High-Precision Exact FHE Made Simple, General, and Fast

Chris Peikert[1], Doron Zarchy[2], and Guy Zyskind[3]

[1]*University of Michigan and Fhenix,* cpeikert@umich.edu.
[2]*Fhenix,* doron@fhenix.io
[3]*Fhenix and University of Miami,* guy@fhenix.io

August 14, 2026

Abstract

Many important applications of fully homomorphic encryption (FHE) require arithmetic on *high-precision* plaintexts, e.g., from the ring $\mathbb{Z}_p$ for a huge prime or power-of-two modulus p. The classic FHE schemes are poorly suited to this, because the inverse error rate of fresh ciphertexts, and the error growth under homomorphic multiplication, are both larger than p, which results in large and inefficient parameters. While there are now several works addressing this problem, the landscape for *exact* (as opposed to approximate) FHE is highly fragmented: known solutions either work only for certain rare plaintext moduli having very special forms (sometimes using non-standard ciphertext rings that lack other important features for FHE), or have quite complicated and high-latency constructions.

This work gives a very simple, general, and efficient technique for high-precision exact FHE, in which the error rates and growth match those of classic schemes for *exponentially smaller* precision. The runtimes can scale only *quasi-linearly* (versus quadratically for classic schemes) with the plaintext precision $\log p$, and are fast in practice. Also in contrast to all prior works, our technique works for *any integer modulus* and over *any underlying (number) ring*—or even with no structured ring at all, making it the first solution that can be based on plain LWE. Moreover, it is *fully compatible with prior FHE techniques* for fast ring arithmetic, plaintext packing and SIMD operations, bootstrapping, etc. For typical parameters and security levels, our (preliminary, unoptimized, single-threaded) implementation does homomorphic $\mathbb{Z}_{2^{64}}$-multiplication in just tens of milliseconds, and obtains a four- to five-fold increase in multiplicative depth versus classic FHE schemes.

1 Introduction

Fully homomorphic encryption (FHE) [RAD78] is a "holy grail" of cryptography, allowing an untrusted server to perform arbitrary computations on a client's encrypted data. Since Gentry's groundbreaking work [Gen09b, Gen09a] showing the first plausible FHE scheme, researchers have made enormous strides (e.g., [BV11a, SV11, GHS12b, BV11b, BGV12, Bra12, GSW13, CKKS17]) in improving the simplicity, theoretical and practical efficiency, and utility of FHE schemes, and the hardness assumptions on which they are based—mainly the LWE problem [Reg05] and its more efficient variants like Ring- and Module-LWE [LPR10, BGV12, LS15]. They have also made huge improvements (e.g., [GHS12a, AP13, AP14, DM15, HS15, CGGI16, CCS19]) in a key bottleneck called *bootstrapping*, which enables arbitrarily complex

homomorphic computations by "shrinking" the intrinsic error in ciphertexts before it grows too large and causes incorrect decryption.

1.1 High-Precision FHE

From the early days of fully homomorphic encryption [NLV11], and especially in recent years [GC14, CLPX18, CCKS23, GV25b, BK25, Kim25b, Kim26a, CPL26, BFG+26], there has been strong interest in supporting efficient homomorphic computation on *high-precision* plaintexts. For "exact" schemes like BGV [BGV12], BFV [Bra12, FV12], or GSW/FHEW/TFHE [GSW13, DM15, CGGI16] that support finite plaintext rings like $\mathbb{Z}_p$, "high precision" means that p is very large—say, 64 bits or more. For "approximate" schemes like CKKS [CKKS17], in which the plaintext ring is $\mathbb{R}$ or $\mathbb{C}$, it means that the homomorphic operations are inexact but accurate to many bits of precision, i.e., $\log p$ bits for a fair comparison with exact schemes.

High-precision homomorphic computation is very well motivated by many privacy applications:

- Algorithms that work with real numbers, like neural-net training and inference, could be run homomorphically by an "encrypted floating-point unit" (eFPU) that supports real numbers with sufficient accuracy.
- In the realm of exact arithmetic, some advanced cryptographic constructions (e.g., [BGG+18]) require *homomorphically* executing lower-level cryptographic primitives, which tend to involve huge (prime) finite fields or composite rings, with hundreds of bits of precision.
- Applications like private cryptocurrencies may need to store and compare large balances, which can be represented as integers modulo some huge integer modulus.
- In general, an "encrypted arithmetic-logic unit" (eALU) for (say) 64-bit registers would allow for direct homomorphic execution of the mod-2^{64} computations done by commodity hardware.

High precision in classic FHE. Unfortunately, the classic FHE schemes are not well suited to high-precision arithmetic, for various reasons. For "second-generation" schemes like BGV/BFV, this is because with each homomorphic multiplication, the intrinsic *error rate* of the ciphertexts grows by a factor of at least the plaintext characteristic p (along with lower-order factors). Since the error rate must be kept below 1 for correct decryption, ciphertexts that will undergo homomorphic computations of multiplicative depth h must have very small error rates, of less than p^{-h}. Ensuring security for such tiny error rates requires these ciphertexts to have correspondingly large dimensions and integer precision, which significantly hurts efficiency. Quantitatively, error rate α requires both the dimension and integer bit length to be proportional to at least $\log(1/\alpha)$, so the ciphertext sizes and running times are $\Omega(h^2 \log^2 p)$. Concretely, these asymptotics manifest quite severely for real parameters in practice.

"Third-generation" schemes like GSW/TFHE natively support only small plaintext rings, so high-precision arithmetic must be implemented via arithmetic circuits over such rings, like $\mathbb{Z}_2 = \{0, 1\}$. Thus, the complexity of a single homomorphic $\mathbb{Z}_p$-multiplication (counting just small-ring homomorphic multiplications) is again $\Omega(\log^2 p)$, for a total cost of $\Omega(h \log^2 p)$ for depth-h computation. Moreover, even homomorphic $\mathbb{Z}_p$-*addition* requires $\log p$ homomorphic *multiplications*, to compute the "carry" digits. This results in rather high concrete running times, even with optimized implementations for moderately large p (see, e.g., Figure 2).

Recent improvements. Various works over the past several years have improved matters, but existing solutions still leave much to be desired—especially for exact homomorphic arithmetic. For approximate FHE

based on CKKS, the work of [CCKS23] gives a very natural solution, in which a (real or complex) plaintext of high precision p is effectively split into $d \geq 2$ pieces of smaller precision $\approx p^{1/d}$, and the error growth under homomorphic multiplication is proportional to $\approx d \cdot p^{1/d} \geq \log p$. However, the error rate of both *fresh* ciphertexts, and those about to be decrypted or bootstrapped, still needs to be significantly less than $1/p$. For multiplicative depth h, this induces ciphertext sizes and running times of $\Omega((h \log p)(h + \log p))$, which is still quadratic in $\log p$. Moreover, approximate FHE is not naturally suited to many applications, especially computations in finite fields and rings. And unlike exact FHE with negligible decryption error, approximate FHE is completely insecure against an adversary that receives decryptions of honestly generated ciphertexts, unless protected by a costly noise-flooding transformation [LM21, LMSS22].

For high-precision *exact* FHE, the landscape is much more fragmented. The known solutions either: are limited to very rare plaintext moduli of special forms [GC14, CLPX18, GV25b]; work for more kinds of moduli, but have quite complicated and high-latency constructions based on heavy use of bootstrapping and approximate FHE [Kim25b, BK25, Kim26a, CPL26]; or work only for powers of two using a specially designed non-cyclotomic ciphertext ring, without demonstrating SIMD packing or supporting the usual cyclotomic automorphisms [BFG+26]. See Section 1.4 for further details and comparisons to our work.

1.2 Contributions

Our main contribution is a very *simple*, *general*, and *practically efficient* technique for high-precision exact homomorphic encryption, in which the error rates and growth—and thus the FHE parameters and efficiency (LWE dimension, ciphertext size, running times, etc.)—match those of prior schemes for *much smaller plaintext precision*. (See Section 1.3 for an overview of the main techniques, and Section 4 for the details.) Our main FHE scheme, which we call *decomposed BFV* (dBFV) for reasons that will become clear in the technical overview, has the following attractive features:[1]

- **Arbitrary modulus and ring:** dBFV works for *any integer plaintext modulus* (along with many other kinds of plaintext rings), including popular cryptographic constants like $2^{255} - 19$ and the "Goldilocks" prime $2^{64} - 2^{32} + 1$, primes that "split completely" for plaintext packing and SIMD computations [SV11], and any power of two.

  It also works for *any underlying number ring*—or even with *no structured ring at all*, making it suitable for use with plain LWE. While FHE based on plain LWE is not practical, we see this as an important feasibility result, showing for the first time that high-precision FHE with low error growth does not require any special ring structure.

- **Flexibility and efficiency:** dBFV is *flexibly parameterizable* and *practically efficient*. Specifically, it directly implements mod-p plaintext arithmetic via efficient operations on d ordinary BFV ciphertexts, with error rates and growth no worse than those of BFV with plaintext modulus $d \cdot p^{1/d}$. This is optimized for $d \approx \log p$, yielding an *exponential* improvement in the error in terms of the plaintext modulus.

  Ultimately, for many moduli of practical interest (e.g., powers of a small base, or Mersenne-like integers) we get running times of $\tilde{O}(h^2 \log p)$ for depth-h computations—just *quasi-linear* in $\log p$. This significantly improves upon the *quadratic* dependence on $\log p$ for ordinary BGV/BFV, GSW/TFHE, and [CCKS23] (for approximate FHE). It also asymptotically matches the *leveled* (i.e., non-bootstrapped) performance of [CLPX18, GV25b], but dBFV has two further advantages: it supports a *much broader*

[1] An analogous decomposed BGV viewpoint is sketched in Appendix C; here we focus on dBFV, whose security and precision-reduction mechanisms are especially direct.

family of moduli and ring configurations than the very narrow ones covered by those works; and it can bootstrap its own ciphertexts (see the next item and Section 1.4 for a detailed comparison).

- **Compatibility:** dBFV is *fully compatible with prior FHE techniques* for (and implementations of) efficient ring arithmetic, plaintext packing and SIMD operations, bootstrapping (in "packed" or "thin" forms), etc.

  In particular, bootstrapping a dBFV ciphertext trivially reduces to bootstrapping its d ordinary BFV ciphertexts *in parallel*, which can be done generically via any previous approach. Moreover, dBFV and dGSW (the natural decomposed analog of GSW) serve as excellent "hosts" for bootstrapping their own ciphertexts (or even those of other high-precision schemes), achieving low-precision parameters (i.e., inverse error rates and error growth proportional to $d \cdot p^{1/d}$) for the entire ciphertext lifecycle. This is in contrast to the bootstrapping methods of [CCKS23, GV25b, GV25a] (but not [Kim25a]), which require tiny error rates $< 1/p$ for at least one of encryption, bootstrapping, or decryption. (See Sections 1.3.4, 1.4 and 5 for details.)

