.1.2 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 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 2.5), 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 $\mathrm{lcm}(u, u') \approx u$.

2.7.1.3 *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 2.5). 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.

2.7.2 Construction

By Theorem 2.7.1 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 (2.7.1).

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

2.7.2.1 *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

$$B_i = B'_i \cdot B_{i-1} \subset R^{(i)} \tag{2.7.3}$$

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 2.3.7, 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 [69] for details.)

2.7.2.2 *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 2.3.6 and 2.3.7, 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

$$C_i = C'_i \cdot C_{i-1} \subset \tilde{C}_i. \tag{2.7.4}$$

2.7.2.3 *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

$$H^{(i)} = \mathcal{O}_{m_{r-i}\,\ell_i} \cong R^{(r-i)} \otimes S^{(i)}, \tag{2.7.5}$$

$$A_i = B_{r-i} \cdot C_i \subset H^{(i)}, \tag{2.7.6}$$

where the tensor product in Equation (2.7.5) 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

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

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 2.7.2. *The sets A_{i-1}, A_i factor as in Equation* (2.7.1)*, i.e., $A_{i-1} = A_{i-1}^{out} \cdot Z_i$ and $A_i = A_i^{in} \cdot Z_i$ for some sets $Z_i \subset E^{(i)}$ and A_{i-1}^{out}, A_i^{in} 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 (2.7.3) 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 (2.7.4) 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 $|A_{i-1}^{\text{out}}| = |B'_{r-i+1}| = |C'_i| = |A_i^{\text{in}}|$ by design of C'_i. □

2.8 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.

2.9 Integer Rounding Procedure

Here we recall (a close variant of) the efficient arithmetic procedure from [50] 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 [50], 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 2.4, as explained in Section 2.4.5. The procedure also easily generalizes to any prime base.

Algorithm 1 Arithmetic procedure for computing $\mathrm{msb}_q \colon \mathbb{Z}_q \to \mathbb{Z}_2$ [50]

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 [50, 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.

2.10 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. Note that when $m_{r-i+1} = 2$, then the ring R_{r-i+1} has dimension 1, and so we can move directly from $m_{r-i+1} = 2$ to $m_{r-i+1} = 1$. In the tables below, and following the notation in Section 2.7:

- 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 $S^{(i)}/S^{(i-1)}$.

- r denotes the number of hybrid rings R_{r-i+1}, $i \in [r]$

All the indices are *lower bounds* needed to support the *functionality* of the ring-rounding technique on the plaintext space (Section 2.7). Larger ciphertext indices may be required to ensure adequate security for all the homomorphic operations; see Section 2.7.1.2.

Table 1: Concrete choices for $m_r = 1024, \varphi(m_r) = 512$

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	1024	17	2	2	8192
2	512	$221 = 17 \cdot 13$	4	4	49152
3	128	$1547 = 221 \cdot 7$	4	6	73728
4	32	$7735 = 1547 \cdot 5$	4	4	73728
5	8	$23205 = 7735 \cdot 3$	2	2	36864
6	4	$69615 = 23205 \cdot 3$	2	3	55296
7	1	69615			55296

Table 2: 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	17	2	2	4096
2	256	$221 = 17 \cdot 13$	4	4	24576
3	64	$1547 = 221 \cdot 7$	4	6	36864
4	16	$7735 = 1547 \cdot 5$	4	4	36864
5	4	$23205 = 7735 \cdot 3$	2	2	18432
6	1	23205			18432

Table 3: 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	$1105 = 221 \cdot 5$	4	4	12288
4	8	$3315 = 1105 \cdot 3$	2	2	6144
5	4	$9945 = 3315 \cdot 3$	2	3	9216
6	1	9945			9216

Table 4: 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	2048
2	64	$119 = 17 \cdot 7$	2	2	3072
3	32	$595 = 119 \cdot 5$	4	4	6144
4	8	$1785 = 595 \cdot 3$	2	2	3072
5	4	$5355 = 1785 \cdot 3$	2	3	4608
6	1	5355			4608

2.11 Bounds on the Compositum

Here we discuss asymptotic bounds for the maximum value obtained by the compositum. For simplicity, we focus on the case that the modulus $q = 2^j$ and the index of the initial ring R is $m_r = 2^{k+1}$, making the *dimension* of the initial ring R equal to the security parameter $\lambda = 2^k$. Initially, we hoped to be able to choose the cyclotomic tower of rings $S^{(i)}$ so that the compositum would be larger than the initial ring R by at most a polylogarithmic factor, but this does not appear possible. We say "appear" because a proof of the impossibility appears beyond our grasp, and indeed, the grasp of number theorists at present.

On the positive side, we are able to show that, for the conditions on the modulus and the index of the initial ring given above, there exist choices for the tower of cyclotomics $S^{(i)}$ such that the compositum at any point in the procedure is at most $2^{k+O(\sqrt{k}\ln(k))}$. We

also give a heuristic argument that choices exist for the tower of cyclotomics such that the compositum at any point in the procedure is at most $2^{k+O(\ln(k)\ln(\ln(k)))}$.

2.11.1 Factors Contributing to the Compositum

Here we discuss the approach used to choose indices for the intermediate "hybrid" rings in our procedure such that the maximum value taken by the compositum at any point in our procedure is minimized. Concrete choices for these indices can be seen in Section 2.10.

During step i of our algorithm, we embed the hybrid ring $H^{(i-1)}$ into the compositum $T^{(i)}$ of it and the adjacent hybrid ring $H^{(i)}$. The dimension of the compositum will be the least common multiple of the dimension of $H^{(i-1)}$ and $H^{(i)}$. To minimize the growth of the compositum, we wish to use as fine-grained rings as possible. In particular, the index of the compositum $T^{(i)}$ at step i divided by the index of the hybrid ring $H^{(i-1)}$ should always be some prime $p = p_i$. The number of slots provided by this prime will be equal to the "relative splitting number" of 2 in the extension $S^{(i)}/S^{(i-1)}$. This, in turn, is equal to the order of 2 modulo ℓ_i divided by the order of 2 modulo ℓ_{i-1}, where ℓ_i is the index of the destination ring S_i at step i.

At the end of the step, we ring-switch down from $T^{(i)}$ to $H^{(i)}$, and the compositum's dimension is divided by the dimension of the intermediate ring extension B'_{r-i+1}, which, to maximize efficiency, is set equal to the largest (not necessarily prime) factor of m_{r-i+1} less than the relative splitting number $\tilde{C}'_i$. As a result, the growth of the compositum during the procedure is due to the size of the primes p_i used, the growth of the order of 2 mod ℓ_i versus ℓ_{i-1} and due to the "wasted slots." These "wasted slots" come into play due to the relative splitting number not being a power of 2. In this case, since m_{r-i+1} is always a power of 2 for the case being considered, $|\tilde{C}'_i|$ is strictly greater than the maximum possible value of $|B'_{r-i+1}|$, and the additional slots in $S^{(i)}/S^{(i-1)}$ are considered "wasted." However, the multiplicative factor for the slots being wasted in each step (except for the last step, where it can be upper bounded by $\varphi(p_i)$) can be upper bounded by 2, since if $|\tilde{C}'_i| \geq 2|B'_{r-i+1}|$, we

can increase $|B'_{r-i+1}|$ by a factor of 2.

Recall that λ is the dimension of the initial ring R. Using the above information, for all i in the process except for the last step, we can upper bound the size of the compositum at step i as

$$\varphi(m_{r-i+1}\ell_i) \leq \lambda \text{ord}_{\ell_i} 2 \cdot 2^i$$

At the final step r, we can thus upper bound it with

$$\varphi(m_1\ell_r) \leq \lambda \text{ord}_{\ell_r} 2 \cdot 2^{r-1}\varphi(p_r - 1) \tag{2.11.1}$$

2.11.2 Overview and Difficulties of Asymptotic Bounds

In order to minimize the maximum dimension taken by the compositum, we want to find small relatively prime integers (preferably themselves prime) such that the order of 2 modulo their product is also small. For the unconditional upper bound, we begin by noticing that the *algebraic* decomposition of $x^n - 1$ into irreducible cyclotomic polynomial factors guarantees that $2^n - 1$ will have a large number of relatively small factors when n is itself a product of many small distinct prime factors. This observation leads to our approach for a concrete upper bound. We choose an integer n of the appropriate size that has many small distinct prime factors. We set $\ell_i = \Phi_{p_i}(2)$ for the ith largest prime factor p_i of n, where $\Phi_{p_i}(2)$ is the p_ith cyclotomic polynomial evaluated at 2. While we could give a slightly smaller upper bound on the size of the compositum by using the dth cyclotomic polynomials evaluated at 2 for *all* small divisors d of n, we have omitted this for ease of exposition.

Difficulties in making more significant improvements on the unconditional upper bound derived using the above approach stem from the lack of progress made in determining the distribution of factors of $2^n - 1$ beyond those that arise from the algebraic factorization into cyclotomic polynomials. For example, when p is prime, $\Phi_p(2) = 2^p - 1$ is a Mersenne number, and despite very intensive research, it is still unknown whether or not there are an infinite number of Mersenne primes prime (or, for that matter, whether there are an infinite number of $2^p - 1$ for prime p that are composite). [56]

It also seems intractable to show reasonably tight lower bounds. In particular, it seems out of reach to show that for sufficiently large k (where the initial ring $R = \mathcal{O}[2^{k+1}]$, there are no choices for the tower of rings $S^{(i)}$ such that the compositum is at most $2^{k+O(\log k)}$. It is not difficult to see that this would imply that asymptotically, the largest prime factor of $2^n - 1$ (n not necessarily prime) must be superpolynomially large in n, since if not, we could simply construct the tower of rings to have ratios of indices equal to the various prime factors of $2^n - 1$. However, while Murata and Pomerance conjecture that the largest prime factor may always be at least $2^{n/\log n}$ asymptotically [72], the best result known, even under strong number theoretic conjectures such as the Generalized Riemann Hypothesis and the abc conjecture, is that for sufficiently large n, the largest prime factor will always be at least $n^{2-\epsilon}$ for any $\epsilon > 0$. [83]

Instead, we give a significantly smaller heuristic upper bound that appears empirically to be relatively tight. Similar to our approach for the unconditional bound, we begin by choosing an *even* integer n that is the product of many small distinct prime factors and therefore has many even divisors. By Fermat's Theorem, if p is a prime such that $p - 1 \mid n$, then $p \mid 2^n - 1$, so that (under a heuristic argument), $2^n - 1$ will have many small factors.

2.11.3 Unconditional Upper Bounds

We will need the following bounds on functions of prime numbers. Let p_n denote the nth prime number.

Lemma 2.11.1 ([9, 39]). *For $n \geq 6$,*

$$n(\ln(n) + \ln(\ln(n)) - 1) < p_n < n(\ln(n) + \ln(\ln(n)))$$

Lemma 2.11.2 (Theorem 4, [9]). *Let $\vartheta(x) = \sum_{p \leq x} \ln p$ be the first Chebyshev function. Then for $x > 1$, we have that*

$$\vartheta(x) < x(1 + \frac{1}{2\ln(x)})$$

Combining the previous two lemmas gives the following.

Corollary 2.11.3. *For $n \geq 3$, $\vartheta(p_n) \leq n(\ln(n) + \ln(\ln(n)) + \frac{1}{2}$.*

We will also need the following lemmata about Mersenne numbers $M_p = 2^p - 1$ for prime p.

Lemma 2.11.4 ([78], Theorem 7.12). *If p is an odd prime, then any divisor of M_p is of the form $2kp + 1$ for some positive integer k.*

Corollary 2.11.5. *For an odd prime p, we have that $\varphi(M_p)/M_p \geq (1/2)^{1/\log_2(p)}$ where φ is Euler's totient function*

Proof. By Lemma 2.11.4, the smallest prime factor of M_p is at least $2p + 1$. Thus M_p has at most $p/\log_2(p)$ distinct prime factors, and so

$$\frac{\varphi(M_p)}{M_p} \geq \left(\frac{2p}{2p+1}\right)^{p/\log_2(p)} \geq \left(\tfrac{1}{2}\right)^{1/\log_2(p)}.$$ □

Lemma 2.11.6. *For positive integers a, b, x, $\gcd(x^a - 1, x^b - 1) = x^{\gcd(a,b)} - 1$*

Proof. Clearly $x^{\gcd(a,b)} - 1$ divides both $x^a - 1$ and $x^b - 1$. Now, let $g := \gcd(x^a - 1, x^b - 1)$. $x^a = x^b = 1 \pmod{g}$ so that by the extended Euclidean algorithm, $x^{\gcd(a,b)} = 1 \pmod{g}$. Thus, $g \mid (x^{\gcd(a,b)} - 1)$, and since g is the greatest common divisor, equality holds. □

We then immediately have the following corollary.

Corollary 2.11.7. *For coprime a, b, we have that $\gcd(M_a, M_b) = 1$*

We now lower bound the number of distinct prime ideals in the factorization of the ideal $2R$ in the $\bar{m} = M_{p_n} = (2^{p_n} - 1)$th cyclotomic ring $R = \mathcal{O}_{\bar{m}}$. Recall from above that this number, which corresponds to the number of "$\mathbb{Z}_q$-slots," is equal to $\varphi(M_{p_n})/\text{ord}_{M_{p_n}}(2)$.(**Jacob:** TODO: Ensure ord is defined somewhere)

Lemma 2.11.8. *Let p_n be the nth prime. Then*

$$\frac{\varphi(M_{p_n})}{ord_{M_{p_n}} 2} \geq 2^{n(\ln(n)-3/8)}$$

Proof. Since $\text{ord}_{M_{p_n}}(2) = p_n$, by Corollary 2.11.5, we have that

$$\varphi(M_{p_n})/\text{ord}_{M_{p_n}}(2) \geq 2^{p_n - 1/\log_2 p_n - \log_2(p_n)}$$

The lemma then follows from direct verification for $n \leq 15$ and Lemma 2.11.1. □

We can now give the unconditional upper bound.

Theorem 2.11.9. *Let $R = R^{(r)}$ have index m_r where $m_r = 2^{k+1}$ for some integer $k > 1$, and let $p = 2$. Then there exists a tower of cyclotomic rings $S^{(i)}$ such that the size of the compositum at any step in our procedure is at most $2^{k+O(\sqrt{k\ln(k)})}$.*

Proof. We define the tower of cyclotomic rings $S^{(i)}$ so that the ratio of their indices $\ell_i/\ell_{i-1} = M_{p_i} = 2^{p_i} - 1$ (with $\ell_0 = 1$). Note that by Corollary 2.11.7, we have that ℓ_i/ℓ_{i-1} and ℓ_{i-1} are coprime. As the order of 2 modulo ℓ_{i-1} is $\prod_{j=1}^{i-1} p_j$ and hence coprime to p_i, we have that the "relative splitting number" of 2 in $S^{(i)}/S^{(i-1)}$ is $|\tilde{C}_i'| = \varphi(M_{p_i})/p_i$. As a result, via Lemma 2.11.8, we may choose the intermediate ring extension $R^{(r-i+1)}/R^{(r-i)}$ to be of dimension

$$|B'_{r-i+1}| = 2^{\lfloor \log_2(\varphi(M_{p_i})/p_i)\rfloor} \geq 2^{\log_2(\varphi(M_{p_i})/p_i)-1} \geq 2^{i(\ln(i)-3/8)-1}.$$

After t total steps, the index of $R^{(r-t)}$ will then be at most $2^{k+1-(\sum_{n=1}^{t} n(\ln(n)-3/8)-1)}$. A standard calculation gives that for $t \geq O(\sqrt{k/\ln(k)})$, the index will be at most 2 (and hence the dimension will be at most 1). As a result, the algorithm stops after at most $O(\sqrt{k/\ln(k)})$ steps.

If the algorithm has not stopped at the end of step i, then the dimension of the hybrid ring $H^{(i)}$ is at most

$$\varphi(m_{r-i}\ell_i) \leq \varphi(m_{r-i+1}\ell_{i-1}) \cdot \varphi(M_{p_i}) \cdot 2^{-\lfloor \log_2(\varphi(M_{p_i})/p_i)\rfloor} \leq \varphi(m_r) 2^i \prod_{j=1}^{i} p_j$$

Thus, by Corollary 2.11.3 the maximum dimension of a hybrid ring is

$$\varphi(m_r)2^{O(\sqrt{k/\ln(k)})}e^{\vartheta(p_{O(\sqrt{k/\ln(k)})})} \leq 2^{k+O\left(\sqrt{k\ln(k)}\right)}$$

The dimension of the compositum in step $i+1$ is the dimension of the hybrid ring $H^{(i)}$ multiplied by the dimension of $S^{(i+1)}/S^{(i)}$. As the latter value is at most $2^{O(\sqrt{k\ln(k)})}$, we can upper bound the compositum's maximum dimension during the algorithm by

$$\varphi(m_{r-i}\ell_i)\varphi(\ell_{i+1})/\varphi(\ell_i) \leq 2^{k+O(\sqrt{k\ln(k)})}$$

□

2.11.4 Heuristic Bounds

We give a heuristic upper bound on the maximum size of the compositum. For an appropriate choice of $n = \theta(\lambda)$ (see below), let $P_n := \{\text{odd primes } p : (p-1) \mid n\}$, $r := |P_n|$, and let p_i denote the ith largest prime in P_n. We define the ring S and its associated tower of cyclotomic rings $S^{(i)}$ so that the ratio of their indices $\ell_i/\ell_{i-1} = p_i$ (with $\ell_0 = 1$). We wish to find the maximum k such that we can apply the ring-switching procedure using the tower of cyclotomics $S^{(i)}$from an initial ring R of index $m_r = 2^{k+1}$. We will show heuristically that, if the initial ciphertext is fully packed, the maximum dimension of the compositum during the procedure will be at most $2^{k+O(\ln(k)\ln(\ln(k)))}$.

Let $W_n := \sum_{p\in P_n}\log_2(\varphi(p))$. Since the order of every prime p in P_n divides n, the total number of "slots" in the full ring S is at least $2^{X_n}/n = 2^{X_n - \log_2 n}$. Accounting for the at most 2^r slots "wasted" during the procedure (see Section 2.11.1), we are then able to accomodate any $k \leq W_n - r - \log_2 n$. Noting the definition of r above, by defining $X_n = \sum_{p\in P_n}(\log_2(\varphi(p)) - 1)$, we can accomodate $k \leq X_n - \log_2 n$.

It remains to determine the approximate value of X_n. We heuristically show that for a randomly chosen integer n (in the neighborhood of λ),

$$X_n \approx x_n := \frac{2}{\ln(2)}d(n/2)$$

where $d(n)$ denotes the number of divisors of n.

Consider an even factor m of n. The prime number theorem implies that a random odd integer in the neighborhood of $m+1$ is prime with probability approximately $2/\ln(m)$. As a result, we can then say the "expected" contribution of m to the sum defining X_n is approximately $\frac{2}{\ln(m)} \cdot \log_2 \phi(m+1) = \frac{2}{\ln(2)}$. Since there are $d(n/2)$ such factors, the heuristics for a random integer n follow immediately.

If n has many distinct prime factors relative to its size, then most factors m of n will have a large number of distinct prime factors relative to their size, which in turn makes it more likely that $m+1$ is prime than a random integer (since $m+1$ cannot divide any of the factors of m), so that heuristically, the "expected" contribution to the above sum in this case will actually be greater than $2/\ln(2)$.

Let $Z_N = \{n \in [N] : d(n/2) \geq 2^{c\ln(N)/\ln(\ln(N))}\}$, where $.5 \leq c \leq 1$. We use the prime number theorem and the results of [9] to show that for large N, $|Z_N| \geq 2^{c\ln(N)/\ln(\ln(N))}$. Indeed, $\pi(2\ln(N)) = \frac{2\ln(N)}{\ln(\ln(N))}$ and the product of any subset of $c\ln(N)/\ln(\ln(N))$ of the primes less than $2\ln(N)$ is at most $N/2$ and thus corresponds to an integer $x \leq N$ such that $d(x/2) \geq 2^{c\ln(N)/\ln(\ln(N))}$. It is easy to see that there are at least $2^{c\ln(N)/\ln(\ln(N))}$ such subsets, so that $|Z_N| \geq 2^{c\ln(N)/\ln(\ln(N))}$. As a result, under the heuristic, it is likely that for any large N, there is indeed some integer $n \leq N$ such that $X_n \geq 2^{c\ln(N)/\ln(\ln(N))} \geq 2^{c\ln(n)/\ln(\ln(n))}$. Since finding such an integer only requires knowing its factors, and since N itself is polynomial in the security parameter (so that it has $\log_2(N)$ bits and may be factored in polynomial time), one may find such an integer via a brute-force search in polynomial time.

Let C denote the largest dimension the compositum takes during the procedure. Then combining the above heuristic results, we have that

$$k \geq 2^{\theta(\ln(n)/\ln(\ln(n)))} - \log_2(n) \quad \text{and} \quad \log_2(C) \leq 2^{\theta(\ln(n)/\ln(\ln(n)))} + \log_2(n),$$

so that

$$\log_2(C) = k + O(\ln(k)\ln(\ln(k)))$$

As a result, if we pack $\lambda = \theta(2^k)$ plaintexts into the initial ciphertext, we have that

the compositum will be at most $\lambda \ln(\lambda)^{O(\ln(\ln(\ln(\lambda))))}$, which is very slightly larger than quasilinear.

CHAPTER III

EFFICIENT BOOTSTRAPPING OF UNPACKED CIPHERTEXTS

3.1 Bootstrapping with Polynomial Error Growth

In a major milestone for fully homomorphic encryption and bootstrapping, Brakerski and Vaikuntanathan (BV) [25] gave a bootstrapping method that incurs only *polynomial* error in the security parameter λ. This allows security to be based on the learning with errors (LWE) problem [75] with inverse-polynomial error rates, and hence on worst-case lattice problems with *polynomial* approximation factors (via the reductions of [75, 74, 22]). The BV method is centered around two main components:

1. the recent homomorphic cryptosystem of Gentry, Sahai, and Waters (GSW) [53], specifically, the "*quasi-additive*" nature of its error growth under homomorphic multiplication; and
2. the "circuit sequentialization" property of Barrington's Theorem [11], which converts any depth-d circuit (of NAND gates) into a length-4^d "branching program," which is essentially a fixed sequence of conditional multiplications.

Since decryption in homomorphic cryptosystems can be implemented in circuit depth $O(\log \lambda)$, Barrington's Theorem yields an equivalent branching program of length $4^d = \mathrm{poly}(\lambda)$. Moreover, the quasi-additive error growth of GSW multiplication means that homomorphic evaluation of the branching program incurs only $\mathrm{poly}(\lambda)$ error, as demonstrated in [25].

The polynomial error growth of the BV bootstrapping algorithm is a terrific feature, but the method also has two significant drawbacks: it comes at a high price in *efficiency*, and the error growth is a *large* polynomial. Both issues arise from the fact that in this context, Barrington's Theorem yields a branching program of large polynomial length. Existing

analyses (e.g., [23, Lemma 4.5]) of decryption circuits (for cryptosystems with 2^λ security) yield depths of $c \log \lambda$ for some unspecified but moderately large constant $c \geq 3$, which translates to a branching program of length at least $\lambda^{2c} \geq \lambda^6$. (Even if the depth were to be improved, there is a fundamental barrier of $c \geq 1$, which yields length $\Omega(\lambda^2)$.) The branching program length is of course a lower bound on the number of homomorphic operations required to bootstrap, and it also largely determines the associated error growth and final lattice approximation factors.

Separately, Brakerski and Vaikuntanathan also show how to obtain better lattice approximation factors through a kind of "dimension leveraging" technique, but this comes at an even higher price in efficiency: if the original error growth was λ^c for some constant c, then the technique involves running the bootstrapping procedure with GSW ciphertexts of dimension $n \approx \lambda^{c/\epsilon}$, where the choice of $\epsilon \in (0, 1)$ yields a final approximation factor of $\tilde{O}(n^{3/2+\epsilon})$. The high cost of dimension leveraging underscores the importance of obtaining smaller error growth and approximation factors via other means.

3.1.1 Our Results

Our main result is a new bootstrapping method having substantially smaller runtime *and* (polynomial) error growth than the recent one from [25]. The improvements come as a result of treating decryption as an *arithmetic* function, in contrast to most earlier works which treated decryption as a boolean circuit. This avoids the circuitous and inefficient path of constructing a shallow circuit and then transforming it via Barrington's Theorem into a branching program of (large) polynomial length. Instead, we show how to *directly* evaluate the decryption function in an elementary and efficient arithmetic form, using just basic facts about cyclic groups. See the next subsection for a detailed overview.

Our method requires only a *quasi-linear* $\tilde{O}(\lambda)$ number of homomorphic operations on GSW ciphertexts, to bootstrap essentially any LWE-based encryption scheme with 2^λ security under conventional assumptions. This performance is *quasi-optimal* (i.e., ignoring

polylogarithmic factors) for a system with bitwise encryption (like GSW), because the decryption function must depend on at least λ secret key bits. When instantiated with a GSW scheme based on ring-LWE [68], in which the cost of each homomorphic operation is only $\tilde{O}(\lambda)$ bit operations, the total runtime of our algorithm is a respectable $\tilde{O}(\lambda^2)$ bit operations.[1]

Regarding error growth, the security of our basic scheme can be based on LWE with error rates as large as $1/\tilde{O}(\lambda \cdot n)$, where $n = \tilde{\Omega}(\lambda)$ is the LWE dimension used in the GSW scheme. Taking $n = \tilde{O}(\lambda)$ to be asymptotically minimal, this translates to lattice approximation factors of $\tilde{O}(n^3)$, which is quite close to the $\tilde{O}(n^{3/2})$ factors that plain public-key encryption can be based upon (and quite a bit smaller than for many other applications of LWE!). We emphasize that these small factors are obtained *directly* from our construction with *small* LWE dimensions. To further improve the assumptions at a (high) cost in efficiency, we can let $n = \lambda^{1/\epsilon}$ to directly yield $\tilde{O}(n^{2+\epsilon})$ factors for any $\epsilon \in (0, 1)$, or we can use the successive dimension/modulus-reduction technique from [25] to obtain $\tilde{O}(n^{3/2+\epsilon})$ factors.