Preliminary implementation. Our central contribution is the *mathematical definition and analysis of dBFV*. However, as proofs of concept, we also wrote preliminary, unoptimized (single-threaded but AVX-enabled) CPU and GPU implementations of the scheme, and benchmarked them against ordinary BFV and TFHE for homomorphic $\mathbb{Z}_{2^{64}}$-arithmetic with real-world parameters; see Figure 2.[2] The running times scale well, and the observed error-growth factors almost perfectly match the theory's predictions: they are proportional to $d \cdot p^{1/d} = db$, times a fixed factor associated with the ring dimension and size of the secret key. For any fixed ring dimension and original error rate (determining the security level), the supported multiplicative depth (before decryption or bootstrapping) grows inversely with the logarithm of the error-growth factor, so our concrete dBFV parameters improve multiplicative depths *four- to five-fold* versus BFV.

Other contributions. dBFV's way of representing (decomposed) plaintexts also has other advantages, like the ability to more efficiently implement homomorphic *comparisons* and "sign" operations on $\mathbb{Z}_p$ plaintexts. Essentially, we can work from the most- to least-significant of the d BFV ciphertexts, which each encrypt some small "digit" from a narrow range. However, because the digits are relaxed—they can exceed the base of the digit representation—we need to carefully account for their sizes in the homomorphic logic. See Section 1.3.5 for more details.

Finally, as a side contribution, in Section 3 we give a simpler presentation of the BFV scheme [Bra12, FV12] and its generalizations [GC14, CLPX18, GV25b], without any plaintext scale factor or ciphertext modulus. We believe that this more transparently exposes the essential properties of BFV, and will make it easier to discover and communicate further improvements and variants. (It certainly helped the authors of this paper.)

1.3 Technical Overview

Here we summarize the key technical ideas behind our approach to high-precision exact homomorphic encryption. Very briefly, our solution stems from a few main observations:

1. A high-precision plaintext ring is isomorphic to a ring of higher-dimensional *digit decompositions* modulo a certain "geometrically good" lattice.

[2] Our preliminary implementations are fairly basic, and we expect significant speedups from improved algorithms, parallelism, etc. Also, our scheme's advantages over BFV in error rate and running time would be even more pronounced for larger plaintext moduli.

Scheme	Type	Modulus	Ring	SIMD	Mult.	Time ($\tilde{O}$)	Bootstr.
[Bra12, FV12]	exact	any	cyclotomic	yes	native	$\log^2 p$	generic
[CGGI16]	exact	any	cyclotomic	limited	circuit	$\log^2 p$	generic
[CKKS17]	approx	$\mathbb{R}, \mathbb{C}$	cyclotomic	yes	native	$\log^2 p$	generic
[CLPX18]	exact	$b^n + 1$	$\mathbb{Z}[X]/(X^n + 1)$	no	native	$\log p$	via CKKS
[GV25b]	exact	cyc. prime	cyclotomic	yes	native	$\log p$	ad hoc
[CCKS23]	approx	$\mathbb{R}, \mathbb{C}$	cyclotomic	yes	native	$\log^2 p$	not given
[BK25]	exact	any	cyclotomic	yes	many boot	?	via CKKS
[Kim26a]	exact	b^k (b small)	cyclotomic	yes	many boot	$\log p$	via CKKS
[BFG+26]	exact	2^n	custom	no	native	$\log p$	via TFHE
dBFV	**exact**	**any**	**any**	**yes**	**native**	**log *p***	**generic**

Figure 1: Comparison of dBFV with representative prior FHE schemes for high-precision plaintext arithmetic over either $\mathbb{Z}_p$ (for exact schemes) or $\mathbb{R}, \mathbb{C}$ with $\log p$ bits of precision (for approximate schemes). "Time" is the *asymptotic* ($\tilde{O}$) running time of one homomorphic multiplication as a function of $\log p$ alone (and not multiplicative depth). The "?" time entry for [BK25] reflects that the paper does not state an asymptotic running time; concretely, its single-threaded 255-bit multiplication takes 150 seconds. "SIMD" is whether the construction demonstrates native slot packing of independent plaintext values, so that many can be processed in parallel per ciphertext. "Mult." indicates whether multiplication is native or implemented by a circuit; "many boot" indicates that it invokes many bootstrappings. "Bootstr." summarizes bootstrapping support: "generic" means that a variety of standard methods and compatible host schemes can be used, whereas the other entries describe the specific approach currently available.

2. Moreover, this ring isomorphism extends to *noisy, small* digit decompositions, where the noise growth under multiplication is proportional to the sizes of the digits.
3. BFV-style ciphertexts that are *lifted* to small representatives modulo the good lattice encrypt noisy small plaintext decompositions, and arithmetic operations on lifted ciphertexts naturally induce the corresponding operations on those decompositions.[3]

1.3.1 Simpler BFV

We start with an overview of our simpler presentation of the basic BFV scheme [Bra12, FV12] and its generalizations [GC14, CLPX18, GV25b], which will make our main contributions more transparent. In contrast with the prior treatments of BFV, ours has no plaintext scale factor Δ or ciphertext modulus q; instead, it is directly defined at a suitable "scale" and uses fractional (rational) values of suitable precision.

The scheme can be defined over any number field K with ring of integers $R = \mathcal{O}_K$ and ideal $\mathfrak{p}$ of R defining the plaintext ring $R/\mathfrak{p}$.[4] For concreteness and simplicity, in this overview we use $K = \mathbb{Q}$ and $R = \mathbb{Z}$ (to get a scheme based on plain LWE, like [Bra12]), with a plaintext ring of $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ for an integer $p \geq 2$. Everything generalizes straightforwardly to number fields and rings like $K = \mathbb{Q}[X]/(X^n + 1)$ and $R = \mathbb{Z}[X]/(X^n + 1)$ where n is a power of two (for a much more efficient scheme based on Ring- or Module-LWE, like [FV12]), and to any "geometrically nice" ideal $\mathfrak{p}$ (as in [GC14, CLPX18, GV25b]).

[3] Analogous properties hold for BGV [BGV12] and GSW [GSW13] ciphertexts, yielding decomposed versions of those systems as well; see Appendices B and C.

[4] Even more generally, R can be any *order* of K.

$\log_2 n$	$\log_2 b$	d	$\log_2$(**Growth**)	**Depth**	**CPU** **Add**	**CPU** **Mul**	**GPU** **Mul**
12	64	1	74	0	<0.05	2.3	0.18
	16	4	31	3	<0.05	13.6	0.74
	8	8	24	4	<0.10	40.1	1.60
	4	16	22	4	0.10	132.1	3.68
13	64	1	75	2	<0.05	11.4	0.26
	16	4	31	6	<0.10	66.7	1.30
	8	8	25	8	0.20	196.3	2.82
	4	16	22	9	0.59	622.0	6.25
14	64	1	76	4	0.15	58.2	0.92
	16	4	34	12	0.55	350.6	3.95
	8	8	26	15	1.49	994.4	8.96
	4	16	22	18	3.42	3156.0	21.33
(TFHE-rs) 11	2	32	–	–	596.7	16 084	

Figure 2: Empirical error-growth factors, supported multiplicative depths (before bootstrapping), and running times in ms for homomorphic $\mathbb{Z}_p$-arithmetic with plaintext modulus $p = 2^{64}$ in our unoptimized, single-threaded but AVX-enabled CPU (AMD EPYC 9B25, 2.7 GHz) and GPU (single NVIDIA A100, 40 GB) implementations of dBFV. Here n is the power-of-two cyclotomic ring dimension, b is the digit-decomposition base, and $d = \log_b(p)$ is the corresponding numbers of digits; $b = p, d = 1$ is ordinary BFV. The secret key has ternary coefficients with probabilities $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ of $-1, 0, +1$ (respectively); for $\log_2 n = 12, 13, 14$ the error rates are at least $2^{-110}, 2^{-220}, 2^{-440}$ (respectively). The error-growth factors are averages (over several multiplications) of the maximums over all the digit ciphertexts; empirically and in theory, the most-significant digit incurs largest error growth. The last row reports the (also single-threaded, for a direct comparison) running time for Zama's optimized TFHE-rs implementation of *bootstrapped* base-4 homomorphic $\mathbb{Z}_{2^{64}}$-arithmetic, which has no meaningful error growth due to the bootstrapping.

A secret key is a short integer vector $\mathbf{s}$ (over $\mathbb{Z}$), and a ciphertext that encrypts a message $\mu \in \mathbb{Z}_p$ is a vector $\mathbf{c}$ over the *additive group* $\mathbb{Q}_p := \mathbb{Q}/p\mathbb{Z}$—*not* over the ring $\mathbb{Z}$ or $\mathbb{Z}_q$ for a ciphertext modulus q!—such that

$$\langle \mathbf{c}, \mathbf{s} \rangle = \mu + e \approx \mu \pmod{p\mathbb{Z}}, \tag{1.1}$$

where $\approx$ hides a "tiny" error $e \in \mathbb{Q}$ satisfying $|e| < \frac{1}{2}$.[5] This implies that μ can be efficiently recovered from its noisy version $\langle \mathbf{c}, \mathbf{s} \rangle$, simply by rounding to the nearest integer. By the above, we can view the ciphertext as a linear polynomial $\mathbf{c}(S) := \langle \mathbf{c}, S \rangle \in \mathbb{Q}_p[S]$ where the formal variable S represents the secret decryption key. Accordingly, decryption works by computing $\mathbf{c}(\mathbf{s}) \in \mathbb{Q}_p$ and rounding to $\mathbb{Z}_p$, yielding μ.

By linearity, the scheme is additively homomorphic, as long as the accumulated error remains small enough. Homomorphic multiplication is less obvious. First, we lift the $(\mathbb{Q}/p\mathbb{Z})$-entries of $\mathbf{c}$ to small representatives in $\mathbb{Q}$, yielding $\hat{\mathbf{c}}$. Because $\mathbf{s}$ is over $\mathbb{Z}$, by Equation (1.1) this satisfies

$$\langle \hat{\mathbf{c}}, \mathbf{s} \rangle = \hat{\mu} + e \in \mathbb{Q}$$

[5] The *precision* of the rational entries of $\mathbf{c}$ is the analog of the ciphertext modulus q in prior presentations of BFV. It suffices to maintain just enough precision to closely approximate the error, and rounding off to less precision is the analog of modulus switching.

for some $\hat{\mu} \in \mathbb{Z}$ whose magnitude is proportional to p and the norm of $\mathbf{s}$ (which is small), and for which $\mu = \hat{\mu} \bmod p$. Hence, letting $\hat{\mathbf{c}}'$ also be a lifting of a ciphertext $\mathbf{c}'$ that encrypts $\mu' \in \mathbb{Z}_p$ under $\mathbf{s}$ with error $e' \in \mathbb{Q}$, by the mixed-product property of the Kronecker (tensor) product $\otimes$,

$$\langle \hat{\mathbf{c}} \otimes \hat{\mathbf{c}}', \mathbf{s} \otimes \mathbf{s} \rangle = \langle \hat{\mathbf{c}}, \mathbf{s} \rangle \cdot \langle \hat{\mathbf{c}}', \mathbf{s} \rangle \approx \hat{\mu} \cdot \hat{\mu}' \in \mathbb{Z} \,,$$

where $\approx$ hides error $e\hat{\mu}' + \hat{\mu}e' + ee' \in \mathbb{Q}$. That is, $\hat{\mathbf{c}} \otimes \hat{\mathbf{c}}' \bmod p\mathbb{Z}$ can be seen as an encryption of the product $\mu \cdot \mu' \in \mathbb{Z}_p$ under the "quadratic" secret key $\mathbf{s} \otimes \mathbf{s}$, with *error that is larger than the errors in* $\mathbf{c}, \mathbf{c}'$ *by factors of about* $|\hat{\mu}'|, |\hat{\mu}|$ (respectively). Because these factors are proportional to p, this is the source of the large error growth for high-precision plaintexts. Finally, $\mathbf{c} \otimes \mathbf{c}'$ can be "linearized" to a ciphertext that decrypts under $\mathbf{s}$ using the standard technique of key-switching [BV11b], which adds some extra but insignificant error.