Simpler GSW variant. As a contribution of independent interest, we also give a variant of the GSW cryptosystem that we believe is technically simpler, along with a tighter analysis of error terms under its homomorphic operations (see Section 3.2). The entire scheme, security proof, and error analysis fit into just a few lines of standard linear algebra notation, and our variant enjoys additional useful properties like full "re-randomization" of error terms as a natural side effect. The error analysis is also very clean and tight, due to the use of *subgaussian* random variables instead of coarser measures like the ℓ_2 or ℓ_∞ norms. One nice consequence of this approach is that the error in a homomorphic product of d ciphertexts grows with $\sqrt{d}$, rather than linearly as in prior analyses. This is important for establishing the small error growth of our bootstrapping method.

[1] Homomorphic operations in standard-LWE-based GSW are quite a bit more expensive, due to matrix multiplications of dimensions exceeding λ.

3.1.2 Technical Overview

Here we give an overview of the main ideas behind our new bootstrapping method. We start by recalling in more detail the main ideas behind the work of Brakerski and Vaikuntanathan [25], which uses the Gentry-Sahai-Waters (GSW) [53] homomorphic encryption scheme to obtain FHE from LWE with inverse-polynomial error rates, and hence from lattice problems with polynomial approximation factors.

The starting point is a simple observation about the GSW encryption scheme: for encryptions $\mathbf{C}_1, \mathbf{C}_2$ of messages $\mu_1, \mu_2 \in \mathbb{Z}$, the error in the homomorphic product $\mathbf{C}_1 \boxdot \mathbf{C}_2$ of $\mu_1 \cdot \mu_2$ is *"quasi-additive"* and *asymmetric* in the ciphertexts' respective errors e_1, e_2, namely, it is $e_1 \cdot \mathrm{poly}(n) + \mu_1 \cdot e_2$, where n is the dimension of the ciphertexts. (The error in the homomorphic sum $\mathbf{C}_1 \boxplus \mathbf{C}_2$ is simply the sum $e_1 + e_2$ of the individual errors.) This property has a number of interesting consequences. For example, Brakerski and Vaikuntanathan use it to show that the homomorphic product of many freshly encrypted 0-1 messages, if evaluated *sequentially* in a right-associative manner, has error that grows at most linearly in the number of ciphertexts. More generally, the homomorphic product of many encrypted *permutation* matrices—i.e., 0-1 matrices in which each row and column has exactly one nonzero entry—has similarly small error growth.

The next main idea from [25] is to use Barrington's Theorem to express the boolean decryption circuit of depth $d = O(\log \lambda)$ as a branching program of length $4^d = \mathrm{poly}(\lambda)$ over the symmetric group S_5, or equivalently, the multiplicative group of 5-by-5 permutations matrices. Their bootstrapping algorithm homomorphically (and sequentially) multiplies appropriate encrypted permutation matrices to evaluate this branching program on a given input ciphertext, thereby homomorphically decrypting it. Since evaluation is just a homomorphic product of $\mathrm{poly}(\lambda)$ permutation matrices, the error in the final output ciphertext is only polynomial, and the LWE parameters can be set to yield security assuming the hardness of lattice problems for polynomial approximation factors.

3.1.2.1 Our Approach

Our bootstrapping method retains the use of symmetric groups and permutation matrices, but it works without the "magic" of Barrington's Theorem, by treating decryption more directly and efficiently as an *arithmetic* function, not a boolean circuit. In more detail, the decryption function for essentially every LWE-based cryptosystem can without loss of generality (via standard bit-decomposition techniques) be written as a "rounded inner product" between the secret key $\mathbf{s} \in \mathbb{Z}_q^d$ and a *binary* ciphertext $\mathbf{c} \in \{0,1\}^d$, as

$$\mathsf{Dec}(\mathbf{s}, \mathbf{c}) = \lfloor \langle \mathbf{s}, \mathbf{c} \rangle \rceil_2 \in \{0,1\}.$$

Here the modular rounding function $\lfloor \cdot \rceil_2 \colon \mathbb{Z}_q \to \{0,1\}$ indicates whether its argument is "far from" or "close to" 0 (modulo q), and the dimension d and modulus q can both be made as small as quasi-linear $\tilde{O}(\lambda)$ in the security parameter via dimension/modulus reduction [23], while still providing provable 2^λ security under conventional lattice assumptions. Note that the inner product itself is just a subset-sum of the $\mathbb{Z}_q$-entries of $\mathbf{s}$ indicated by $\mathbf{c}$, and uses only the additive group structure of $\mathbb{Z}_q$.

Embedding $\mathbb{Z}_q$ into S_q. As a warm up, we first observe that the additive group $\mathbb{Z}_q$ embeds (i.e., has an injective homomorphism) into the symmetric group S_q, the multiplicative group of q-by-q permutation matrices. (This is just a special case of Cayley's Theorem, which says that any finite group G embeds into $S_{|G|}$.) The embedding is very simple: $x \in \mathbb{Z}_q$ maps to the permutation that cyclically rotates by x positions. Moreover, any such permutation can be represented by an indicator vector in $\{0,1\}^q$ with its 1 in the position specified by x, and its permutation matrix is obtained from the cyclic rotations of this vector. In this representation, a sum $x + y$ can be computed in $O(q^2)$ bit operations by expanding x's indicator vector into its associated permutation matrix, and then multiplying by y's indicator vector. This representation also makes the *rounding function* $\lfloor \cdot \rceil_2 \colon \mathbb{Z}_q \to \{0,1\}$ trivial to evaluate: one just sums the entries of the indicator vector corresponding to those values in $\mathbb{Z}_q$ that round to 1.

These ideas already yield a new and simple bootstrapping algorithm that appears to have better runtime and error growth than can be obtained using Barrington's Theorem. The bootstrapping key is an encryption of each coordinate of the secret key $\mathbf{s} \in \mathbb{Z}_q^d$, represented as a dimension-q indicator vector, for a total of $d \cdot q = \tilde{O}(\lambda^2)$ GSW ciphertexts. To bootstrap a ciphertext $\mathbf{c} \in \{0,1\}^d$, the inner product $\langle \mathbf{s}, \mathbf{c} \rangle \in \mathbb{Z}_q$ is computed homomorphically as a subset-sum using the addition method described above, in $O(d \cdot q^2) = \tilde{O}(\lambda^3)$ homomorphic operations. The rounding function is then applied homomorphically, using just $O(q) = \tilde{O}(\lambda)$ additions.

Embedding $\mathbb{Z}_q$ into smaller symmetric groups. While the above method yields some improvements over prior work, it is still far from optimal. Our second main idea is an efficient way of embedding $\mathbb{Z}_q$ into a *much smaller* symmetric group S_r for some $r = \tilde{O}(1)$, such that the rounding function can still be efficiently evaluated (homomorphically). We do so by letting the modulus $q = \prod_i r_i$ be the product of many small prime powers r_i of distinct primes. (We can use such a q by modulus switching, as long as it remains sufficiently large to preserve correctness of decryption.) Using known bounds on the distribution of primes, it suffices to let the r_i be maximal prime powers bounded by $O(\log \lambda)$, of which there are at most $O(\log \lambda / \log\log \lambda)$.

By the Chinese Remainder Theorem, the additive group $\mathbb{Z}_q$ is isomorphic (via the natural homomorphism) to the product group $\prod_i \mathbb{Z}_{r_i}$, which then embeds into $\prod_i S_{r_i}$ as discussed above. Therefore, we can represent any $x \in \mathbb{Z}_q$ as a tuple of $O(\log \lambda)$ indicator vectors of length $r_i = O(\log \lambda)$ representing $x \pmod{r_i}$, and can perform addition by operating on the indicator vectors as described above. In this representation, the rounding function is no longer just a sum, but it can still be expressed relatively simply as

$$\lfloor x \rceil_2 = \sum_{v \in \mathbb{Z}_q \text{ s.t. } \lfloor v \rceil_2 = 1} [x = v],$$

where each equality test $[x = v]$ returns 0 for false and 1 for true.[2] In turn, each equality test

[2]Note that we are not using any special property of the rounding function here; any boolean function

$[x = v]$ is equivalent to the product of equality tests $[x = v \pmod{r_i}]$, each of which can be implemented trivially in our representation by selecting the appropriate entry of the indicator vector for $x \pmod{r_i}$. All of these operations have natural homomorphic counterparts in our representation, so we get a corresponding bootstrapping algorithm.

As a brief analysis, each coordinate of the secret key $\mathbf{s} \in \mathbb{Z}_q^d$ is encrypted as $\sum_i r_i = \tilde{O}(1)$ GSW ciphertexts, for a total of $\tilde{O}(d) = \tilde{O}(\lambda)$ ciphertexts in the bootstrapping key. Similarly, each addition or equality test over $\mathbb{Z}_q$ takes $\tilde{O}(1)$ homomorphic operations, for a total of $\tilde{O}(d + q) = \tilde{O}(\lambda)$. Both of these measures are quasi-optimal when relying on a scheme that encrypts one bit per ciphertext (like GSW). By contrast, bootstrapping using Barrington's Theorem requires at least $4^{c \log \lambda} = \lambda^{2c}$ homomorphic operations to evaluate the branching program, where $c \log \lambda$ is the depth of the decryption circuit *using NAND gates* (of fan-in 2). To our knowledge, upper bounds on the constant c have not been optimized or even calculated explicitly, but existing analyses like [23, Lemma 4.5] yield $c \gg 3$, and the necessary dependence on λ inputs bits for 2^λ security yields a fundamental barrier of $c \geq 1$.

3.1.2.2 Related Work on Branching Programs

Several works have extended and improved Barrington's Theorem for the simulation of general circuits and formulas via branching programs, e.g., [28, 33]. Of particular interest here is the thesis of Sinha [81], which gave quasi-linear-size, log-width branching programs for threshold functions (i.e., those which output 1 if at least some k of the n inputs are 1) and "mod" functions (i.e., those which output 1 if the number of 1s in the input is zero modulo some d). Similarly to our techniques, Sinha's construction uses the Chinese Remainder Theorem over many small primes in an essential way.

Because decryption in LWE-based cryptosystems involves modular addition, and can be implemented in constant depth (and polynomial size) by threshold gates, it might be possible to bootstrap in a quasi-linear number of homomorphic operations by using Sinha's

$f \colon \mathbb{Z}_q \to \{0, 1\}$ can be expressed similarly by summing over $f^{-1}(1)$.

results in place of Barrington's Theorem. However, we have not seen a way to make this work concretely.

3.1.3 Generalizations and Improvements

Since the preliminary publication of this work, several improvements and generalizations have appeared. Here we give a brief overview of these new works.

The most important of these subsequent works, by Ducas and Micciancio [38], specializes our algorithm to the ring setting. Their bootstrapping algorithm has the same high-level structure as the one in this work. However, they utilize properties of the ring structure to improve beyond a simple ring-based implementation of our algorithm. In our work, we use an indicator vector to represent elements of the cyclic group $\mathbb{Z}_q$, and reduce from an additional linear factor to an additional logarithmic factor by choosing q to have a large number of distinct prime factors. In their work, they work over a ring that contains q-th roots of unity, and they represent $i \in \mathbb{Z}_q$ as x^i where x is a primitive q-th root of unity. This allows them to eliminate the additional logarithmic factor. As a result, they achieve what is currently the fastest bootstrapping algorithm (when considering speed in a non-amortized manner), taking only 0.6 seconds to bootstrap on a standard personal computer.

Another work by Orsini, van der Pol and Smart [73] generalizes our idea of using a specialized representation for bootstrapping. They describe a generalized framework where bootstrapping consists of two parts. The first part involves computing a representation of $\mathbb{Z}_q$ in some group $\mathbb{G}$, which they refer to as rep. The second step involves homomorphically evaluating the "rounding" part of bootstrapping via a map they refer to as red. By initially using a polynomial representation, they show that the circuit for red can be evaluated in depth $\log \log q$, an improvement over previous algorithms when viewed in this framework. Unfortunately, the rep step takes depth $\log q$, similarly to previous bootstrapping algorithms. Moreover, it appears that given only addition and multiplication as basic operations, $\log q$

depth is in fact a tight lower bound for evaluating mod p modulo q [12], meaning improvements beyond $\log q$ will likely require an underlying somewhat homomorphic encryption scheme which has either more expressive basic operations or a less complex decryption circuit.

Finally, Hiromasa, Abe and Okamoto [60] show how to modify the GSW encryption scheme to support the encryption of matrices of $\{0, 1\}$ elements. While the original GSW scheme allows this by encrypting each element of the matrix in a separate ciphertext, this requires the ciphertexts to grow by a multiplicative factor linear in the number of matrix elements. Their technique allows the encryption of $r \times r$ matrices of $\{0, 1\}$ elements while growing each dimension of the ciphertexts additively instead of multiplicatively, providing a much more efficient solution. Using this technique, they are able to reduce the growth of the error in the scheme by a factor $\sqrt{n}$ (where n is the underlying lattice dimensions).

3.2 GSW Cryptosystem

Here we present a variant of the Gentry-Sahai-Waters homomorphic encryption scheme [53] (hereafter called GSW), which we believe is simpler to understand at a technical level. We also give a tighter analysis of its error growth under homomorphic operations.

3.2.1 Background

We first recall some standard background (see, e.g., [71] for further details).