1.3.2 (Relaxed) Digit Decompositions

Recall from Section 1.2 that our main technique is able to implement plaintext arithmetic in $\mathbb{Z}_p$ essentially using d ordinary BFV ciphertexts, with error rates and error growth roughly matching that of BFV with plaintext modulus $d \cdot p^{1/d}$. A natural first attempt at obtaining this kind of profile is simply to represent mod-p plaintexts in base b using $d = \lceil \log_b(p) \rceil$ digits, and encrypt each digit separately using homomorphic encryption for $\mathbb{Z}_b$. Indeed, this is how high-precision arithmetic is handled under TFHE [CGGI16], with base $b = 2$. Unfortunately, this approach scales very poorly, because it implements $\mathbb{Z}_p$-arithmetic using circuits of $\mathbb{Z}_b$-gates. So, homomorphic $\mathbb{Z}_p$-multiplication takes $\Omega(d^2)$ homomorphic $\mathbb{Z}_b$-multiplications, and even $\mathbb{Z}_p$-addition takes $\Omega(d)$ *nonlinear* operations (requiring multiplication) to compute the "carry" digits.

Surprisingly, our main technique is not so different from this first attempt, but it supports very natural and efficient homomorphic addition (using only linear operations) and multiplication. A key idea is to allow for a *relaxed* base-b representation of the plaintext, where the magnitudes of the digits may be larger than b, but are still proportional to it. This is heavily inspired by an analogous approach to high-precision plaintexts [CCKS23] in *approximate* FHE [CKKS17], but the technical details are rather different (see Section 1.4 and Figure 1 for a comparison). Notably, and in contrast with [CCKS23], our technique additionally allows for encrypting *fresh ciphertexts* with a much larger LWE error rate proportional to $1/p^{1/d}$, rather than $1/p$. So, our approach matches the behavior of BFV with a much smaller plaintext modulus not only in its error growth under homomorphic operations, but also in the error rate of its fresh ciphertexts.

Gadget decomposition for plaintexts. To start, we recall the concept of *gadget decomposition* [MP12], in particular for the powers-of-b gadget, which is widely used in FHE and lattice cryptography more broadly. For a plaintext $\mu \in \mathbb{Z}_p$ and a base b (where typically $b \ll p$), a gadget decomposition is any vector $\boldsymbol{\mu} \in \mathbb{Z}^d$ for which $\mu = \langle \boldsymbol{\mu}, \mathbf{b} \rangle \pmod p$, where the gadget vector $\mathbf{b} = (1, b, b^2, \ldots, b^{d-1}) \in \mathbb{Z}_p^d$ for $d \approx \log_b(p)$, i.e., $b \approx p^{1/d}$. From another perspective, a decomposition of μ is any *polynomial* $\boldsymbol{\mu} = \boldsymbol{\mu}(B) \in \mathbb{Z}[B]$ for which $\mu = \boldsymbol{\mu}(b) \bmod p$. In other words, it is any element of the coset $\mu + \mathcal{L} \in \mathbb{Z}[B]/\mathcal{L}$, where

$$\mathcal{L} = \{\mathbf{z} \in \mathbb{Z}[B] : \mathbf{z}(b) = 0 \pmod p\} \supseteq p \cdot \mathbb{Z}[B]$$

is an (infinite-dimensional) lattice of polynomials.[6] Of particular interest are small decompositions, whose coefficients are proportional to b, but are not necessarily in $\{0, \ldots, b-1\}$.

With this view, a simple observation is that base-b decompositions are both additively and multiplicatively homomorphic: if $\boldsymbol{\mu}, \boldsymbol{\mu}' \in \mathbb{Z}[B]$ are decompositions of $\mu, \mu' \in \mathbb{Z}_p$ (respectively), then $\boldsymbol{\mu} + \boldsymbol{\mu}', \boldsymbol{\mu} \cdot \boldsymbol{\mu}' \in \mathbb{Z}[B]$

[6] In more formal algebraic terms, $\mathcal{L}$ is an ideal of $\mathbb{Z}[B]$, evaluation at b induces a ring isomorphism from $\mathbb{Z}[B]/\mathcal{L}$ to $\mathbb{Z}_p$, and a decomposition is any representative of the corresponding mod-$\mathcal{L}$ coset.

are decompositions of $\mu + \mu', \mu \cdot \mu' \in \mathbb{Z}_p$ (respectively), simply because polynomial evaluation at b is a ring homomorphism.[7] More importantly—and to our knowledge, this is novel in the context of lattice cryptography—even *noisy, small* decompositions support these homomorphisms: if $\boldsymbol{\mu}, \boldsymbol{\mu}'$ are small and we let $\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu} + \mathbf{e}, \tilde{\boldsymbol{\mu}}' = \boldsymbol{\mu}' + \mathbf{e}' \in \mathbb{Q}[B]$ for some "tiny" errors $\mathbf{e}, \mathbf{e}' \in \mathbb{Q}[B]$ (whose coefficients have magnitudes much smaller than $\frac{1}{2}$), then $\tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\mu}}'$ and $\tilde{\boldsymbol{\mu}} \cdot \tilde{\boldsymbol{\mu}}'$ are noisy, somewhat larger decompositions of $\mu + \mu'$ and $\mu \cdot \mu'$, respectively. This is because

$$\begin{aligned} \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\mu}}' &= (\boldsymbol{\mu} + \boldsymbol{\mu}') + (\mathbf{e} + \mathbf{e}') \\ \tilde{\boldsymbol{\mu}} \cdot \tilde{\boldsymbol{\mu}}' &= (\boldsymbol{\mu} \cdot \boldsymbol{\mu}') + (\mathbf{e} \cdot \boldsymbol{\mu}' + \boldsymbol{\mu} \cdot \mathbf{e}' + \mathbf{e} \cdot \mathbf{e}') \ , \end{aligned}$$

and each induced error (the second addend of each right-hand side) is comparably tiny in $\mathbb{Q}[B]$.

More specifically, under homomorphic addition the original errors simply sum, and under homomorphic multiplication they are enlarged by the sizes of $\boldsymbol{\mu}, \boldsymbol{\mu}'$, which are proportional to only $bd \approx b \log_b(p) \approx d \cdot p^{1/d}$. We emphasize that here it is vital that the original decompositions $\boldsymbol{\mu}, \boldsymbol{\mu}'$ be *small*, because they amplify the original errors. Lastly, all this immediately generalizes from $\mathbb{Z}_p$ to $R/\mathfrak{p}$, where a decomposition of $\mu \in R/\mathfrak{p}$ is just a polynomial $\boldsymbol{\mu}(B) \in R[B]$ for which $\mu = \boldsymbol{\mu}(b) \bmod \mathfrak{p}$ for a specified base $b \in R$.

1.3.3 Decomposed BFV

We can now describe our BFV variant for high-precision plaintexts. The first key idea is to implement the plaintext ring via the isomorphic ring of digit decompositions. More specifically, a ciphertext effectively hides a *noisy, small* digit decomposition of the plaintext; for this reason, we call the scheme *(digit) decomposed BFV*, or dBFV for short.

For this overview, we continue with the example of plaintext ring $\mathbb{Z}_p$, which, as we have seen above, is isomorphic to $\mathbb{Z}[B]/\mathcal{L}$.[8] In dBFV, a ciphertext that encrypts $\mu \in \mathbb{Z}_p$ under a secret key $\mathbf{s}$ (over $\mathbb{Z}$) is a *bivariate* polynomial $\mathbf{c}(B, S) \in (\mathbb{Q}[B]/\mathcal{L})[S]$, which is typically affine linear in S, that satisfies

$$\mathbf{c}(B, \mathbf{s}) = \boldsymbol{\mu}(B) + \mathbf{e}(B) = \mu + \mathbf{e}(B) \pmod{\mathcal{L}} \tag{1.2}$$

for an arbitrary decomposition $\boldsymbol{\mu}(B) \in \mu + \mathcal{L}$ and tiny error $\mathbf{e}(B) \in \mathbb{Q}[B]$. Accordingly, decryption just computes $\mathbf{c}(B, \mathbf{s}) \in \mathbb{Q}[B]/\mathcal{L}$, rounds off its coefficients to their nearest integers to get $\boldsymbol{\mu}(B) \bmod \mathcal{L}$, and evaluates it at b to get $\mu \in \mathbb{Z}_p$. We stress that it is important to round the coefficients *before* evaluation, because otherwise the tiny error would be "blown up" by the large powers of b, causing an incorrect output. Also notice that, since $p\mathbb{Z}[B] \subseteq \mathcal{L}$, we may lift the ciphertext to some $\sum_i \mathbf{c}_i(S) \cdot B^i \in \mathbb{Q}_p[B, S]$, and view each $\mathbf{c}_i(S)$ as an ordinary mod-p BFV ciphertext that encrypts the ith digit (modulo p) of some digit decomposition $\boldsymbol{\mu}(B) = \sum_i \mu_i \cdot B^i$ of μ.

Encryption. We can generate a fresh dBFV ciphertext simply by creating an ordinary mod-p BFV ciphertext $\mathbf{c}(S) \in \mathbb{Q}_p[S]$, viewing it as a constant-in-B polynomial, and reducing it modulo $\mathcal{L}$ (see Item 2 of Remark 4.4). However, this is a highly suboptimal way of encrypting in dBFV. This is because the error, whose magnitude must be $\ll 1/2$ to support homomorphisms and correct decryption, has a *rate* (relative to the modulus p) significantly smaller than $1/p$, so for security the LWE dimension and precision need to be correspondingly large. The related work [CCKS23] has an analogous drawback.

[7] Note that multiplication typically yields a higher-degree decomposition; we address this later.

[8] Again, everything naturally generalizes to plaintext ring $R/\mathfrak{p}$; in particular, dBFV can directly use any "SIMD slots" arising from the factorization of $\mathfrak{p}$ and the Chinese Remainder Theorem.

To address this issue, in Section 4.3 we give a different encryption method for dBFV, based on a natural variant of LWE over $\mathbb{Q}[B]/\mathcal{L}$ instead of $\mathbb{Q}_p$, whose security can be tightly based on a corresponding ordinary (plain/Module/Ring) LWE problem (see Appendix A). In this method, error of the same coefficient-wise *size* (over $\mathbb{Q}$) has much larger *rate* relative to the modulus lattice $\mathcal{L}$, proportional to about $1/p^{1/d} \gg 1/p$. This matches the initial error rate of ordinary BFV with a modulus of about $p^{1/d}$, allowing for smaller LWE parameters—even ignoring the improved error *growth* under homomorphisms, which we describe next.