3.2.1.1 Subgaussian Random Variables

In our construction it is very convenient to analyze the behavior of "error" terms using the standard notion of *subgaussian* random variables. (For further details and full proofs, see [85].) A real random variable X (or its distribution) is subgaussian with parameter $r > 0$ if for all $t \in \mathbb{R}$, its (scaled) moment-generating function satisfies $\mathbb{E}[\exp(2\pi tX)] \leq \exp(\pi r^2 t^2)$. By a Markov argument, X has Gaussian tails, i.e., for all $t \geq 0$, we have

$$\Pr[|X| \geq t] \leq 2\exp(-\pi t^2/r^2). \tag{3.2.1}$$

(If $\mathbb{E}[X] = 0$, then Gaussian tails also imply subgaussianity.) Any B-bounded centered random variable X (i.e., $\mathbb{E}[X] = 0$ and $|X| \leq B$ always) is subgaussian with parameter $B\sqrt{2\pi}$.

Subgaussianity is homogeneous, i.e., X is subgaussian with parameter r, then cX is subgaussian with parameter $c \cdot r$ for any constant $c \geq 0$. Subgaussians also satisfy *Pythagorean additivity*: if X_1 is subgaussian with parameter r_1, and X_2 is subgaussian with parameter r_2 conditioned on *any* value of X_1 (e.g., if X_1 and X_2 are independent), then $X_1 + X_2$ is subgaussian with parameter $\sqrt{r_1^2 + r_2^2}$. By induction this extends to the sum of any finite number of variables, each of which is subgaussian conditioned on any values of the previous ones.

We extend the notion of subgaussianity to vectors: a random real vector $\mathbf{x}$ is subgaussian with parameter r if for all fixed real unit vectors $\mathbf{u}$, the marginal $\langle \mathbf{u}, \mathbf{x} \rangle \in \mathbb{R}$ is subgaussian with parameter r. In particular, it follows directly from the definition that the concatenation of variables or vectors, each of which is subgaussian with common parameter r conditioned on any values of the prior ones, is also subgaussian with parameter r. Homogeneity and Pythagorean additivity clearly extend to subgaussian vectors as well, by linearity.

The next claim follows directly from the material in [85, Section 5.2.4 and Proposition 5.16].

Lemma 3.2.1. *Let $\mathbf{x} \in \mathbb{R}^n$ be a random vector with independent coordinates that are subgaussian with parameter r. Then for some universal constant $C > 0$, we have $\Pr[\|\mathbf{x}\|_2 > C \cdot r\sqrt{N}] \leq 2^{-\Omega(N)}$.*

3.2.1.2 Gadget Matrix

For a modulus q, let $\ell = \lceil \log_2 q \rceil$ and define the "gadget" (column) vector $\mathbf{g} = (1, 2, 4, \ldots, 2^{\ell-1}) \in \mathbb{Z}_q^\ell$. Note that the penultimate entry $2^{\ell-2}$ of $\mathbf{g}$ is in the interval $[q/4, q/2) \bmod q$. It will be convenient to use the following randomized "decomposition" function.

Claim 3.2.2 (Adapted from [71]). *There is a randomized, efficiently computable function* $\mathbf{g}^{-1}\colon \mathbb{Z}_q \to \mathbb{Z}^\ell$ *such that* $\mathbf{x} \leftarrow \mathbf{g}^{-1}(a)$ *is subgaussian with parameter* $O(1)$*, and always satisfies* $\langle \mathbf{g}, \mathbf{x} \rangle = a$*.*

Briefly, the algorithm described in the claim works as follows (though for our application it is not necessary to understand its internals): let $\mathbf{S} \in \mathbb{Z}^{\ell \times \ell}$ be the basis of the lattice $\Lambda^\perp(\mathbf{g}^t) = \{\mathbf{z} \in \mathbb{Z}^\ell : \langle \mathbf{g}, \mathbf{z} \rangle = 0 \in \mathbb{Z}_q\}$ as constructed in [71], whose Gram-Schmidt vectors all have Euclidean norm $O(1)$. Given $a \in \mathbb{Z}_q$, run a randomized version of the nearest-plane algorithm [7] with basis $\mathbf{S}$ to sample from the coset $\Lambda^\perp_a(\mathbf{g}^t) = \{\mathbf{x} : \langle \mathbf{g}, \mathbf{x} \rangle = a\}$, where in each iteration of the algorithm we choose the coefficient of the ith basis vector to have expectation zero over $\{c_i - 1, c_i\}$ for an appropriate $c_i \in \frac{1}{q}\mathbb{Z} \cap [0, 1)$. In particular, the coefficient is subgaussian with parameter $\sqrt{2\pi}$ given any fixed values of the previous coefficients. The final output $\mathbf{x}$ is the linear combination of the (orthogonal) Gram-Schmidt vectors of $\mathbf{S}$ with these coefficients, and is therefore subgaussian with parameter $O(1)$.

For vectors and matrices over $\mathbb{Z}_q$, define the randomized function $\mathbf{G}^{-1}\colon \mathbb{Z}_q^{n \times m} \to \mathbb{Z}^{n\ell \times m}$ by applying $\mathbf{g}^{-1}$ independently to each entry. Notice that for any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, if $\mathbf{X} \leftarrow \mathbf{G}^{-1}(\mathbf{A})$ then $\mathbf{X}$ has subgaussian parameter $O(1)$ and

$$\mathbf{G} \cdot \mathbf{X} = \mathbf{A}, \quad \text{where} \quad \mathbf{G} = \mathbf{g}^t \otimes \mathbf{I}_n = \mathrm{diag}(\mathbf{g}^t, \ldots, \mathbf{g}^t) \in \mathbb{Z}_q^{n \times n\ell} \tag{3.2.2}$$

is the block matrix with n copies of $\mathbf{g}^t$ as diagonal blocks, and zeros elsewhere.

3.2.2 Cryptosystem and Homomorphic Operations

The GSW scheme is parameterized by a dimension n, a modulus q with $\ell = \lceil \log_2 q \rceil$, and some error distribution χ over $\mathbb{Z}$ which we assume to be subgaussian. Formally, the message space is the ring of integers $\mathbb{Z}$, though for bootstrapping we only work with ciphertexts encrypting messages in $\{0, 1\} \subset \mathbb{Z}$. The ciphertext space is $\mathcal{C} = \mathbb{Z}_q^{n \times n\ell}$. For simplicity we present just a symmetric-key scheme, which is sufficient for our purposes (it can be converted to a public-key or even attribute-based scheme, as described in [53]).

Our GSW variant differs from the original scheme described in [53] in two main ways:

1. In [53], a ciphertext is a square binary matrix $\mathbf{C} \in \{0,1\}^{n\ell}$, a secret key is a "structured" mod-q vector $\mathbf{s} \in \mathbb{Z}_q^{n\ell}$ (having large entries), and $\mathbf{s}$ is an "approximate mod-q eigenvector" of $\mathbf{C}$, in the sense that $\mathbf{s}^t\mathbf{C} \approx \mu\mathbf{s}^t \pmod q$, where $\mu \in \mathbb{Z}$ is the message.

   In our variant, a ciphertext is a rectangular mod-q matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times n\ell}$, a secret key is some (unstructured, short) integer vector $\mathbf{s} \in \mathbb{Z}^n$, and $\mathbf{s}^t\mathbf{C} \approx \mu \cdot \mathbf{s}^t\mathbf{G} \pmod q$, i.e., $\mathbf{s}$ and $\mathbf{G}^t\mathbf{s}$ are corresponding left- and right- "approximate singular vectors" of $\mathbf{C}$.

   The difference between these two variants turns out to be purely syntactic, in that we can efficiently and "losslessly" switch between them (without needing the secret key). However, we believe that our variant leads to simpler notation and easier-to-understand operations and analysis.

2. The second difference is more substantial: our homomorphic multiplication procedure uses the *randomized* $\mathbf{G}^{-1}(\cdot)$ operation from Claim 3.2.2. This yields a few important advantages, such as a very tight and simple error analysis using subgaussianity (see Lemma 3.2.6), and the ability to *completely re-randomize* the error in a ciphertext (see Corollary 3.2.7).

We now describe the scheme formally.

GSW.$\mathsf{Gen}()$**:** choose $\bar{\mathbf{s}} \leftarrow \chi^{n-1}$ and output secret key $\mathbf{s} = (\bar{\mathbf{s}}, 1) \in \mathbb{Z}^n$.

GSW.$\mathsf{Enc}((\bar{\mathbf{s}}, 1), \mu \in \mathbb{Z})$**:** choose $\bar{\mathbf{C}} \leftarrow \mathbb{Z}_q^{(n-1)\times n\ell}$ and $\mathbf{e} \leftarrow \chi^m$, let $\mathbf{b}^t = \mathbf{e}^t - \bar{\mathbf{s}}^t\bar{\mathbf{C}} \pmod q$, and output the ciphertext

$$\mathbf{C} = \begin{pmatrix} \bar{\mathbf{C}} \\ \\ \mathbf{b}^t \end{pmatrix} + \mu\mathbf{G} \in \mathcal{C},$$

where $\mathbf{G}$ is as defined in Equation (3.2.2). Notice that $\mathbf{s}^t\mathbf{C} = \mathbf{e}^t + \mu \cdot \mathbf{s}^t\mathbf{G} \pmod q$.

GSW.$\mathsf{Dec}(\mathbf{s}, \mathbf{C} \in \mathcal{C})$**:** let $\mathbf{c}$ be the penultimate column of $\mathbf{C}$, and output $\mu = \lfloor \langle \mathbf{s}, \mathbf{c} \rangle \rceil_2$, where $\lfloor \cdot \rceil_2 \colon \mathbb{Z}_q \to \{0,1\}$ indicates whether its argument is closer modulo q to 0 or to

$2^{\ell-2}$ (the penultimate entry of $\mathbf{g}$).[3]

Homomorphic addition is defined as $\mathbf{C}_1 \boxplus \mathbf{C}_2 = \mathbf{C}_1 + \mathbf{C}_2$.

Homomorphic multiplication is defined as $\mathbf{C}_1 \boxdot \mathbf{C}_2 \leftarrow \mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2)$, and is *right associative*. Notice that this is a randomized procedure, because $\mathbf{G}^{-1}$ is randomized.

The IND-CPA security of the scheme follows immediately from the assumed hardness of $\mathrm{LWE}_{n-1,q,\chi}$, where the entries of the secret are drawn from the error distribution χ (which is no easier than for a uniformly random secret; see [5, Lemma 2]). This is because a fresh ciphertext is just $\mu\mathbf{G}$ plus a matrix of $n\ell$ independent LWE samples under secret $\bar{\mathbf{s}}$, which are pseudorandom by assumption and hence hide $\mu\mathbf{G}$.

3.2.3 Analysis

Here we analyze the scheme's correctness and homomorphic operations.

Definition 3.2.3. We say that a ciphertext $\mathbf{C}$ is *designed* to encrypt message $\mu \in \mathbb{Z}$ (under a secret key $\mathbf{s}$) if it is a fresh encryption of μ, or if $\mathbf{C} = \mathbf{C}_1 \boxplus \mathbf{C}_2$ where $\mathbf{C}_1, \mathbf{C}_2$ are respectively designed to encrypt $\mu_1, \mu_2 \in \mathbb{Z}$ and $\mu = \mu_1 + \mu_2$, or similarly for homomorphic multiplication.

Definition 3.2.4. We say that a ciphertext $\mathbf{C}$ that is designed to encrypt $\mu \in \mathbb{Z}$ (under $\mathbf{s}$) has *error* vector $\mathbf{e}^t \in \mathbb{Z}^{n\ell}$ if $\mathbf{s}^t\mathbf{C} - \mu \cdot \mathbf{s}^t\mathbf{G} = \mathbf{e}^t \pmod q$.

For convenience later on, we also say the matrix $\mu\mathbf{G}$ is designed to encrypt μ, and has error $\mathbf{0}$. (This is essentially implied by the above definitions, since $\mu\mathbf{G}$ is indeed a fresh encryption of μ, assuming that zero is in the support of χ.) The next claim on the correctness of decryption follows immediately from the fact that $\mathbf{s} = (\bar{\mathbf{s}}, 1)$ and the penultimate column of $\mathbf{G}$ is $(0, \ldots, 0, 2^{\ell-2})$, where $2^{\ell-2} \in [q/4, q/2) \bmod q$.

[3] Note that we can decrypt messages in $\mathbb{Z} \cap [-\frac{q}{2}, \frac{q}{2})$, or any other canonical set of representatives of $\mathbb{Z}_q$, by "decoding" $\mathbf{s}^t\mathbf{C}$ to the nearest multiple of $\mathbf{s}^t\mathbf{G}$. The above decryption algorithm will be sufficient for our purposes.

Claim 3.2.5. *If* $\mathbf{C}$ *is designed to encrypt some* $\mu \in \{0,1\} \subset \mathbb{Z}$*, and has error vector* $\mathbf{e}^t$ *whose penultimate coordinate has magnitude less than* $q/8$*, then GSW.*$\mathsf{Dec}(\mathbf{s}, \mathbf{C})$ *correctly outputs* μ*.*

3.2.4 Error Growth from Homomorphic Operations

We now analyze the behavior of the error terms under homomorphic operations.

Lemma 3.2.6. *Suppose* $\mathbf{C}_1, \mathbf{C}_2$ *are respectively designed to encrypt* $\mu_1, \mu_2 \in \mathbb{Z}$ *and have error vectors* $\mathbf{e}_1^t, \mathbf{e}_2^t$*. Then* $\mathbf{C}_1 \boxplus \mathbf{C}_2$ *has error vector* $\mathbf{e}_1^t + \mathbf{e}_2^t$*, and* $\mathbf{C}_1 \boxdot \mathbf{C}_2$ *has error vector* $\mathbf{e}_1^t\mathbf{X} + \mu_1\mathbf{e}_2^t$*, where* $\mathbf{X} \leftarrow \mathbf{G}^{-1}(\mathbf{C}_2)$ *is the matrix used in the evaluation of* $\boxdot$*. In particular, for* any *values of* $\mathbf{C}_i, \mathbf{e}_i, \mu_i$*, the latter error vector is of the form* $\mathbf{e}^t + \mu_1\mathbf{e}_2^t$*, where the entries of* $\mathbf{e}$ *are independent and subgaussian with parameter* $O(\|\mathbf{e}_1\|)$*.*

Importantly, the error in $\mathbf{C}_1 \boxdot \mathbf{C}_2$ is *quasi-additive* and *asymmetric* with respect to the errors in $\mathbf{C}_1, \mathbf{C}_2$: while the first error vector $\mathbf{e}_1^t$ is multiplied by a short (subgaussian) matrix $\mathbf{X}$, the second error vector $\mathbf{e}_2^t$ is only multiplied by the (scalar) *message* μ_1, which we will ensure remains in $\{0,1\}$.

Proof. The first claim is immediate, by linearity. For the second claim, because $\mathbf{G} \cdot \mathbf{X} = \mathbf{C}_2$ we have

$$\begin{aligned}
\mathbf{s}^t(\mathbf{C}_1 \boxdot \mathbf{C}_2) &= \mathbf{s}^t\mathbf{C}_1 \cdot \mathbf{X} \\
&= (\mathbf{e}_1^t + \mu_1 \cdot \mathbf{s}^t\mathbf{G})\mathbf{X} \\
&= \mathbf{e}_1^t\mathbf{X} + \mu_1(\mathbf{e}_2^t + \mu_2 \cdot \mathbf{s}^t\mathbf{G}) \\
&= (\mathbf{e}_1^t\mathbf{X} + \mu_1\mathbf{e}_2^t) + \mu_1\mu_2 \cdot \mathbf{s}^t\mathbf{G}.
\end{aligned}$$

□

As observed in [25], the asymmetric noise growth allows for performing a long chain of homomorphic multiplications while only incurring a polynomial-factor error growth, because $\boxdot$ is defined to be right associative. For convenience of analysis, in such a chain we always include the fixed ciphertext $\mathbf{G}$, which is designed to encrypt $\mu = 1$ and has zero

error, as the rightmost ciphertext in the chain. This ensures that the error vector of the output ciphertext is subgaussian and essentially *independent* of the errors in the input ciphertexts (apart from their lengths), which leads to a simpler and tighter analysis. (In [25] a weaker independence guarantee was achieved by a separate "partial re-randomization" procedure, which requires additional public key material.)

Corollary 3.2.7. *Suppose that* $\mathbf{C}_i$ *for* $i \in [k]$ *are respectively designed to encrypt* $\mu_i \in \{0, \pm 1\}$ *and have error vectors* $\mathbf{e}_i^t$*. Then for* any *fixed values of these variables,*

$$\mathbf{C} \leftarrow \boxed{\bullet}_{i \in [k]} \mathbf{C}_i \ \boxdot \ \mathbf{G} = \mathbf{C}_1 \boxdot (\mathbf{C}_2 \boxdot (\cdots (\mathbf{C}_k \boxdot \mathbf{G}) \cdots))$$

has an error vector whose entries are mutually independent and subgaussian with parameter $O(\|\mathbf{e}\|)$*, where* $\mathbf{e}^t = (\mathbf{e}_1^t, \ldots, \mathbf{e}_k^t) \in \mathbb{Z}^{kn\ell}$ *is the concatenation of the individual error vectors.*

Proof. By Lemma 3.2.6, the error vector in $\mathbf{C}$ is $\sum_i \mathbf{e}_i^t \mathbf{X}_i$, where each $\mathbf{e}_i^t \mathbf{X}_i$ is a fresh independent vector that has mutually independent coordinates and is subgaussian with parameter $O(\|\mathbf{e}_i\|)$. The claim then follows by Pythagorean additivity. □

3.3 Symmetric Groups and $\mathbb{Z}_q$-Embeddings

Here we recall some basic facts about symmetric groups, which can be found in most abstract algebra textbooks, e.g., [62]. Let S_r denote the symmetric group of order r, i.e., the group of permutations (bijections) $\pi \colon \{1, \ldots, r\} \to \{1, \ldots, r\}$ with function composition as the group operation. The group S_r is isomorphic to the multiplicative group of r-by-r *permutation matrices* (i.e., 0-1 matrices with exactly one nonzero element in each row and each column), via the map that associates $\pi \in S_r$ with the permutation matrix $\mathbf{P}_\pi = [\mathbf{e}_{\pi(1)} \ \mathbf{e}_{\pi(2)} \ \cdots \ \mathbf{e}_{\pi(r)}]$, where $\mathbf{e}_i \in \{0, 1\}^r$ is the ith standard basis vector. For the remainder of this work we identify permutations with their associated permutation matrices.

The additive cyclic group $(\mathbb{Z}_r, +)$ embeds into the symmetric group S_r via the injective homomorphism that sends the generator $1 \in \mathbb{Z}_r$ to the "cyclic shift" permutation $\pi \in S_r$,

defined as $\pi(i) = i + 1$ for $1 \leq i < r$ and $\pi(r) = 1$.[4] Clearly, this embedding and its inverse can be computed efficiently. Notice also that the permutation matrices in the image of this embedding can be represented more compactly by just their first column, because the remaining columns are just the successive cyclic shifts of this column. Similarly, such permutation matrices can be multiplied in only $O(r^2)$ operations, since we only need to multiply one matrix by the first column of the other.

For our efficient bootstrapping algorithm, we need to efficiently embed a group $(\mathbb{Z}_q, +)$, for some sufficiently large q of our choice, into a symmetric group of order much smaller than q (e.g., polylogarithmic in q). This can be done as follows: suppose that $q = r_1 r_2 \cdots r_t$, where the r_i are pairwise coprime. Then by the Chinese Remainder Theorem, the ring $\mathbb{Z}_q$ is isomorphic to the direct product of rings $\mathbb{Z}_{r_1} \times \mathbb{Z}_{r_2} \times \cdots \times \mathbb{Z}_{r_t}$, and hence their additive groups are isomorphic as well. Combining this with the group embeddings of $(\mathbb{Z}_{r_i}, +)$ into S_{r_i}, we have an (efficient) group embedding from $(\mathbb{Z}_q, +)$ into $S_{r_1} \times S_{r_2} \times \cdots \times S_{r_t}$.[5]

Importantly for our purposes, q can be exponentially large in terms of $\max_i r_i$ above. This can be shown using lower bounds on the second Chebyshev function

$$\psi(x) := \sum_{p^k \leq x} \log p = \log\Big(\prod_{p \leq x} p^{\lfloor \log_p x \rfloor}\Big),$$

where the first summation is over all prime powers $p^k \leq x$, and the second is over all primes $p \leq x$; note that $p^{\lfloor \log_p x \rfloor}$ is the largest power of p not exceeding x. Therefore, the product q of all maximal prime powers $r_i = p^{\lfloor \log_p x \rfloor} \leq x$ is $\exp(\psi(x))$. Asymptotically, it is known that $\psi(x) = x \pm O(x/\log x)$, and we also have the nonasymptotic bound $\psi(x) \geq 3x/4$ for all $x \geq 7$ [80, Theorem 11]. In summary:

Lemma 3.3.1. *For all $x \geq 7$, the product of all maximal prime powers $r_i \leq x$ is at least $\exp(3x/4)$.*

[4] This is just a special case of Cayley's theorem, which says that any group G embeds into the symmetric group $S_{|G|}$.

[5] The latter group can be seen as a subgroup of S_r for $r = \sum_i r_i$, but it will be more efficient to retain the product structure.

3.4 Homomomorphic Encryption for Symmetric Groups

Brakerski and Vaikuntanathan [25] showed how to use the GSW encryption scheme to homomorphically compose permutations of five elements (i.e., to homomorphically compute the group operation in the symmetric group S_5) with small additive noise growth; the use of S_5 comes from its essential role in Barrington's theorem [11]. In [25], the homomorphic composition of permutations is intertwined with the evaluation of a branching program given by Barrington's theorem. Here we give, as a "first-class object," a homomorphic cryptosystem for any symmetric group S_r. The ability to use several different small values of r, along with a homomorphic equality test that we design, will be central to our bootstrapping algorithm.

3.4.1 Encryption Scheme

We now describe our (symmetric-key) homomorphic encryption scheme for symmetric groups, called HEPerm. Let $\mathcal{C}$ denote the ciphertext space for an appropriate instantiation of the GSW scheme, which we treat as a "black box." A secret key sk for HEPerm is simply a secret key for the GSW scheme.

- HEPerm.$\mathsf{Enc}(sk, \pi \in S_r)$: let $\mathbf{P} = (p_{i,j}) \in \{0,1\}^{r \times r}$ be the permutation matrix associated with π. Output an entry-wise encryption of $\mathbf{P}$, i.e., the ciphertext

$$\mathbf{C} = (c_{i,j}) \in \mathcal{C}^{r \times r}, \text{ where } c_{i,j} \leftarrow \mathsf{Enc}(sk, p_{i,j}).$$

(Decryption follows in the obvious manner.) As with the GSW system, we say that a ciphertext $\mathbf{C} \in \mathcal{C}^{r \times r}$ is *designed* to encrypt a permutation $\pi \in S_r$ (or its permutation matrix $\mathbf{P}_\pi$) if its $\mathcal{C}$-entries are designed to encrypt the corresponding entries of $\mathbf{P}_\pi$. For convenience, we let $\mathbf{J} \in \mathcal{C}^{r \times r}$ denote the ciphertext that encrypts the identity permutation with zero noise, which is built in the expected way from the fixed zero-error GSW ciphertexts that encrypt 0 and 1.

We now show how to homomorphically compute two operations: the standard composition operation for permutations, and an equality test.

Homomorphic composition $\mathbf{C}^\pi \boxdot \mathbf{C}^\sigma$: on ciphertexts $\mathbf{C}^\pi = (c^\pi_{i,j}), \mathbf{C}^\sigma = (c^\sigma_{i,j}) \in \mathcal{C}^{r\times r}$ encrypting permutations $\pi, \sigma \in S_r$ respectively, we compute one encrypting the permutation $\pi \circ \sigma$ by homomorphically evaluating the naïve matrix-multiplication algorithm.[6] That is, output $\mathbf{C} = (c_{i,j}) \in \mathcal{C}^{r\times r}$ where

$$c_{i,j} \leftarrow \boxplus_{\ell\in[r]} (c^\pi_{i,\ell} \boxdot c^\sigma_{\ell,j}) \in \mathcal{C}. \tag{3.4.1}$$

Just like $\boxdot$, we define $\boxdot$ to be right associative.

Homomorphic equality test $\mathsf{Eq}?(\mathbf{C}^\pi = (c^\pi_{i,j}), \sigma \in S_r)$**:** given a ciphertext encrypting some permutation $\pi \in S_r$ and a permutation $\sigma \in S_r$ (in the clear), output a ciphertext $c \in \mathcal{C}$ encrypting 1 if $\pi = \sigma$ and 0 otherwise, as

$$c \leftarrow \boxdot_{i\in[r]} c^\pi_{\sigma(i),i} \boxdot g,$$

where $g \in \mathcal{C}$ denotes the fixed zero-error encryption of 1. (Recall that $\boxdot$ is right associative.)

Observe that for the above two operations, the GSW ciphertext(s) in the output are designed to encrypt the appropriate $\{0,1\}$-message. For $\mathsf{Compose}$ this is simply by correctness of the matrix-multiplication algorithm. For $\mathsf{Eq}?$ this is because the output ciphertext is designed to encrypt 1 if and only if every $c_{\sigma(i),i}$ is designed to encrypt 1, which is the case if and only if $\mathbf{C}^\pi$ is in fact designed to encrypt σ. All that remains is to analyze the behavior of the error terms, which we do next.

[6] Note that asymptotically faster algorithms (e.g., Strassen's) do not appear suitable here, because the intermediate steps of these algorithms can result in values having magnitude greater than one, which can cause the error in the ciphertexts to grow much faster. In any case, this will not matter much in our application, since r will be small.

3.4.2 Analysis

Recalling that the GSW scheme is parameterized by n and q, denote its space of error vectors by $\mathcal{E} = \mathbb{Z}^m$ where $m = n\lceil \log_2 q \rceil$. The Euclidean norm on $\mathcal{E}^r = \mathbb{Z}^{mr}$ is defined in the expected way. In what follows it is often convenient to consider vectors and matrices over $\mathcal{E}$, i.e., each entry is itself a (row) vector in $\mathcal{E} = \mathbb{Z}^m$, and we switch between $\mathcal{E}^{h \times w}$ and $\mathbb{Z}^{h \times wm}$ as is convenient.