Homomorphisms, lifting, and degree reduction. It is immediate from Equation (1.2) that dBFV is additively homomorphic. Less obviously, it is also multiplicatively homomorphic, as follows. Analogously to ordinary mod-p BFV, lifting the coefficients of a dBFV ciphertext $\mathbf{c}(B, S) \in (\mathbb{Q}[B]/\mathcal{L})[S]$ to *small representatives modulo* $\mathcal{L}$ in $\mathbb{Q}[B]$ yields a $\hat{\mathbf{c}}(B, S) \in \mathbb{Q}[B, S]$ for which

$$\hat{\mathbf{c}}(B, \mathbf{s}) = \hat{\boldsymbol{\mu}}(B) + \mathbf{e}(B) \in \mathbb{Q}[B] \tag{1.3}$$

for some *small* decomposition $\hat{\boldsymbol{\mu}}(B) \in \mathbb{Z}[B]$ of μ, whose size is proportional to the sizes of $\mathbf{s}$ and $\hat{\mathbf{c}}$. So, simply multiplying such lifted ciphertexts *as (bivariate) polynomials* induces multiplication of their noisy small plaintext decompositions. As we saw above, this yields a noisy decomposition (of larger degree in B) of the plaintext product, with relatively small error growth proportional to just $d \cdot p^{1/d}$. Together with the above encryption method, this means that dBFV enjoys the same (or slightly better) error profile as ordinary BFV with a plaintext modulus of about $d \cdot p^{1/d}$, in both the *initial* error rate and the error *growth* under homomorphic operations.

To complete the homomorphic multiplication, making the output have the same syntactic form as a fresh ciphertext, we first reduce the product of the lifted ciphertexts modulo $\mathcal{L}$, then reduce its degree in B, and finally linearize it in S via key switching. Reducing modulo $\mathcal{L}$ and key switching are standard; the new ingredients are lifting the input ciphertexts to small representatives modulo $\mathcal{L}$ before multiplication, and reducing the degree in B afterward. These two operations are described in Section 4.6. For lifting, we use the short basis of $\mathcal{L}$ given in [MP12], and its associated fast algorithm for the special case $p = b^d$; for general p and b, we give a new, fast lifting algorithm that is a hybrid of the nearest-plane and round-off algorithms using the basis (see Section 4.6.2). For degree reduction, we observe that any high power of B is congruent modulo $\mathcal{L}$ to a small linear combination of the low powers of B, so we can replace any rational multiple of the former with the same multiple of the latter, thus *projecting* the ciphertext (and its underlying error polynomial) down to have bounded degree in B. Related linear maps also let us change the base b of a dBFV ciphertext as needed to get a tradeoff between ciphertext size (and runtime) and error growth; see Section 4.5.

1.3.4 Bootstrapping dBFV

As mentioned in Section 1.2, bootstrapping our mod-p dBFV scheme trivially reduces to bootstrapping its d ordinary BFV ciphertexts in parallel: lifting the dBFV ciphertext to some $\sum_i \mathbf{c}_i(S) \cdot B^i \in \mathbb{Q}_p[B, S]$, we simply bootstrap each mod-p BFV ciphertext $\mathbf{c}_i(S)$, then linearly combine the results with the corresponding powers of B. Bootstrapping these BFV ciphertexts can be done via any of the many approaches from the literature, using either a second-generation BGV/BFV/CKKS-style scheme [GHS12a, AP13, HS15, CH18, OPP23, GV23, GIKV23, MHWW24, KSS24, GV25a], or a third-generation GSW/FHEW/TFHE-style scheme [BV14, AP14, DM15, CGGI16, KDE+24], as the "host" for the bootstrapping key and the homomorphic computation.

We emphasize that with this approach, the host scheme must work on high-precision (mod-p or larger) plaintexts. So, using an ordinary second-generation scheme as the host would incur enormous error growth

during bootstrapping, thus requiring a very small error rate in the bootstrapping key (with the corresponding inefficiencies). And while an ordinary third-generation host scheme would incur fairly small error *growth* (nearly independent of p), the error *rate* in the bootstrapping key would still need to be $\ll 1/p$.

We resolve these issues by using our decomposed schemes as hosts (for bootstrapping their own ciphertexts, or even those of other high-precision schemes). As a second-generation host we can just use dBFV itself, which results in error rates (in the bootstrapping key) and error growth (during bootstrapping) that match the low-precision $d \cdot p^{1/d}$ regime *end-to-end*, for the entire ciphertext lifecycle. As a third-generation host, we introduce dGSW—a natural decomposed variant of GSW—and show that, like dBFV, it can encrypt mod-p plaintexts with error rates proportional to just $1/p^{1/d}$. See Section 5 for the details of bootstrapping, and Appendix B for the details of dGSW.

1.3.5 Homomorphic Comparisons in dBFV

The appearance of small plaintext digit decompositions $\hat{\boldsymbol{\mu}}(B)$ in the lifted decryption formula (Equation (1.3)) has other benefits besides smaller error growth in homomorphic multiplication. Importantly, it also allows for more efficient *homomorphic comparisons* and *sign* operations on $\mathbb{Z}_p$ for $p = b^d$, which we identify with the representatives $\{-\lfloor \frac{p}{2} \rfloor, \ldots, \lfloor \frac{p-1}{2} \rfloor\}$ for these purposes. Here we outline the main ideas, and leave a detailed development and implementation to future work.

In a usual base-b representation, the sign of the most-significant digit determines the sign of the entire value, because the cumulative magnitude of the remaining digits cannot be large enough to flip this sign. But in our setting this is no longer true, because the digits are relaxed—their magnitudes can be larger than b, though they are bounded by $\gamma \cdot b$ for some fairly small $\gamma \geq 1$ (proportional to the norm of the secret key). So, we use the following more general observation: if the most-significant digit has magnitude at least γ, then its sign *does* determine the sign of the entire value; otherwise, its exact value may matter for determining the sign. In other words, we can "clip" the most-significant digit (or more generally, the composite value of multiple high digits) to the interval $[-\gamma, \gamma]$ without changing the sign of the entire value.

We apply this idea iteratively (and homomorphically) from the most- to least-significant digit, accumulating a clipped value of the digits processed so far. At each iteration we use TFHE programmable bootstrapping (PBS) to clip the current accumulated value to have magnitude at most $\approx \gamma$, then multiply it by b and add it to the next-most-significant digit; finally, we output the sign of the last accumulated value. This strategy keeps the magnitude of the accumulator fairly small (at most $\approx 2\gamma \cdot b$), and preserves the sign of the original value. Since PBS requires the input plaintext to come from a fairly small interval (of at most a few thousand values), we may first need to do a homomorphic change of base (Section 4.5) to make b small enough (though prior computations can still use a moderately large b).

Notice that the above strategy does $d \approx \log_b(p)$ *sequential* PBS over the digits. There is also a related *parallel* strategy, inspired by and generalizing classic *carry-lookahead logic* for digital-adder circuits, which does about $4d$ PBS arranged in depth roughly $\log d$. The basic idea is to homomorphically compute, for each digit in parallel, its quotient and remainder when divided by b; these are respectively analogous to the "carry" and "propagate" bits in digital logic. Then similar logic stitches these together in a tree topology to determine the sign of the full value.

1.4 Comparison to Related Work

Here we give a more detailed overview of prior work on FHE for high-precision plaintexts, and compare our work to it (see Figure 1 for a summary).

BGV/BFV-style exact FHE. One line of work [GC14, CLPX18, GV25b, BFG+26] works over polynomial rings like $R = \mathbb{Z}[X]/(f(X))$, and adapts BGV/BFV-style exact-arithmetic schemes to use specially constructed ideals $\mathfrak{p}$ as the plaintext moduli, rather than just $\mathfrak{p} = pR$ for an integer p. Essentially, these works arrange for $R/\mathfrak{p}$ to have a desired ring structure—e.g., being isomorphic to $\mathbb{Z}_p$ for a large integer p—and to have much smaller R-representatives than R_p has, owing to the existence of a known short basis of $\mathfrak{p}$. The isomorphism directly induces the desired kind of plaintext arithmetic under homomorphic operations, whereas the smaller representatives lead to smaller error growth under homomorphic multiplication. In more detail:

- The first two works [GC14, CLPX18] are based on an observation of [HS01], originally used for the NTRU cryptosystem [HPS98]: for the ring $R = \mathbb{Z}[X]/(X^n + 1)$ and the ideal $\mathfrak{p} = (X - b)R$, the quotient ring $R/\mathfrak{p}$ is isomorphic to $\mathbb{Z}_p$ for $p = b^n + 1$, and it has representatives of size proportional to $b \approx p^{1/n}$. However, because n is typically in the many hundreds or even several thousands, these works are limited to enormous plaintext rings $\mathbb{Z}_p$, far larger than what applications typically need. Practical bootstrapping for these schemes was only recently provided by [Kim25a], via reduction to CKKS bootstrapping.
- The third work [GV25b] uses a broader class of ideals $\mathfrak{p}$, corresponding to ones in *lower-degree subrings* of R, which can yield moderate-sized integer moduli p, such as the "Goldilocks prime" $2^{64} - 2^{32} + 1$. However, the set of supported moduli is still very limited, to the "cyclotomic primes." This work also gives a bootstrapping procedure that is practical for moderate-precision plaintexts, by converting to and bootstrapping ordinary BFV ciphertexts with (large) integer modulus p; however, this requires error rates $\ll 1/p$ and thus makes the running time proportional to $\log^2 p$. Recently, bootstrapping for [GV25b] was improved in two distinct ways: [GV25a] keeps more of the computation outside of ordinary BFV, thus improving the error growth during bootstrapping, but still requiring ordinary BFV with modulus p; and [Kim25a] handles high precision by reduction to CKKS bootstrapping.
- The fourth work, REFHE [BFG+26]—which is concurrent to and independent of ours—uses a specially designed *non-cyclotomic* ring $R = \mathbb{Z}[X]/(X^N - X + 2)$ and the ideal $\mathfrak{p} = (X - 2)R$. This gives the native plaintext space $\mathbb{Z}_{2^N}$, which can simulate $\mathbb{Z}_{2^k}$ for any $k \leq N$ via the natural quotient homomorphism $\mathbb{Z}_{2^N} \to \mathbb{Z}_{2^k}$. However, the construction encrypts only one logical N-bit word per full degree-N ciphertext; it does not provide SIMD packing, and the usual cyclotomic automorphisms are unavailable. Fast multiplication lifts to $\mathbb{Z}[X]$, multiplies without wraparound using an NTT-friendly ring of about twice the degree, and reduces modulo $X^N - X + 2$; the authors estimate this to cost about twice a degree-N cyclotomic multiplication.

  The REFHE work also describes, but does not implement, a method for bootstrapping its ciphertexts. One complete REFHE bootstrap extracts the plaintext bits sequentially, internally invoking several TFHE-style programmable bootstraps (PBS) per bit, with an amortization technique that reduces their total number. This also enables homomorphic comparison via one complete REFHE bootstrap. For $k = 64$, the REFHE authors estimate that bootstrapping with one level of multiplicative depth remaining costs about one-half as much as bitwise $\mathbb{Z}_{2^k}$-multiplication in TFHE (see Figure 2).

Summarizing the above line of work, the approach of using BGV/BFV with a special modulus ideal has been limited to very specific kinds of plaintext rings, which must be closely tied to the choice and structure of the polynomial ring and ideal.

By contrast, our solution is *completely general*: it works for *any* integer modulus (and many other kinds of modulus ideals), irrespective of the choice of ring, and even in the absence of a structured ring—i.e., it works

even with plain LWE. Indeed, our technique is orthogonal to and fully compatible with the above and other techniques for designing rich plaintext spaces, such as SIMD packing [SV11]. In addition, bootstrapping our scheme very simply and *generically* reduces to a small number of *parallel* BFV bootstrappings, which can be instantiated by any of several existing approaches—e.g., for packed or unpacked ciphertexts, with (d)BFV or (d)GSW as the "host" scheme. Finally, we support PBS-based homomorphic comparisons (see Section 1.3.5) using fewer, parallel PBS calls: we can handle a small number of plaintext bits per PBS, whereas REFHE performs a comparison using its full word-bootstrapping procedure, which extracts the bits sequentially using multiple PBS calls per bit.

CKKS-style approximate FHE. We also know of several works that use the CKKS approximate FHE scheme as a foundation for computing with high-precision plaintexts [CCKS23, Kim25b, BK25, Kim26a, CPL26]. Before summarizing these, we note that for approximate FHE schemes, obtaining security in the presence of decryptions of honestly generated ciphertexts (known as IND-CPA^D security) requires a "noise flooding" transformation that decreases error rates by a factor exponential in a statistical security parameter [LM21, LMSS22], which is fairly costly for efficiency. This is in contrast to *exact* FHE schemes with the standard notion of negligible decryption error, which have such security automatically.

The first work [CCKS23] gives a very natural and efficient solution for high-precision approximate arithmetic, in which a plaintext is effectively split into two or more pieces of equal precision, and the error growth under multiplication is roughly proportional to this precision. Our work is inspired by this approach at a conceptual and technical level, but the details are rather different owing to the differences between the plaintext rings ($\mathbb{R}$ or $\mathbb{C}$ versus $R/\mathfrak{p}$), the decomposition of plaintexts, approximate versus exact arithmetic, etc. In addition, our work obtains *exponentially* better error rates (versus ordinary BFV) for fresh ciphertexts, whereas [CCKS23] has rates inversely proportional to the precision (just as in ordinary CKKS). This gives us running times just *quasi-linear* in the bits of plaintext precision, versus at least quadratic in [CCKS23].

Several other recent works [BK25, Kim25b, Kim26a, CPL26] use discrete CKKS for exact homomorphic integer arithmetic [DMPS24]. They represent large integers in radix or nested-residue form, and use CKKS-style bootstrapping and modular reduction to maintain these encodings throughout homomorphic operations [KN25]. These methods support powers of small bases and, in some cases, arbitrary moduli, and can achieve good amortized (SIMD) throughput; however, their latency remains quite high. For example, [Kim26a] reports 24.9 seconds for one 64-bit multiplication, versus 12.2 milliseconds amortized per slot. After the initial version of this work, further discrete-CKKS methods improved the latency and flexibility of integer and logical operations [Rov26, Kim26b].

By contrast, our solution *natively* supports arithmetic modulo any integer, via very simple and fast homomorphic operations—just ordinary polynomial addition and multiplication (after lifting to short representatives). So, our homomorphic addition is almost instantaneous, and in our (unoptimized) implementation, homomorphic multiplication is in the tens or hundreds of milliseconds for moduli of 64–128 bits.

Other related work. Two other recent works [KLSS23, BCG$^+$24] are related to our approach and helped inspire it. In short, they show that representing *ciphertexts* in a relaxed (non-unique) digit-decomposed form (versus over $\mathbb{Z}_q$) can yield improved practical efficiency for FHE schemes. But unlike our work, they do not address large plaintext spaces, improve error growth, or change the underlying mathematical definitions of ciphertexts and their homomorphic operations.

Acknowledgments. We thank the anonymous Crypto'26 reviewers, Robin Geelen, and Simon Pohmann for helpful suggestions, questions, and observations.

2 Mathematical Background

In this work, every ring is implicitly commutative with identity. For a module M over some ring, we let $M[X]$ denote the set of polynomials in formal variable X having coefficients in M, and let $M^{<d}[X]$ be the set of such polynomials having degree less than d. (These are also modules over the same ring, but are not necessarily rings themselves, unless M is a ring.) Any such polynomial can be evaluated at any element of the ring in the usual way, yielding an element of M.

Let S be a subgroup of an additive group G. For $g, g' \in G$ we write $g = g' \pmod{S}$ if $g - g' \in S$. The *coset* $g + S = \{g + s : s \in S\}$ is the set of all group elements congruent to g modulo S. The *quotient* G/S (pronounced "G modulo S") is the set of all such cosets for $g \in G$. It forms an additive group, under the group law $(g + S) + (g' + S) = (g + g') + S$. It may also be a module over a ring (e.g., if G and S are modules over the same ring), or even a ring itself. For computational purposes, a coset may be represented by any one of its elements, but for security and efficiency reasons it may be important to use a specific, unique *distinguished representative* (which should be specifically defined).

2.1 (Cyclotomic) Number Fields and Rings

The FHE schemes described in this paper can be defined over any number field K and its ring of integers $R = \mathcal{O}_K$ (or even more generally, any *order* of K). For concreteness and simplicity, we mainly focus on *cyclotomic* fields and rings, which are the most widely used kind in lattice cryptography and FHE.

For a positive integer m, the mth cyclotomic number field is $K = \mathbb{Q}(\zeta_m)$, and its ring of integers is $R = \mathbb{Z}[\zeta_m]$, where ζ_m denotes a primitive mth root of unity. The degree of K over $\mathbb{Q}$ (and rank of R over $\mathbb{Z}$) is $n = \varphi(m)$, the Euler totient of m. For an integer $m' \mid m$, the m'th cyclotomic field can be seen as a subfield of the mth one (and similarly for their rings of integers), by identifying $\zeta_{m'}$ with $\zeta_m^{m/m'}$.

- The simplest cyclotomic is $K = \mathbb{Q}$ and $R = \mathbb{Z}$. With this choice, the security of our cryptosystems is ultimately based on plain LWE [Reg05], like the one of Brakerski [Bra12].

- Much better efficiency can be obtained using higher-degree cyclotomics. The simplest and most common choice in practice is a two-power cyclotomic field $K \cong \mathbb{Q}[X]/(X^n + 1)$ and ring $R \cong \mathbb{Z}[X]/(X^n + 1)$ where $m = 2n$ is a power of two, $X^n + 1$ is the mth cyclotomic polynomial, and ζ_m corresponds to X under the isomorphisms. But any other cyclotomic field and ring may be used as well, as explored in [LPR13]. With this kind of choice, the security of our cryptosystems is based on Ring-LWE [LPR10], like Fan and Vercauteren's variant [FV12] of [Bra12], or on Module-LWE [BGV12, LS15] (which interpolates between plain and Ring-LWE).

For most of the paper we treat the field K and ring R *abstractly*, deferring to Appendix D the issue of how a computer can represent and efficiently operate on their elements, and on polynomials over the field and ring. In summary, *all arithmetic operations we need can be implemented in time* ***quasi-linear*** *in the bit lengths of the inputs*.

A (nonzero) *ideal* of R is a (nontrivial) additive subgroup $\mathfrak{a}$ that is closed under multiplication by R, i.e., $a \cdot r \in \mathfrak{a}$ for every $a \in \mathfrak{a}$ and $r \in R$; in this work, all ideals are implicitly nonzero. The quotient $R/\mathfrak{a} = \{r + \mathfrak{a} : r \in R\}$ of cosets of $\mathfrak{a}$ forms a ring, under the usual definition of addition and multiplication of cosets: $(r + \mathfrak{a}) + (r' + \mathfrak{a}) := (r + r') + \mathfrak{a}$ and $(r + \mathfrak{a}) \cdot (r' + \mathfrak{a}) := (r \cdot r') + \mathfrak{a}$. Any ideal $\mathfrak{a}$ of a degree-n number ring has a $\mathbb{Z}$-basis of rank n, i.e., a set of n elements such that every element of $\mathfrak{a}$ can be written uniquely as a $\mathbb{Z}$-linear combination of the basis elements.

2.2 Canonical Embedding

Elements of K have "sizes," which are most naturally measured using the *canonical embedding* (also sometimes called the *Minkowski* embedding). A number field of degree n (over $\mathbb{Q}$) has n ring embeddings (i.e., injective ring homomorphisms) $\sigma_i \colon K \to \mathbb{C}$ into the complex numbers. The canonical embedding $\boldsymbol{\sigma} \colon K \to \mathbb{C}^n$ is merely the concatenation of all these embeddings, i.e., $\boldsymbol{\sigma}(a) := (\sigma_i(a))_i$.

We measure the "sizes" of field elements in terms of the canonical embedding, i.e., for $a \in K$ we define the canonical Euclidean and max norms (respectively) as

$$\|a\|_2 := \|\boldsymbol{\sigma}(a)\|_2 = \Big(\sum_i |\sigma_i(a)|^2\Big)^{1/2},$$
$$\|a\|_\infty := \|\boldsymbol{\sigma}(a)\|_\infty = \max_i |\sigma_i(a)| .$$

Because $\sigma_i(a \cdot b) = \sigma_i(a) \cdot \sigma_i(b)$, the max norm $\|a\|_\infty$ bounds the "expansion" of multiplication by a: $\|a \cdot b\|_p \le \|a\|_\infty \cdot \|b\|_p$ for $p = 2, \infty$ (and indeed, for any other ℓ_p norm as well). Therefore, we use the max norm as a generalization of the absolute value to measure the size of a with respect to multiplication.

Polynomials over K. It is also convenient to extend the above norms from K to polynomials of finite degree in $K[B]$. For $\mathbf{u}(B) = \sum_i u_i \cdot B^i \in K[B]$ and any $p \in \{2, \infty\}$, define the (coefficient-wise) canonical ℓ_p norm $\|\mathbf{u}\|_p := \max_i \|u_i\|_p$, and define the canonical operator norm $\|\mathbf{u}\|_{\mathrm{op}} := \sum_i \|u_i\|_\infty$. Then for any finite-degree $\mathbf{u}(B), \mathbf{v}(B) \in K[B]$ and any $p \in \{2, \infty\}$,

$$\|\mathbf{u} \cdot \mathbf{v}\|_p \;\le\; \|\mathbf{u}\|_{\mathrm{op}} \cdot \|\mathbf{v}\|_p \,, \tag{2.1}$$

i.e., $\|\mathbf{u}\|_{\mathrm{op}}$ bounds the "expansion" of polynomial multiplication by $\mathbf{u}$ in either canonical ℓ_p norm. (This follows from the above expansion bound $\|a \cdot b\|_p \le \|a\|_\infty \cdot \|b\|_p$ on K, together with the triangle inequality over the coefficients of $\mathbf{u}$.[9]) The Euclidean variant ($p = 2$) is the one relevant for decryption, which requires bounding the canonical Euclidean norm of each error coefficient (see Section 2.3).

Cyclotomics. Concretely, for a cyclotomic field $K = \mathbb{Q}(\zeta_m)$, the complex embeddings are induced by $\sigma_i(\zeta_m) = \omega_m^i$ for $i \in \mathbb{Z}_m^*$, where $\omega_m = \exp(2\pi\sqrt{-1}/m) \in \mathbb{C}$ is a primitive mth root of unity. In particular, when $m = 2n$ is a power of two and we view $K \cong \mathbb{Q}[X]/(X^n + 1)$, the canonical embedding corresponds to polynomial evaluation at all the primitive mth roots of unity ω_m^{2j+1} for $j \in \mathbb{Z}_n$. This is a scaled orthogonal transformation, with scale factor $\sqrt{n}$, on the coefficient vector of the polynomial. So in power-of-two cyclotomics the coefficient vector of a field element is a close proxy for its canonical embedding.