The following lemma describes the behavior of errors under the homomorphic composition operation $\boxdot$. Note that working with vectors and matrices over $\mathcal{E}$ lets us write a statement that is syntactically very similar to the one from Lemma 3.2.6, with a very similar proof.

Lemma 3.4.1. *Let $\mathbf{C}^\pi, \mathbf{C}^\sigma \in \mathcal{C}^{r \times r}$ respectively be designed to encrypt permutation matrices $\mathbf{P}^\pi, \mathbf{P}^\sigma \in \{0,1\}^{r \times r}$ with error matrices $\mathbf{E}^\pi, \mathbf{E}^\sigma \in \mathcal{E}^{r \times r}$. Then for* any *fixed values of these variables, $\mathbf{C}^\pi \boxdot \mathbf{C}^\sigma$ has error matrix $\mathbf{E} + \mathbf{P}^\pi \cdot \mathbf{E}^\sigma \in \mathcal{E}^{r \times r}$, where the $\mathbb{Z}$-entries of $\mathbf{E}$ are mutually independent, and those in its ith row are subgaussian with parameter $O(\|\mathbf{e}_i^\pi\|)$, where $\mathbf{e}_i^\pi$ is the ith row of $\mathbf{E}^\pi$.*

Proof. Let $\mathbf{C} \leftarrow \mathbf{C}^\pi \boxdot \mathbf{C}^\sigma$. It suffices to show that for all i, j, its (i,j)th entry $c_{i,j} \in \mathcal{C}$ has error

$$e_{i,j} + e^\sigma_{\pi^{-1}(i),j} \in \mathcal{E} = \mathbb{Z}^m,$$

where all the $\mathbb{Z}$-entries of all the $e_{i,j} \in \mathbb{Z}^m$ are mutually independent and subgaussian with parameter $O(\|\mathbf{e}_i^\pi\|)$, and $e^\sigma_{\ell,j}$ is the (ℓ, j)th entry of $\mathbf{E}^\sigma$. This follows directly from Equation (3.4.1) and Lemma 3.2.6: the error in each ciphertext $c^\pi_{i,\ell} \boxdot c^\sigma_{\ell,j}$ is $p^\pi_{i,\ell} \cdot e^\sigma_{\ell,j}$ plus a fresh vector whose entries are independent and subgaussian with parameter $O(\|e^\pi_{i,\ell}\|)$. Since $p^\pi_{i,\ell} = 1$ for $\ell = \pi^{-1}(i)$ and 0 otherwise, the claim follows by Pythagorean additivity of independent subgaussians. □

Similarly to a multiplication chain of GSW ciphertexts, we can perform a (right-associative) chain of compositions while incurring only small error growth. For convenience

of analysis, we always include the fixed zero-error ciphertext $\mathbf{J} \in \mathcal{C}^{r \times r}$ (which encrypts the identity permutation) as the rightmost ciphertext in the chain. The following corollary follows directly from Lemma 3.4.1 in the same way that Corollary 3.2.7 follows from Lemma 3.2.6.

Corollary 3.4.2. *Suppose that $\mathbf{C}_i \in \mathcal{C}^{r \times r}$ for $i \in [k]$ are respectively designed to encrypt permutation matrices $\mathbf{P}_i \in \{0,1\}^{r \times r}$ and have error matrices $\mathbf{E}_i \in \mathcal{E}^{r \times r}$. Then for* any *fixed values of these variables,*

$$\mathbf{C} \leftarrow \boxed{\circ}_{i \in [k]} \mathbf{C}_i \boxdot \mathbf{J} = \mathbf{C}_1 \boxdot (\mathbf{C}_2 \boxdot (\cdots (\mathbf{C}_k \boxdot \mathbf{J}) \cdots))$$

has an error matrix whose $\mathbb{Z}$-entries are mutually independent, and those in its ith row are subgaussian with parameter $O(\|\mathbf{e}_i\|)$, where $\mathbf{e}_i^t \in \mathcal{E}^{kr}$ is the ith row of the concatenated error matrices $[\mathbf{E}_1 \mid \cdots \mid \mathbf{E}_k]$.

Finally, since the $\mathsf{Eq}?$ procedure simply performs a chain of (right-associative) multiplications of GSW ciphertexts, Corollary 3.2.7 applies.

3.4.3 Optimizations for $\mathbb{Z}_r$ Embeddings

For bootstrapping, we use the above scheme only to encrypt elements in the cyclic subgroup $C_r \subseteq S_r$ that embeds the additive group $(\mathbb{Z}_r, +)$. As described in the preliminaries, an element $\pi \in C_r$ can be represented more compactly as an indicator (column) vector $\mathbf{p} \in \{0,1\}^r$ (rather than a matrix in $\{0,1\}^{r \times r}$), and its associated permutation matrix $\mathbf{P}_\pi$ is made up of the r cyclic rotations of $\mathbf{p}$. In addition, the composition of two permutations represented in this way as $\mathbf{p}, \mathbf{q}$ is given by the matrix-vector product $\mathbf{P}_\pi \cdot \mathbf{q}$, which may be computed in $O(r^2)$ operations, rather than $O(r^3)$ as in the general case. All of this translates directly to *encrypted* permutations in the expected way, i.e., ciphertexts are entry-wise encryptions in $\mathcal{C}^r$ of indicator vectors, etc.

Similarly, the equality test $\mathsf{Eq}?$ can be performed more efficiently when we restrict to the subgroup C_r: given r ciphertexts encrypting the entries of an indicator vector in $\{0,1\}^r$ and an $s \in \mathbb{Z}_r$, just output the ciphertext in the position corresponding to s.

Since our bootstrapping scheme uses $\mathbb{Z}_r$ embeddings only for $r = O(\log \lambda)$, these optimizations lead to polylogarithmic factor improvements in runtime and error, but no more.

Signed and generalized permutations. We can also use the GSW scheme to obtain a homomorphic encryption scheme for the group of *signed* permutations (also known as the *signed symmetric group* or *hyperoctahedral group*), by encrypting signed permutation matrices. (A signed permutation matrix is one in which every row and column has exactly one nonzero entry, which may be $+1$ or -1.) Even more generally, when using a ring-LWE-based GSW scheme over the mth cyclotomic ring, we can get homomorphic encryption for the *generalized symmetric group* $\mathbb{Z}_m \wr S_r$, by encrypting generalized permutation matrices whose nonzero entries are mth roots of unity. All our analysis goes through essentially unchanged for these cases, since we only rely on the fact that the nonzero entries of the encrypted matrices have magnitude one.

Generalized symmetric groups contain somewhat larger cyclic groups than symmetric groups do, so they can be used as an optimization by letting us use slightly smaller orders r. However, the overall difference does not appear to be too significant.

3.5 Bootstrapping

We now describe our bootstrapping procedure.

3.5.1 Specification and Usage

We start by specifying the abstract preconditions and output guarantees of our bootstrapping algorithm, and describe how to use it (with some additional pre- and post-processing) to bootstrap known LWE-based encryption schemes.

The scheme to be bootstrapped must have *binary* ciphertexts in $\{0,1\}^d$ and secret keys in $\mathbb{Z}_q^d$ for some dimension d and modulus q that should be made as small as possible ($q, d = \tilde{O}(\lambda)$ are possible), and a decryption function of the form $\mathsf{Dec}_{\mathbf{s}}(\mathbf{c}) = f(\langle \mathbf{s}, \mathbf{c} \rangle) \in \{0,1\}$

for some arbitrary function $f\colon \mathbb{Z}_q \to \{0,1\}$. We rely on an appropriate instantiation of the GSW cryptosystem, as described in further detail in Section 3.5.2 below.

$\mathsf{BootGen}(\mathbf{s} \in \mathbb{Z}_q^d, sk)$ takes as input a secret key vector $\mathbf{s} \in \mathbb{Z}_q^d$ from the scheme to be bootstrapped, and a secret key sk for GSW. It outputs a bootstrapping key bk, which appropriately encrypts $\mathbf{s}$ under sk.

$\mathsf{Bootstrap}(bk, \mathbf{c} \in \{0,1\}^d)$ takes as input the bootstrapping key bk and a ciphertext vector $\mathbf{c} \in \{0,1\}^d$ (which decrypts under the secret key $\mathbf{s}$). It outputs a GSW ciphertext which decrypts (under sk) to the same bit as $\mathbf{c}$ does (under $\mathbf{s}$), but with less error.

Pre- and post-processing. We can bootstrap all known LWE-based bit-encryption schemes using the above algorithms as follows. In all LWE-based encryption schemes, decryption can be expressed as a "rounded inner product" $\lfloor \langle \mathbf{s}, \mathbf{c} \rangle \rceil_2$ for some appropriate rounding function $\lfloor \cdot \rceil_2 \colon \mathbb{Z}_q \to \{0,1\}$, as required. Note that a GSW ciphertext can trivially be put in this form by just taking its penultimate column (see GSW.Dec in Section 3.2.2). As for the other conditions we need (binary ciphertexts and small d, q), LWE encryption schemes are not always presented in a way that fulfills them, but fortunately there are standard transformations that do so, as we now describe. (See [23, 22] for further details.)

First, since we do not need to perform any further homomorphic operations on the ciphertext, we can use dimension- and modulus-reduction [23] to get a ciphertext $\bar{\mathbf{c}}$ (over $\mathbb{Z}_q$) of dimension $\tilde{O}(\lambda)$ and modulus $q = \tilde{O}(\lambda)$, while preserving correct decryption. These steps can be implemented with 2^λ security under conventional lattice assumptions.[7]

In somewhat more detail, modulus-switching is used to reduce the size of the ciphertext modulus. It was originally introduced to reduce the size and depth of the decryption circuit, and has been used for several other purposes as well in more recent papers [20, 25]. We recall the main result, that if we switch from a modulus q to a modulus $p < q$ by scaling by

[7] To make the modulus quasi-linear, we need to use randomized (subgaussian) rounding in the modulus-reduction step.

p/q and then rounding, the noise in the decryption relation increases by at most $\|\mathbf{s}\|_1$ (the ℓ_1 norm of the secret key $\mathbf{s}$). Formally,

$$|\langle \lfloor \tfrac{p}{q}\mathbf{c} \rceil, \mathbf{s}\rangle - \tfrac{p}{q}\langle \mathbf{c}, \mathbf{s}\rangle| \leq \|\mathbf{s}\|_1,$$

where $\mathbf{c}$ and $\mathbf{s}$ are viewed as integers.

Then, we can obtain a binary ciphertext $\mathbf{c}$ using "bit decomposition:" let $\mathbf{G}$ be as defined in Section 3.2.2, and for the ciphertext $\bar{\mathbf{c}}$ over $\mathbb{Z}_q$ under secret key $\bar{\mathbf{s}}$, let $\mathbf{c}$ be a $\{0,1\}$-vector such that $\mathbf{G}\mathbf{c} = \bar{\mathbf{c}}$, and let $\mathbf{s} = \mathbf{G}^t\bar{\mathbf{s}}$ so that $\langle \mathbf{s}, \mathbf{c}\rangle = \langle \bar{\mathbf{s}}, \bar{\mathbf{c}}\rangle \in \mathbb{Z}_q$. (The secret key $\mathbf{s}$ is therefore the one we need to provide to BootGen.)

After bootstrapping, the output is a GSW ciphertext $\mathbf{C}$ encrypted under sk (which is just an integer vector). If desired, we can convert this ciphertext back to one for the original LWE cryptosystem, simply by taking the penultimate column of $\mathbf{C}$. We can also key-switch from sk back to the original secret key $\mathbf{s}$. (As usual in bootstrapping, going "full circle" in this way requires an appropriate circular security assumption.)

3.5.2 Procedures

Our algorithms rely on instantiations of GSW and HEPerm with parameters n, Q, χ. Importantly, the ciphertext modulus Q is *not* the modulus q of the scheme we are bootstrapping, but rather some $Q \gg q$ that is sufficiently larger than the error in Bootstrap's output ciphertext. Let $\mathcal{C}$ denote the GSW ciphertext space.

Our procedures need q to be of the form $q = \prod_{i\in[t]} r_i$ where the r_i are small and powers of distinct primes (and hence pairwise coprime). Specifically, using Lemma 3.3.1 we can choose $q = \tilde{O}(\lambda)$ to be large enough by letting it be the product of all maximal prime-powers r_i that are bounded by $O(\log \lambda)$, of which there are $t = O(\log \lambda / \log\log \lambda)$. Let ϕ be the group embedding of $(\mathbb{Z}_q, +) \cong (\mathbb{Z}_{r_1} \times \cdots \times \mathbb{Z}_{r_t}, +)$ into $S = S_{r_1} \times \cdots \times S_{r_t}$ described in Section 3.3, and let ϕ_i denote the ith component of this embedding, i.e., the one from $\mathbb{Z}_q$ into S_{r_i}.

$\mathsf{BootGen}(\mathbf{s} \in \mathbb{Z}_q^d, sk)$**:** given secret key $\mathbf{s} \in \mathbb{Z}_q^d$ for the scheme to be bootstrapped and a secret key sk for HEPerm, embed each coordinate $s_j \in \mathbb{Z}_q$ of $\mathbf{s}$ as $\phi(s_j) \in S$ and encrypt the components under HEPerm. That is, generate and output the bootstrapping key

$$bk = \{\mathbf{C}_{i,j} \leftarrow \text{HEPerm.}\mathsf{Enc}(sk, \phi_i(s_j)) : i \in [t], j \in [d]\}.$$

Recalling that we are working with embeddings of $\mathbb{Z}_{r_i}$, each $\mathbf{C}_{i,j} \in \mathcal{C}^{r_i}$ can be represented as a tuple of r_i GSW ciphertexts encrypting an indicator vector (see Section 3.4.3). Because $t, r_i = O(\log \lambda)$ and $d = \tilde{O}(\lambda)$, the bootstrapping key consists of $\tilde{O}(\lambda)$ GSW ciphertexts.