2.3 Lifting and Decoding

For any positive integer d, an *R-lattice* in K^d is a finitely generated R-submodule whose K-span is all of K^d; in particular, R and all its fractional ideals are R-lattices in K. Under the canonical embedding, every R-lattice in K^d is a rank-nd lattice over $\mathbb{Z}$.

In this work, we need to efficiently *lift* cosets modulo certain R-lattices to representatives, which for many purposes must additionally be short. We also need to *decode modulo R* under "tiny" error in K, i.e., given any

[9] Heuristically, when the coefficients of $\mathbf{v}$ behave like independent, mean-zero random variables, each coefficient of $\mathbf{u} \cdot \mathbf{v}$ behaves as a central-limit-style sum, and $\|\mathbf{u}\|_{\mathrm{op}}$ in (2.1) can be replaced by roughly $\sqrt{\deg(\mathbf{u})} \cdot \|\mathbf{u}\|_\infty$.

$t = e \pmod R$ where $e \in K$ is sufficiently small, we need to efficiently recover e.[10] These operations can be done via standard efficient lattice algorithms, like the "round-off" or "nearest-plane" algorithms [Bab85], using a short basis of the relevant R-lattice. A *lifting rule* for an R-lattice $\mathcal{L}$ takes a coset in $K^d/\mathcal{L}$ (or $K_{\mathbb{R}}^d/\mathcal{L}$, where $K_{\mathbb{R}} := K \otimes_{\mathbb{Q}} \mathbb{R}$ is the continuous analog of K) as input and returns a representative of that coset; it is *short* if its outputs satisfy an appropriate shortness bound. It may be randomized, in which case it samples from a distribution determined only by the input coset, using fresh independent randomness.

An input coset $\mathbf{a} + \mathcal{L}$ can be lifted using the following "round-off" method. Fix a $\mathbb{Z}$-basis $\{\mathbf{b}_j\}$ of $\mathcal{L}$, write $\mathbf{a} + \mathcal{L} = \sum_j \alpha_j \mathbf{b}_j$, where the coefficients α_j are cosets of $\mathbb{Z}$, lift each α_j to its distinguished representative in $[-\frac{1}{2}, \frac{1}{2})$, and output the corresponding linear combination of the basis elements. The resulting lift has canonical Euclidean or max norm at most half the sum of the corresponding norms of the basis elements. (When an R-basis of $\mathcal{L}$ is available, it can be used directly by lifting the mod-R coefficient cosets of the input coset.) Somewhat tighter bounds, or distributional properties, can be obtained using special features of the basis and/or more sophisticated lifting methods, like the nearest-plane algorithm [Bab85] or a randomized variant as in [GPV08].

Decoding $t = e \pmod R$ can be done in the same way, by lifting t to a short representative $e' \in K$ (as described above) using a suitably "good" $\mathbb{Z}$-basis $\{b_j\}$ of R. This correctly recovers $e' = e$ as long as all the $\mathbb{Q}$-coefficients of $e \in K$ with respect to the basis are in $[-\frac{1}{2}, \frac{1}{2})$. The jth coefficient is $\mathrm{Tr}(e b_j^\vee)$, where $\{b_j^\vee\} \subset K$ is the *trace-dual* of the basis $\{b_j\}$, which satisfies $\mathrm{Tr}(b_j b_{j'}^\vee) = \delta_{j,j'}$. So by Cauchy-Schwarz, decoding is correct if $\|e\|_2 < 1/(2 \max_j \|b_j^\vee\|_2)$, i.e., as long as the canonical Euclidean norm of e is small enough relative to the canonical Euclidean norms of the trace-dual basis elements. Tighter results can be obtained using properties of the distribution of e and/or of the specific basis, like (near-)orthogonality; see, e.g., [LPR13, Sections 2.4.1 and 6].

Cyclotomics. Power-of-two cyclotomic rings $R \cong \mathbb{Z}[X]/(X^n + 1)$ are geometrically very nice for the above purposes: they have $\mathbb{Z}$-bases $\{X^i\}_{i=0}^{n-1}$ that are *orthogonal* in the canonical embedding, whose elements all have Euclidean norms $\sqrt{n}$ and max norms 1. The trace-dual basis elements are simply the scaled inverses X^{-i}/n, so they are also orthogonal, with Euclidean norms $1/\sqrt{n}$ and max norms $1/n$. Other cyclotomic rings have integral power bases whose elements have these same norms, but these bases need not be orthogonal, which complicates the geometry somewhat; see [LPR13] for details.

More generally, any principal fractional ideal $\mathfrak{a} = aR$ for some nonzero $a \in K$ has a $\mathbb{Z}$-basis that is merely a times a $\mathbb{Z}$-basis of R. As noted in Section 2.2 above, the geometric effect of this can be bounded coarsely by the max norm $\|a\|_\infty$, or more finely by using special properties of a and the basis of R. Prior works have used (small or large) integer scalings of R, or in some cases [GC14, CLPX18, GV25b] special small non-integer elements $a \in R \setminus \mathbb{Z}$. In this work, we mainly use integer scalings of R and their inverses.

3 BFV Homomorphic Encryption

As a warmup for our main result (and a contribution of independent interest), here we start with a high-level mathematical description of the BFV homomorphic encryption scheme [Bra12, FV12], incorporating some generalizations to its plaintext space [GC14, CLPX18, GV25b]. Our description works at a "scale" that eliminates the need for the *plaintext scaling factor* Δ and *ciphertext modulus* q (or Q) from prior works,

[10] To conform with the definition and recommended usage of Ring-LWE in general number fields [LPR10, LPR13], we may also need to decode modulo the *dual* lattice $R^\vee$. The treatment here adapts to that case simply by swapping the roles (and $\mathbb{Z}$-bases) of R and $R^\vee$. Or, we can suitably "scale up" $R^\vee$ to R, which suffices for the cyclotomic rings we focus on in this work.

which makes the presentation simpler and more transparent, and leads more naturally to the dBFV scheme in Section 4.

3.1 Mathematical Structures

The BFV scheme is defined over a fixed, public number field K and its ring of integers R, which are usually (power-of-two) cyclotomics. The plaintext "modulus" is a fixed, public ideal $\mathfrak{p}$ of R, which defines the quotient ring $R/\mathfrak{p}$ of plaintexts. It also defines the quotient R-module $K/\mathfrak{p}$ (which is not a ring, because it is not closed under multiplication).

A simple and common choice is $\mathfrak{p} = pR$ for an integer $p \in \mathbb{Z}$. However, following [GC14, CLPX18, GV25b], other choices of $\mathfrak{p}$ can yield useful results as well. In addition, plaintexts can be restricted to a smaller-rank subring of $R/\mathfrak{p}$, such as just the $\mathbb{Z}$-cosets $\{z + \mathfrak{p} : z \in \mathbb{Z}\}$. More generally, when $R = \mathbb{Z}[\zeta_m]$ is the mth cyclotomic, we can restrict to the $\mathbb{Z}[\zeta_{m'}]$-cosets of $\mathfrak{p}$, where $\zeta_{m'} = \zeta_m^{m/m'}$ for some $m' \mid m$.

We stress that, unlike previous presentations of the BFV scheme, our version *does not have any "ciphertext modulus"* (usually called q). The ideal $\mathfrak{p}$ that defines the *plaintext* ring $R/\mathfrak{p}$ is the only "modulus" in the system; ciphertexts are simply vectors over the number field K modulo $\mathfrak{p}$ (see Definition 3.1 below). However, the *precision* of these field elements is closely analogous to the size of the ciphertext modulus, as will become clear below.

3.2 Secret Keys and Ciphertexts

Here we define the form and properties of BFV secret keys and ciphertexts. A secret key is a vector $\mathbf{s} \in R^k$, all of whose entries are "small" (e.g., in the canonical embedding, or in the ℓ_1 norm over its coefficients). When basing security on LWE, k is the LWE dimension. When using Ring-LWE (over R), $k = 1$. In general, when using Module-LWE, k is the rank of the underlying problem.

Definition 3.1 (BFV ciphertext). A *BFV ciphertext* that encrypts a plaintext $\mu \in R/\mathfrak{p}$ under a secret key $\mathbf{s} \in R^k$ is a vector $\mathbf{c} = (c_0, \mathbf{c}_1) \in (K/\mathfrak{p}) \times (K/\mathfrak{p})^k$ that satisfies the (modular) *decryption relation*

$$\mathbf{c}(\mathbf{s}) := c_0 + \langle \mathbf{c}_1, \mathbf{s} \rangle = \mu + e \pmod{\mathfrak{p}} \tag{3.1}$$

where the error $e \in K$ is small enough to be decodable mod R (see Section 2.3).

Importantly, a BFV ciphertext can be seen as defining an affine linear form $\mathbf{c}(S) := c_0 + \langle \mathbf{c}_1, S \rangle$ in the unknown secret key $\mathbf{s}$, as represented by the formal variable S. So to decrypt using $\mathbf{s}$, one simply computes $\mathbf{c}(\mathbf{s}) \in K/\mathfrak{p}$ and rounds it off (or decodes) to $\mu \in R/\mathfrak{p}$ (see Section 3.3 for the formalization).

Because $\mathbf{s}$ is over R, the entries of $\mathbf{c}$ can be lifted to short representatives $\hat{\mathbf{c}}$ (see Section 2.3) without affecting the decryption relation. Because $\mathbf{s}$ is also small, this makes $\hat{\mathbf{c}}(\mathbf{s}) \in K$ proportionately small as well, so the decryption relation can be rewritten as the (non-modular) equation

$$\hat{\mathbf{c}}(\mathbf{s}) = \hat{\mu} + e \in K \tag{3.2}$$

for some relatively small $\hat{\mu} \in R$ for which $\mu = \hat{\mu} \bmod \mathfrak{p}$. More concretely, the norm of $\hat{\mu}$ is proportional to the norms of $\mathbf{s}$ and of a typical mod-$\mathfrak{p}$ representative. This non-modular form of the decryption relation is important for bounding the error growth in homomorphic multiplication; see Section 3.4 for details.

Discretization and switching. Abstractly, the entries of a ciphertext are in K (modulo $\mathfrak{p}$), but in an implementation they must be discretized to have suitable finite precision. As shown below in Section 3.4, the size of the error e (relative to R) in a ciphertext $\mathbf{c}$ determines the amount of homomorphic computation that can be performed on it, so $\mathbf{c}$ must have high enough precision to support a small enough error e. More specifically, we take the entries of $\mathbf{c}$ to be in $\Gamma/\mathfrak{p}$ for some R-lattice $\Gamma \supseteq \mathfrak{p}$, called the *discretization lattice*, which must be fine enough for the error to be as small as needed.

As the error grows due to homomorphic operations, the ciphertext can be "rounded off" to have correspondingly less precision, i.e., to a coarser discretization lattice. We call this *discretization switching*; it is the analog of "modulus switching" in prior treatments of BFV/BGV, but at a different scale. To switch $\mathbf{c} = (c_0, \mathbf{c}_1)$ from Γ to some $\Gamma' \supseteq \mathfrak{p}$, reduce its entries modulo Γ' and lift the resulting cosets to short representatives (using a short basis of Γ' as in Section 2.3), obtaining $\boldsymbol{\delta} = (\delta_0, \boldsymbol{\delta}_1) \in K^{1+k}$, and output $\mathbf{c}' = \mathbf{c} - \boldsymbol{\delta} = (c_0 - \delta_0, \mathbf{c}_1 - \boldsymbol{\delta}_1) \in (\Gamma'/\mathfrak{p})^{1+k}$. This $\mathbf{c}'$ has error

$$e' = e - (\delta_0 + \langle \boldsymbol{\delta}_1, \mathbf{s} \rangle) \; . \tag{3.3}$$

Thus, a ciphertext may be switched to any sufficiently fine Γ' for which the additional error remains tolerable. Typically, Γ' is chosen so that the switching error is significantly smaller than e.

3.3 Encryption and Decryption

3.3.1 Decryption

Following the decryption relation for ciphertexts (Equation (3.1)), the decryption algorithm works as follows:

1. compute $d := \mathbf{c}(\mathbf{s}) = \mu + e \in K/\mathfrak{p}$,
2. "decode" $d = e \pmod R$ to get $e \in K$ (see Section 2.3) and output $\mu = d - e \in R/\mathfrak{p}$.

3.3.2 Encryption

Symmetric encryption of a plaintext $\mu \in R/\mathfrak{p}$ under a secret key $\mathbf{s}$ works as follows.[11] Fix a sufficiently fine R-lattice $\Gamma \supseteq R$ in K; a common choice is $\Gamma = h^{-1}R$ for some suitable nonzero $h \in R$.

1. Sample a (decodable mod R) error $e \in \Gamma$ from an appropriate (Module-)LWE error distribution.
2. Choose uniformly random $\mathbf{c}_1 \leftarrow (\Gamma/\mathfrak{p})^k$.
3. Let $c_0 = \mu + e - \langle \mathbf{c}_1, \mathbf{s} \rangle \in \Gamma/\mathfrak{p}$.
4. Output the ciphertext $\mathbf{c} = (c_0, \mathbf{c}_1) \in (\Gamma/\mathfrak{p})^{1+k}$.[12]

By inspection, the output satisfies the decryption relation (Equation (3.1)).

Security from Module-LWE. It is immediate that this encryption is CPA secure, assuming the hardness of the corresponding decisional Module-LWE problem. More specifically, the assumption is that for the random choice of secret key $\mathbf{s}$, independent samples of the form $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in (\Gamma/\mathfrak{p})^{k+1}$, where each $\mathbf{a} \leftarrow (\Gamma/\mathfrak{p})^k$ is uniform and each $e \in \Gamma$ is drawn from the error distribution, are computationally indistinguishable from uniform. (Each ciphertext corresponds to one such sample, with $\mathbf{c}_1 = -\mathbf{a}$ and $c_0 = \mu + b$, so it is

[11] As is standard, asymmetric encryption can be done similarly, using a public key consisting of an encryption of zero.

[12] In an implementation, for security it is vital that the output ciphertext be represented by *distinguished representatives* modulo $\mathfrak{p}$.

also pseudorandom.) For the common choice $\Gamma = h^{-1}R$, rescaling the samples by h makes this assumption equivalent to its more familiar form over $R/h\mathfrak{p}$, that samples $(\mathbf{a} \leftarrow (R/h\mathfrak{p})^k, b = \langle \mathbf{a}, \mathbf{s} \rangle + he) \in (R/h\mathfrak{p})^{k+1}$ are pseudorandom.

We emphasize that the "error rate" of this Module-LWE problem is the size of the error e *relative to the modulus* $\mathfrak{p}$*, not* R. So, for a very "sparse" ideal $\mathfrak{p}$, the error rate is necessarily very small, which requires a correspondingly large LWE dimension for security.

3.4 Homomorphic Operations

The BFV cryptosystem supports the following homomorphic operations on ciphertexts.

3.4.1 Addition and Multiplication

Given ciphertexts $\mathbf{c}, \mathbf{c}' \in (K/\mathfrak{p})^{1+k}$ that respectively encrypt $\mu, \mu' \in R/\mathfrak{p}$ with errors $e, e' \in K$ under the same secret key $\mathbf{s}$, their homomorphic sum is defined as

$$\mathbf{c}_+(S) := \mathbf{c}(S) + \mathbf{c}'(S) = (c_0 + c'_0, \mathbf{c}_1 + \mathbf{c}'_1) \in (K/\mathfrak{p})^{1+k} .$$

As long as $e_+ := e + e'$ is decodable mod R, this is correct—i.e., it satisfies the decryption relation for plaintext $\mu + \mu' \in R/\mathfrak{p}$—because by linearity,

$$\mathbf{c}_+(\mathbf{s}) = \mathbf{c}(\mathbf{s}) + \mathbf{c}'(\mathbf{s}) = (\mu + \mu') + e_+ \pmod{\mathfrak{p}} .$$

Homomorphic multiplication of ciphertexts $\mathbf{c}, \mathbf{c}'$ works as follows:

1. First lift $\mathbf{c}, \mathbf{c}'$ to short mod-$\mathfrak{p}$ representatives $\hat{\mathbf{c}}, \hat{\mathbf{c}}' \in K^{1+k}$, which by Equation (3.2) encrypt some fairly small $\hat{\mu}, \hat{\mu}' \in R$ that are congruent to $\mu, \mu' \in R/\mathfrak{p}$ (respectively).[13]

2. Then define their *intermediate* homomorphic product as

$$\hat{\mathbf{c}}_\times = (\hat{c}_{\times,0}, \hat{\mathbf{c}}_{\times,1}, \hat{\mathbf{c}}_{\times,2}) := (\hat{c}_0 \cdot \hat{c}'_0 \ , \ \hat{c}_0 \cdot \hat{\mathbf{c}}'_1 + \hat{\mathbf{c}}_1 \cdot \hat{c}'_0 \ , \ \hat{\mathbf{c}}_1 \otimes \hat{\mathbf{c}}'_1) \in K \times K^k \times K^{k^2} ,$$

   which represents the affine *quadratic* form $\hat{\mathbf{c}}_\times(S) := \hat{\mathbf{c}}(S) \cdot \hat{\mathbf{c}}'(S) = \hat{c}_{\times,0} + \langle \hat{\mathbf{c}}_{\times,1}, S \rangle + \langle \hat{\mathbf{c}}_{\times,2}, S \otimes S \rangle$. Reduce this modulo $\mathfrak{p}$ to get $\mathbf{c}_\times(S) := \hat{\mathbf{c}}_\times(S) \bmod \mathfrak{p}$.

By linearity and the mixed-product property, we can see that

$$\hat{\mathbf{c}}_\times(\mathbf{s}) = \hat{\mathbf{c}}(\mathbf{s}) \cdot \hat{\mathbf{c}}'(\mathbf{s}) = (\hat{\mu} + e) \cdot (\hat{\mu}' + e') = \hat{\mu}\hat{\mu}' + \underbrace{(\hat{\mu}e' + e\hat{\mu}' + ee')}_{e_\times} \in K .$$

Hence $\mathbf{c}_\times(\mathbf{s}) = \mu\mu' + e_\times \pmod{\mathfrak{p}}$, so $\mathbf{c}_\times$ "decrypts" to $\mu\mu' \in R/\mathfrak{p}$ (via a quadratic evaluation at $\mathbf{s}$), as long as $e_\times$ is decodable mod R. Indeed, $e_\times$ is fairly small because $\hat{\mu}, \hat{\mu}'$ are fairly small and e, e' are small. More concretely, the error in $\mathbf{c}_\times$ is roughly $\hat{\mu} + \hat{\mu}'$ larger than the errors in the input ciphertexts. So, the error growth under homomorphic multiplication is proportional to the size of typical mod-$\mathfrak{p}$ representatives.

Finally, the quadratic component $\mathbf{c}_{\times,2} \in (K/\mathfrak{p})^{k^2}$ of $\mathbf{c}_\times(S)$, which corresponds to the self-tensored secret key $\mathbf{s} \otimes \mathbf{s}$, can be "linearized" via key-switching, using a suitable encryption of $\mathbf{s} \otimes \mathbf{s}$ under $\mathbf{s}$. (See Section 3.4.2 below.) When the result of this key-switching is added to the affine linear part $(c_{0,\times}, \mathbf{c}_{1,\times}) \in (K/\mathfrak{p})^{1+k}$ of $\mathbf{c}_\times(S)$, this yields an affine linear ciphertext in $(K/\mathfrak{p})^{1+k}$ that encrypts $\mu\mu'$ under $\mathbf{s}$.

[13] Lifting the ciphertext from $K/\mathfrak{p}$ to K is important because multiplication on $K/\mathfrak{p}$ is not well defined, since the product of an element of K and an element of $\mathfrak{p}$ is not necessarily in $\mathfrak{p}$.

3.4.2 Key Switching and Linearization

Key switching [BV11b] is a general technique that converts a ciphertext encrypted under one secret key into a ciphertext encrypted under another secret key, using a suitable encryption of the former key under the latter. In particular, it can be used to "linearize" the quadratic part of the ciphertext that results from homomorphic multiplication as above. In our context, it uses the "digit representation" of elements in $K/\mathfrak{p}$ as "small" polynomials over R from Appendix D, which we first briefly summarize.

Fix an integer base $g \geq 2$, a positive integer precision parameter ℓ, and a positive integer e such that $g^e \in \mathfrak{p}$; for the following construction, represent the relevant elements of $K/\mathfrak{p}$ via $g^{-\ell}R/g^eR$. Then we can express any such a as a Laurent polynomial $\mathbf{a}(G) \in R[G, G^{-1}]$ supported on degrees $-e < i \leq \ell$, with small coefficients and satisfying $\mathbf{a}(g^{-1}) = a \pmod{\mathfrak{p}}$. Equivalently, $\langle \mathbf{g}^{-1}(a), \mathbf{g} \rangle = a$, where $\mathbf{g}^{-1}(a)$ is the short coefficient vector over R of $\mathbf{a}(G)$, and $\mathbf{g}$ is the vector over $K/\mathfrak{p}$ consisting of the corresponding powers of g in the evaluation of $\mathbf{a}(g^{-1})$. More generally, for any vectors $\mathbf{u}, \mathbf{v}$ over K, by linearity we have that $\langle \mathbf{g}^{-1}(\mathbf{u}), \mathbf{v} \otimes \mathbf{g} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle$, where $\mathbf{g}^{-1}$ is applied entry-wise to $\mathbf{u}$.

Key switching. Now let vectors $\mathbf{s}', \mathbf{s}$ over R respectively be "source" and "target" secret keys, which need not have the same dimension, and may be related. An $\mathbf{s}'$-to-$\mathbf{s}$ *key-switching hint* $\mathbf{H}(S)$ is essentially a tuple of affine-in-S ciphertexts, which can be seen as a matrix $\mathbf{H} = [\mathbf{h}_0 \mid \mathbf{H}_1]$ over $K/\mathfrak{p}$ whose rows are the ciphertexts, which satisfies

$$\mathbf{H}(\mathbf{s}) = \mathbf{h}_0 + \mathbf{H}_1 \cdot \mathbf{s} = \mathbf{s}' \otimes \mathbf{g} + \mathbf{e} \pmod{\mathfrak{p}}$$

for some small error $\mathbf{e}$ over K. In other words, it is essentially an encryption of $\mathbf{s}' \otimes \mathbf{g}$ under $\mathbf{s}$. Note that this "plaintext" is over $K/\mathfrak{p}$, not $R/\mathfrak{p}$, but the hint can still be generated in the usual way, using Module-LWE samples having at least as much precision as $\mathbf{s}' \otimes \mathbf{g}$ has (in order to fully conceal it). For security, a circularity assumption may be needed if $\mathbf{s}$ and $\mathbf{s}'$ are dependent.