$\mathsf{Bootstrap}(bk, \mathbf{c} \in \{0,1\}^d)$**:** given a binary ciphertext $\mathbf{c} \in \{0,1\}^d$, do the following:

Inner Product: Homomorphically compute an encryption of

$$v = \langle \mathbf{s}, \mathbf{c} \rangle = \sum_{j\,:\,c_j = 1} s_j \in \mathbb{Z}_q$$

using the encryptions of the $s_j \in \mathbb{Z}_q$ as embedded into the permutation group S, via a chain of compositions. Formally, for each $i \in [t]$ compute (recalling that $\boxdot$ is right associative, and $\mathbf{J}$ is the fixed HEPerm encryption of the identity permutation)

$$\mathbf{C}_i \leftarrow \bigboxdot_{j \text{ s.t. } c_j = 1} \mathbf{C}_{i,j} \boxdot \mathbf{J}. \tag{3.5.1}$$

Again, because we are working with embeddings of $\mathbb{Z}_{r_i}$, each $\mathbf{C}_i \in \mathcal{C}^{r_i}$.

Round: Homomorphically map $v \in \mathbb{Z}_q$ to $f(v) \in \mathbb{Z}_2 = \{0,1\}$: for each $x \in \mathbb{Z}_q$ such that $f(x) = 1$, homomorphically test whether $v \stackrel{?}{=} x$ by homomorphically multiplying the GSW ciphertexts resulting from all the equality tests $v \stackrel{?}{=} x \pmod{r_i}$. Then homomorphically sum the results of all the $v \stackrel{?}{=} x$ tests.

Formally, compute and output the GSW ciphertext (recalling that $\boxdot$ is right associative, and $\mathbf{G}$ is the fixed GSW encryption of 1)

$$\mathbf{C} \leftarrow \bigboxplus_{x \in \mathbb{Z}_q \text{ s.t. } f(x)=1} \Big(\bigboxdot_{i \in [t]} \mathsf{Eq?}(\mathbf{C}_i, \phi_i(x)) \boxdot \mathbf{G} \Big). \tag{3.5.2}$$

Note that since we are working with embeddings of $\mathbb{Z}_{r_i}$, each $\mathsf{Eq}?(\mathbf{C}_i, \phi_i(x))$ is just some GSW ciphertext component of $\mathbf{C}_i \in \mathcal{C}^{r_i}$ (see Section 3.4.3).

Because $t, r_i = O(\log \lambda)$ and $d = \tilde{O}(\lambda)$ and by Equations (3.5.1) and (3.5.2), $\mathsf{Bootstrap}$ performs $\tilde{O}(\lambda)$ homomorphic multiplications and additions on GSW ciphertexts.

3.5.3 Analysis

The following is our main theorem.

Theorem 3.5.1. *The above bootstrapping scheme can be instantiated to be correct (with overwhelming probability) and secure assuming that the decisional Shortest Vector Problem (GapSVP) and Shortest Independent Vectors Problem (SIVP) are (quantumly) hard to approximate in the worst case to within $\tilde{O}(n^2\lambda)$ factors on n-dimensional lattices.*

Because all known (quantum) algorithms for $\mathrm{poly}(n)$-factor approximations to GapSVP and SIVP on n-dimensional lattices take $2^{\Omega(n)}$ time, for 2^λ hardness we can take $n = \Theta(\lambda)$, yielding a final approximation factor of $\tilde{O}(n^3)$. This comes quite close to the $O(n^{3/2+\epsilon})$ factors obtained in [25], but *without* any expensive "dimension leveraging:" we use GSW ciphertexts of dimension only $n = O(\lambda)$, rather than some large polynomial in λ. Alternatively, at the cost of a larger dimension $n = \lambda^{1/\epsilon}$, but without using the successive dimension-reduction procedure from [25], we can obtain factors as small as $\tilde{O}(n^{2+\epsilon})$ for any constant $\epsilon > 0$.

The remainder of this subsection is devoted to proving the above theorem.

Security. If the HEPerm key sk is generated independently of $\mathbf{s}$, then IND-CPA security of the bootstrapping key follows immediately from the security of HEPerm, hence from LWE with parameters $n - 1, Q, \chi$, and finally from worst-case lattice problems. (We instantiate these parameters below to obtain the claimed approximation factors.) As usual, if the keys

are not independent, then we need to make an appropriate circular security assumption. (To date, such an assumption is the only known way to obtain unbounded FHE.)

Correctness and error analysis. For correctness, we first show that the ciphertext $\mathbf{C}$ output by $\mathsf{Bootstrap}$ is designed to encrypt the appropriate bit. Then we quantify the error in $\mathbf{C}$ and instantiate the parameters so that it indeed decrypts to the intended bit.

Lemma 3.5.2 (Correctness). *For $bk \leftarrow \mathsf{BootGen}(\mathbf{s}, sk)$, the GSW ciphertext $\mathbf{C} \leftarrow \mathsf{Bootstrap}(bk, \mathbf{c})$ is designed to encrypt $\mathsf{Dec}_{\mathbf{s}}(\mathbf{c}) = f(\langle \mathbf{s}, \mathbf{c} \rangle) \in \{0, 1\}$.*

Proof. First, by construction the HEPerm ciphertext $\mathbf{C}_{i,j}$ is designed to encrypt $\phi_i(s_j)$. Therefore, because $\phi_i \colon \mathbb{Z}_q \to S_{r_i}$ is a group homomorphism, the ciphertext $\mathbf{C}_i$ as defined in Equation (3.5.1) is designed to encrypt $\phi_i(\sum_{j \,:\, c_j = 1} s_j) = \phi_i(\langle \mathbf{s}, \mathbf{c} \rangle) = \phi_i(v)$. By correctness of $\mathsf{Eq}?$ and the isomorphism $\mathbb{Z}_q \cong \mathbb{Z}_{r_1} \times \cdots \times \mathbb{Z}_{r_t}$ given by Chinese Remainder Theorem, the homomorphic product $\boxdot_{i \in [t]} \mathsf{Eq}?(\mathbf{C}_i, \phi_i(x)) \boxdot \mathbf{G}$ is designed to encrypt 1 if and only if $v = x$. Finally, because the homomorphic sum is taken over every $x \in \mathbb{Z}_q$ such that $f(x) = 1$, it is designed to encrypt 1 if and only if $f(v) = 1$. □

We now quantify the error in the ciphertext output by $\mathsf{Bootstrap}$. Recall that GSW and HEPerm are parameterized by a dimension n, a modulus Q with $\ell = \lceil \log_2 Q \rceil$, and an error distribution χ over $\mathbb{Z}$ that is subgaussian with parameter s, where typically $s = \Theta(\sqrt{n})$. Let $r = \sum_{i \in [t]} r_i$ be the sum of the maximal prime-power divisors r_i of q, and recall that each $r_i = O(\log \lambda)$.

Lemma 3.5.3. *For any $\mathbf{c} \in \{0,1\}^d$, the error vector in the refreshed ciphertext $\mathbf{C} \leftarrow \mathsf{Bootstrap}(bk, \mathbf{c})$ has independent subgaussian entries with parameter $O(sn\ell\sqrt{rdq}) = \tilde{O}(sn\ell\lambda)$, except with probability $2^{-\Omega(n\ell)}$ over the random choices of bk and $\mathsf{Bootstrap}$.*

By Claim 3.2.5, the ciphertext $\mathbf{C}$ therefore decrypts correctly (except with $\mathrm{negl}(\lambda)$ probability) as long as the modulus Q of the GSW system is at least $sn\ell\sqrt{rdq} \cdot \omega(\sqrt{\log \lambda})$.

Proof. Recall that the GSW ciphertext and error spaces are respectively $\mathcal{C} = \mathbb{Z}_Q^{n \times n\ell}$ and $\mathcal{E} = \mathbb{Z}^{n\ell}$, and the HEPerm ciphertext and error spaces for the embedding of $\mathbb{Z}_{r_i}$ into the symmetric group S_{r_i} are $\mathcal{C}^{r_i}$ and $\mathcal{E}^{r_i}$, respectively. To perform homomorphic composition, we take cyclic rotations to get ciphertexts and error matrices in $\mathcal{C}^{r_i \times r_i}$ and $\mathcal{E}^{r_i \times r_i}$.

We analyze the error in the various ciphertexts $\mathbf{C}_{i,j}, \mathbf{C}_i, \mathbf{C}$ produced by BootGen and Bootstrap. Essentially, this proceeds by a couple of invocations of Lemma 3.2.1 (which bounds the ℓ_2 norm of a vector having independent subgaussian entries) and Corollaries 3.2.7 and 3.4.2 (which guarantee fresh subgaussian errors in a homomorphic chain of multiplications/compositions). Specifically:

- The error vector in a fresh GSW ciphertext has independent subgaussian entries with parameter s, so by Lemma 3.2.1, its ℓ_2 norm is $O(s\sqrt{n\ell})$, except with probability $2^{-\Omega(n\ell)}$. Therefore, in the concatenation of the rotation-expanded error matrices of $\mathbf{C}_{i,j}$ over any subset of $j \in [d]$, every row has ℓ_2 norm $O(s\sqrt{r_i n\ell d})$.

- By Corollary 3.4.2, all the $\mathbb{Z}$-entries in all the error matrices $\mathbf{E}_i \in \mathcal{E}^{r_i}$ for $\mathbf{C}_i$ (see Equation (3.5.1)) are mutually independent, and are subgaussian with parameter $O(s\sqrt{r_i n\ell d})$. By Lemma 3.2.1, it follows that any single $\mathcal{E}$-entry of $\mathbf{E}_i$ has ℓ_2 norm $O(sn\ell\sqrt{r_i d})$ except with probability $2^{-\Omega(n\ell)}$, and hence their concatenation over all $i \in [t]$ has ℓ_2 norm $O(sn\ell\sqrt{rd})$.

- By the above and Corollary 3.2.7, each GSW ciphertext produced inside the parenthesized expression of Equation (3.5.2) has a fresh error vector with independent subgaussian entries with parameter $O(sn\ell\sqrt{rd})$. Finally, by Pythagorean additivity of independent subgaussians, the error vector of $\mathbf{C}$ has independent subgaussian entries with parameter $O(sn\ell\sqrt{rdq})$, as claimed. □

Instantiating the parameters. We now instantiate all the parameters to finish the proof of Theorem 3.5.1. To rely on the (quantum) worst-case hardness of LWE [75], we take

$s = 3\sqrt{n} = \Theta(\sqrt{n})$. Then by Lemma 3.5.3, we simply need to take a sufficiently large $Q = \tilde{\Omega}(n^{3/2}\lambda \log Q)$; some $Q = \tilde{O}(n^{3/2}\lambda)$ suffices. The LWE inverse error rate is therefore $Q/s = \tilde{O}(n\lambda)$, yielding an approximation factor of $\tilde{O}(n^2\lambda)$ for worst-case lattice problems in dimension n. Slightly worse factors can be obtained by relying on *classical* reductions for the hardness of LWE [74, 22].

REFERENCES

[1] ALPERIN-SHERIFF, J., "Short signatures with short public keys from homomorphic trapdoor functions," in *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pp. 236–255, 2015.

[2] ALPERIN-SHERIFF, J. and PEIKERT, C., "Circular and KDM security for identity-based encryption," in *Public Key Cryptography*, pp. 334–352, 2012.

[3] ALPERIN-SHERIFF, J. and PEIKERT, C., "Practical bootstrapping in quasilinear time," in *CRYPTO (1)*, pp. 1–20, 2013.

[4] ALPERIN-SHERIFF, J. and PEIKERT, C., "Faster bootstrapping with polynomial error," in *CRYPTO (1)*, pp. 297–314, 2014.

[5] APPLEBAUM, B., CASH, D., PEIKERT, C., and SAHAI, A., "Fast cryptographic primitives and circular-secure encryption based on hard learning problems," in *CRYPTO*, pp. 595–618, 2009.

[6] ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R. H., KONWINSKI, A., LEE, G., PATTERSON, D. A., RABKIN, A., STOICA, I., and ZAHARIA, M., "Above the clouds: A berkeley view of cloud computing," Technical Report UCB/EECS-2009-28, University of California at Berkeley, 2009.

[7] BABAI, L., "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986. Preliminary version in STACS 1985.

[8] BANERJEE, A., PEIKERT, C., and ROSEN, A., "Pseudorandom functions and lattices," in *EUROCRYPT*, pp. 719–737, 2012.

[9] BARKLEY ROSSER, J. and SCHOENFELD, L., "Approximate formulas for some functions of prime numbers," *Illinois J. Math*, vol. 6, pp. 64–94, 1962.

[10] BARRINGTON, D. A., "Bounded-width polynomial-size branching programs recognize exactly those languages in nc 1," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pp. 1–5, ACM, 1986.

[11] BARRINGTON, D. A. M., "Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1," in *STOC*, pp. 1–5, 1986.

[12] BARRINGTON, D. A. M., BEIGEL, R., and RUDICH, S., "Representing boolean functions as polynomials modulo composite numbers (extended abstract)," in *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pp. 455–461, 1992.

[13] BENALOH, J. C., "Secret sharing homomorphisms: Keeping shares of A secret sharing," in *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pp. 251–260, 1986.

[14] BLACK, J., ROGAWAY, P., and SHRIMPTON, T., "Encryption-scheme security in the presence of key-dependent messages," in *Selected Areas in Cryptography*, pp. 62–75, 2002.

[15] BONEH, D., GENTRY, C., GORBUNOV, S., HALEVI, S., NIKOLAENKO, V., SEGEV, G., VAIKUNTANATHAN, V., and VINAYAGAMURTHY, D., "Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits," in *EUROCRYPT*, pp. 533–556, 2014.

[16] BONEH, D., HALEVI, S., HAMBURG, M., and OSTROVSKY, R., "Circular-secure encryption from decision Diffie-Hellman," in *CRYPTO*, pp. 108–125, 2008.

[17] BONEH, D., WU, D. J., and ZIMMERMAN, J., "Immunizing multilinear maps against zeroizing attacks." Cryptology ePrint Archive, Report 2014/930, 2014. `http://eprint.iacr.org/`.

[18] BRAKERSKI, Z., "Fully homomorphic encryption without modulus switching from classical GapSVP," in *CRYPTO*, pp. 868–886, 2012.

[19] BRAKERSKI, Z., "When homomorphism becomes a liability," in *TCC*, pp. 143–161, 2013.

[20] BRAKERSKI, Z., GENTRY, C., and VAIKUNTANATHAN, V., "(Leveled) fully homomorphic encryption without bootstrapping," in *ICTS*, pp. 309–325, 2012.

[21] BRAKERSKI, Z. and GOLDWASSER, S., "Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back)," in *CRYPTO*, pp. 1–20, 2010.

[22] BRAKERSKI, Z., LANGLOIS, A., PEIKERT, C., REGEV, O., and STEHLÉ, D., "Classical hardness of learning with errors," in *STOC*, pp. 575–584, 2013.

[23] BRAKERSKI, Z. and VAIKUNTANATHAN, V., "Efficient fully homomorphic encryption from (standard) LWE," in *FOCS*, pp. 97–106, 2011.

[24] BRAKERSKI, Z. and VAIKUNTANATHAN, V., "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *CRYPTO*, pp. 505–524, 2011.

[25] BRAKERSKI, Z. and VAIKUNTANATHAN, V., "Lattice-based FHE as secure as PKE," in *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pp. 1–12, 2014.

[26] BRENNER, M., WIEBELITZ, J., VON VOIGT, G., and SMITH, M., "Secret program execution in the cloud applying homomorphic encryption," in *Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on*, pp. 114–119, IEEE, 2011.

[27] BRICKELL, E. F. and YACOBI, Y., "On privacy homomorphisms (extended abstract)," in *Advances in Cryptology - EUROCRYPT '87, Workshop on the Theory and Application of of Cryptographic Techniques, Amsterdam, The Netherlands, April 13-15, 1987, Proceedings*, pp. 117–125, 1987.

[28] CAI, J.-Y. and LIPTON, R., "Subquadratic simulations of circuits by branching programs," *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, vol. 0, pp. 568–573, 1989.

[29] CANETTI, R., LIN, H., TESSARO, S., and VAIKUNTANATHAN, V., "Obfuscation of probabilistic circuits and applications." Cryptology ePrint Archive, Report 2014/882, 2014. `http://eprint.iacr.org/`.

[30] CASH, D., GREEN, M., and HOHENBERGER, S., "New definitions and separations for circular security," in *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, pp. 540–557, 2012.

[31] CHEON, J. H., HAN, K., LEE, C., RYU, H., and STEHLE, D., "Cryptanalysis of the multilinear map over the integers." Cryptology ePrint Archive, Report 2014/906, 2014. `http://eprint.iacr.org/`.

[32] CHEON, J. H., KIM, W., and NAM, H. S., "Known-plaintext cryptanalysis of the domingo-ferrer algebraic privacy homomorphism scheme," *Inf. Process. Lett.*, vol. 97, no. 3, pp. 118–123, 2006.

[33] CLEVE, R., "Towards optimal simulations of formulas by bounded-width programs," *Computational Complexity*, vol. 1, no. 1, pp. 91–105, 1991.

[34] COHEN, J. D. and FISCHER, M. J., "A robust and verifiable cryptographically secure election scheme (extended abstract)," in *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pp. 372–382, 1985.

[35] CORON, J.-S., LEPOINT, T., and TIBOUCHI, M., "Cryptanalysis of two candidate fixes of multilinear maps over the integers." Cryptology ePrint Archive, Report 2014/975, 2014. `http://eprint.iacr.org/`.

[36] DOMINGO-FERRER, J., "A new privacy homomorphism and applications," *Inf. Process. Lett.*, vol. 60, no. 5, pp. 277–282, 1996.

[37] DOMINGO-FERRER, J., "A provably secure additive and multiplicative privacy homomorphism," in *Information Security, 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30 - October 2, 2002, Proceedings*, pp. 471–483, 2002.

[38] DUCAS, L. and MICCIANCIO, D., "FHEW: bootstrapping homomorphic encryption in less than a second," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pp. 617–640, 2015.

[39] DUSART, P., "The k th prime is greater than k (ln k+ ln ln k-1) for k 2," *Mathematics of Computation*, pp. 411–415, 1999.

[40] FONTAINE, C. and GALAND, F., "A survey of homomorphic encryption for nonspecialists," *EURASIP Journal on Information Security*, vol. 2007, p. 15, 2007.

[41] FRÖHLICH, A. and TAYLOR, M., *Algebraic Number Theory*. Cambridge University Press, 1991.

[42] GARG, S., GENTRY, C., HALEVI, S., RAYKOVA, M., SAHAI, A., and WATERS, B., "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *FOCS*, pp. 40–49, 2013.

[43] GENTRY, C., *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

[44] GENTRY, C., "Fully homomorphic encryption using ideal lattices," in *STOC*, pp. 169–178, 2009.

[45] GENTRY, C., "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, pp. 97–105, Mar. 2010.

[46] GENTRY, C. and HALEVI, S., "Fully homomorphic encryption without squashing using depth-3 arithmetic circuits," in *FOCS*, pp. 107–109, 2011.

[47] GENTRY, C. and HALEVI, S., "Implementing Gentry's fully-homomorphic encryption scheme," in *EUROCRYPT*, pp. 129–148, 2011.

[48] GENTRY, C., HALEVI, S., MAJI, H. K., and SAHAI, A., "Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero." Cryptology ePrint Archive, Report 2014/929, 2014. `http://eprint.iacr.org/`.

[49] GENTRY, C., HALEVI, S., PEIKERT, C., and SMART, N. P., "Ring switching in BGV-style homomorphic encryption," in *SCN*, pp. 19–37, 2012. Full version at `http://eprint.iacr.org/2012/240`.

[50] GENTRY, C., HALEVI, S., and SMART, N. P., "Better bootstrapping in fully homomorphic encryption," in *Public Key Cryptography*, pp. 1–16, 2012.

[51] GENTRY, C., HALEVI, S., and SMART, N. P., "Fully homomorphic encryption with polylog overhead," in *EUROCRYPT*, pp. 465–482, 2012.

[52] GENTRY, C., HALEVI, S., and SMART, N. P., "Homomorphic evaluation of the AES circuit," in *CRYPTO*, pp. 850–867, 2012.

[53] GENTRY, C., SAHAI, A., and WATERS, B., "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *CRYPTO (1)*, pp. 75–92, 2013.

[54] GOLDWASSER, S. and MICALI, S., "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984. Preliminary version in STOC 1982.

[55] GORBUNOV, S., VAIKUNTANATHAN, V., and WICHS, D., "Leveled fully homomorphic signatures from standard lattices." Cryptology ePrint Archive, Report 2014/897, 2014. `http://eprint.iacr.org/`.

[56] GUY, R., *Unsolved problems in number theory*, vol. 1. Springer, 2004.

[57] HAITNER, I. and HOLENSTEIN, T., "On the (im)possibility of key dependent encryption," in *TCC*, pp. 202–219, 2009.

[58] HALEVI, S. and SHOUP, V., "Algorithms in helib," in *CRYPTO (1)*, pp. 554–571, 2014.

[59] HALEVI, S. and SHOUP, V., "Bootstrapping for helib," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pp. 641–670, 2015.

[60] HIROMASA, R., ABE, M., and OKAMOTO, T., "Packing messages and optimizing bootstrapping in GSW-FHE," in *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pp. 699–715, 2015.

[61] HIRT, M. and SAKO, K., "Efficient receipt-free voting based on homomorphic encryption," in *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pp. 539–556, 2000.

[62] JACOBSON, N., *Basic Algebra I*. Dover Publications, 2012.

[63] KEARNS, M. J. and VALIANT, L. G., "Cryptographic limitations on learning boolean formulae and finite automata," *J. ACM*, vol. 41, no. 1, pp. 67–95, 1994.

[64] KIAYIAS, A. and YUNG, M., "The vector-ballot e-voting approach," in *Financial Cryptography, 8th International Conference, FC 2004, Key West, FL, USA, February 9-12, 2004. Revised Papers*, pp. 72–89, 2004.

[65] KOPPULA, V., RAMCHEN, K., and WATERS, B., "Separations in circular security for arbitrary length key cycles," *IACR Cryptology ePrint Archive*, vol. 2013, p. 683, 2013.

[66] KUSHILEVITZ, E. and OSTROVSKY, R., "Replication is NOT needed: SINGLE database, computationally-private information retrieval," in *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pp. 364–373, 1997.

[67] LYUBASHEVSKY, V., PEIKERT, C., and REGEV, O., "On ideal lattices and learning with errors over rings," in *EUROCRYPT*, pp. 1–23, 2010.

[68] LYUBASHEVSKY, V., PEIKERT, C., and REGEV, O., "On ideal lattices and learning with errors over rings," *J. ACM*, 2013. To appear. Preliminary version in Eurocrypt 2010.

[69] LYUBASHEVSKY, V., PEIKERT, C., and REGEV, O., "A toolkit for ring-LWE cryptography," in *EUROCRYPT*, pp. 35–54, 2013.

[70] MELL, P. and GRANCE, T., "The nist definition of cloud computing," tech. rep., Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, 2011.

[71] MICCIANCIO, D. and PEIKERT, C., "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *EUROCRYPT*, pp. 700–718, 2012.

[72] MURATA, L. and POMERANCE, C., "On the largest prime factor of a mersenne number," in *Number theory, CRM Proc. Lecture Notes of American Mathematical Society*, vol. 36, pp. 209–218, 2004.

[73] ORSINI, E., VAN DE POL, J., and SMART, N. P., "Bootstrapping BGV ciphertexts with a wider choice of p and q," in *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pp. 673–698, 2015.

[74] PEIKERT, C., "Public-key cryptosystems from the worst-case shortest vector problem," in *STOC*, pp. 333–342, 2009.

[75] REGEV, O., "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, 2009. Preliminary version in STOC 2005.

[76] RIVEST, R. L., ADLEMAN, L., and DERTOUZOS, M. L., "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[77] ROHLOFF, K. and COUSINS, D., "A scalable implementation of fully homomorphic encryption built on ntru," in *Financial Cryptography and Data Security* (BHME, R., BRENNER, M., MOORE, T., and SMITH, M., eds.), Lecture Notes in Computer Science, pp. 221–234, Springer Berlin Heidelberg, 2014.

[78] ROSEN, K. H., *Elementary number theory and its applications*. Reading, Mass., 1993.

[79] ROTHBLUM, R., "On the circular security of bit-encryption," in *TCC*, pp. 579–598, 2013.

[80] SCHOENFELD, L., "Sharper bounds for the Chebyshev functions $\theta(x)$ and $\psi(x)$. ii," *Mathematics of Computation*, vol. 30, no. 134, pp. pp. 337–360, 1976.

[81] SINHA, R. K., *Some topics in parallel computation and branching programs*. PhD thesis, University of Washington, 1995.

[82] SMART, N. and VERCAUTEREN, F., "Fully homomorphic SIMD operations." Cryptology ePrint Archive, Report 2011/133, 2011. http://eprint.iacr.org/.

[83] STEWART, C. L., "On divisors of lucas and lehmer numbers," *Acta Mathematica*, vol. 211, no. 2, pp. 291–314, 2013.

[84] VAN DIJK, M., GENTRY, C., HALEVI, S., and VAIKUNTANATHAN, V., "Fully homomorphic encryption over the integers," in *EUROCRYPT*, pp. 24–43, 2010.

[85] VERSHYNIN, R., *Compressed Sensing, Theory and Applications*, ch. 5, pp. 210–268. Cambridge University Press, 2012. Available at http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf.

[86] WAGNER, D., "Cryptanalysis of an algebraic privacy homomorphism," in *Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 1-3, 2003, Proceedings*, pp. 234–239, 2003.

[87] YI, X., KAOSAR, M. G., PAULET, R., and BERTINO, E., "Single-database private information retrieval from fully homomorphic encryption," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 5, pp. 1125–1134, 2013.