Now consider a vector $\mathbf{u}'$ over $K/\mathfrak{p}$ that represents a (non-affine) linear form $\mathbf{u}'(S') := \langle \mathbf{u}', S' \rangle$. The key-switching operation replaces this with

$$\mathbf{u}(S) := \langle \mathbf{g}^{-1}(\mathbf{u}'), \mathbf{H}(S) \rangle = \mathbf{g}^{-1}(\mathbf{u}')^t \cdot \mathbf{H}(S) \ .$$

Observe that because $\mathbf{g}^{-1}(\mathbf{u}')$ is over R and $\mathbf{H}(S)$ is over the R-module $K/\mathfrak{p}$,

$$\mathbf{u}(\mathbf{s}) = \langle \mathbf{g}^{-1}(\mathbf{u}'), \mathbf{H}(\mathbf{s}) \rangle = \langle \mathbf{g}^{-1}(\mathbf{u}'), \mathbf{s}' \otimes \mathbf{g} + \mathbf{e} \rangle = \mathbf{u}'(\mathbf{s}') + \langle \mathbf{g}^{-1}(\mathbf{u}'), \mathbf{e} \rangle \pmod{\mathfrak{p}} \ .$$

That is, $\mathbf{u}(S)$ evaluates under $\mathbf{s}$ to whatever $\mathbf{u}'(S')$ evaluates to under $\mathbf{s}'$, up to the small extra additive error $\langle \mathbf{g}^{-1}(\mathbf{u}'), \mathbf{e} \rangle$. The size of this extra error is proportional to the sizes of the error $\mathbf{e}$ in the key-switching hint and of the short vector $\mathbf{g}^{-1}(\mathbf{u}')$.

Linearization. We use key switching to linearize the quadratic part of a product ciphertext $\mathbf{c}_\times(S) = c_{\times,0} + \langle \mathbf{c}_{\times,1}, S \rangle + \langle \mathbf{c}_{\times,2}, S \otimes S \rangle$ as follows. We view the quadratic component $\mathbf{c}_{\times,2}(S) = \langle \mathbf{c}_{\times,2}, S \otimes S \rangle$ as a *linear* form in $S' = S \otimes S$, which would be applied on $\mathbf{s}' = \mathbf{s} \otimes \mathbf{s}$. So, using an $\mathbf{s}'$-to-$\mathbf{s}$ key-switching hint (prepared and published in advance), we switch from the linear form $\mathbf{u}(S') := \mathbf{c}_{\times,2}(S')$ to an affine one in S, and add it to the remaining affine part $c_{\times,0} + \langle \mathbf{c}_{\times,1}, S \rangle$. The resulting ciphertext is affine in S, and evaluates (under $\mathbf{s}$) to whatever the quadratic ciphertext $\mathbf{c}_\times(S)$ evaluates to, up to the small additive error from key switching.

3.4.3 Plaintext-Modulus Lifting and Division

Our presentation of the BFV scheme easily supports lifting to a "larger" plaintext modulus, as well as homomorphic division by a public divisor, when both the plaintext and its modulus are divisible by that divisor. These operations are useful in BFV bootstrapping (see Section 5).

For lifting, let $\mathfrak{p} \mid \mathfrak{p}'$, i.e., $\mathfrak{p}' \subseteq \mathfrak{p}$. Then we can directly lift from plaintext modulus $\mathfrak{p}$ to $\mathfrak{p}'$: given a BFV ciphertext $\mathbf{c}$ over $K/\mathfrak{p}$ that encrypts $\mu \in R/\mathfrak{p}$, lifting it to an arbitrary $\mathbf{c}'$ over $K/\mathfrak{p}'$ such that $\mathbf{c}' = \mathbf{c} \pmod{\mathfrak{p}}$ yields a ciphertext that encrypts (with the same error) some plaintext $\mu' \in R/\mathfrak{p}'$ for which $\mu' = \mu \bmod \mathfrak{p}$. This follows immediately from the decryption relation (Equation (3.1)) and the fact that $\mathbf{s}$ is over R.

For division, suppose that $\mathfrak{a} \mid \mathfrak{p}$ where $\mathfrak{a} = aR$ for some nonzero $a \in R$. (A common choice in practice is $a = 2$.) Then when the plaintext is known to be divisible by a (i.e., an element of $\mathfrak{a}$), we can homomorphically divide both it and the modulus by a, simply by dividing the ciphertext by a. This also has the effect of dividing the error by a.

Lemma 3.2. *Suppose that BFV ciphertext $\mathbf{c}$ (over $K/\mathfrak{p}$) encrypts plaintext $\mu \in \mathfrak{a}/\mathfrak{p}$ with error e under secret key $\mathbf{s}$. Then $a^{-1} \cdot \mathbf{c}$ (which is modulo $\mathfrak{a}^{-1}\mathfrak{p}$) encrypts $a^{-1}\mu \in R/(\mathfrak{a}^{-1}\mathfrak{p})$ with error $a^{-1} \cdot e$ under $\mathbf{s}$.*

Proof. This follows immediately from the decryption relation (Equation (3.1)). □

3.4.4 Automorphisms

The cyclotomic number field $K = \mathbb{Q}(\zeta_m)$ has a full complement of $n = \deg(K/\mathbb{Q}) = \varphi(m)$ *automorphisms*, i.e., ring isomorphisms from K to itself. Concretely, they are induced by mapping ζ_m to ζ_m^i, for any $i \in \mathbb{Z}_m^*$. The FHE literature uses the *homomorphic* evaluation of automorphisms for a wide variety of purposes.

Our presentation of the BFV scheme supports this for any automorphism τ of K that fixes the plaintext modulus ideal $\mathfrak{p}$, i.e., $\tau(\mathfrak{p}) = \mathfrak{p}$. In particular, if $\mathfrak{p} = pR$ for an integer $p \in \mathbb{Z}$ then this condition holds for *every* automorphism τ, because τ fixes $\mathbb{Z}$ pointwise. In general, if K' is a subfield of K with ring of integers R', and $\mathfrak{p} = pR$ for some $p \in R'$, then the condition holds for any automorphism of K that fixes K' pointwise, i.e., one in the Galois group of K/K'.

Given a ciphertext $\mathbf{c} = (c_0, \mathbf{c}_1) \in (K/\mathfrak{p})^{1+k}$ that encrypts $\mu \in R/\mathfrak{p}$ with error $e \in K$ under a secret key $\mathbf{s}$, we first let $\mathbf{c}' = \tau(\mathbf{c}) \in (K/\mathfrak{p})^{1+k}$, applying τ component-wise. Letting $\mathbf{s}' = \tau(\mathbf{s})$, because τ is a ring homomorphism, and $\tau(\mathfrak{p}) = \mathfrak{p}$, we have that

$$\mathbf{c}'(\mathbf{s}') = \tau(c_0) + \langle \tau(\mathbf{c}_1), \tau(\mathbf{s}) \rangle = \tau(c_0 + \langle \mathbf{c}_1, \mathbf{s} \rangle) = \tau(\mu) + \tau(e) \pmod{\mathfrak{p}} .$$

That is, $\mathbf{c}'$ encrypts $\tau(\mu)$ with error $\tau(e)$ under secret key $\mathbf{s}' = \tau(\mathbf{s})$. (Note that the canonical embedding of $\tau(e)$ is just a permutation of that of e, so they have the same canonical norms.) So, using an $\mathbf{s}'$-to-$\mathbf{s}$ key-switching hint (prepared in advance), we can switch $\mathbf{c}'$ to a ciphertext that encrypts $\tau(\mu)$ under $\mathbf{s}$, with error $\tau(e)$ plus the small additive error from key switching.

4 Decomposed BFV for High-Precision Plaintexts

Recall from Section 3 that in the BFV scheme, the inverse error rate (relative to the plaintext modulus $\mathfrak{p}$) of fresh ciphertexts, and the error growth factor under homomorphic multiplication, are both proportional to the sparsity of the modulus $\mathfrak{p}$, i.e., the size of a typical mod-$\mathfrak{p}$ representative. So, for very sparse moduli, like $\mathfrak{p} = pR$ for some large p like 2^{64}, supporting even moderate-depth homomorphic multiplication

requires starting from ciphertexts with very tiny error rates, which necessitates large LWE dimensions and high-precision numbers, significantly harming efficiency.

In this section we show how to get homomorphic encryption for high-precision plaintexts with much better initial error rates and error growth. More specifically, we show how to get homomorphic mod-p plaintext arithmetic essentially via d BFV ciphertexts, with an error profile matching or beating that of BFV with modulus $d \cdot p^{1/d}$. We call this collection of techniques the *decomposed BFV* (dBFV) scheme, because it attains these properties via "digit decomposed" plaintexts and ciphertexts.

4.1 Plaintext Digit Decomposition

For the remainder of the section, fix some relatively small base $b \in R$ defining the ideal $\mathfrak{b} = bR$. The first key idea behind the dBFV scheme is to replace the plaintext quotient ring $R/\mathfrak{p}$ with the *isomorphic* quotient ring $R[B]/\mathcal{L}$, where $\mathcal{L}$ is the $R[B]$-ideal of polynomials that evaluate at $B = b$ to zero modulo $\mathfrak{p}$:[14]

$$\mathcal{L} = \mathcal{L}_{\mathfrak{p}}^{\perp}(b) := \{\mathbf{z}(B) \in R[B] : \mathbf{z}(b) \in \mathfrak{p}\} \supseteq \mathfrak{p}[B] \; . \tag{4.1}$$

This is an infinite-dimensional lattice (more precisely, R-module), but we will usually deal with polynomials of some bounded degree $< d$ in B, in which case we use the finite-dimensional restriction $\mathcal{L}^{<d} := \mathcal{L} \cap R^{<d}[B]$. The evaluation-at-$b$ map from $R[B]$ to $R/\mathfrak{p}$ is clearly a surjective ring homomorphism with kernel $\mathcal{L}$, so it induces a ring isomorphism from $R[B]/\mathcal{L}$ to $R/\mathfrak{p}$ (and an R-module isomorphism from $R^{<d}[B]/\mathcal{L}^{<d}$ to $R/\mathfrak{p}$); the inverse is just the natural reduction-mod-$\mathcal{L}$ map.[15] Also observe that the natural map from $K^{<d}[B]/\mathcal{L}^{<d}$ to $K^{<d'}[B]/\mathcal{L}^{<d'}$ for $d \le d'$, or to $K[B]/\mathcal{L}$, is an injective R-module homomorphism.

Definition 4.1 (Plaintext digit decomposition). A base-b *digit decomposition* of a plaintext $\mu \in R/\mathfrak{p}$ is any polynomial $\boldsymbol{\mu}(B) \in R[B]$ satisfying $\mu = \boldsymbol{\mu}(b) \bmod \mathfrak{p}$; equivalently, it is any element of the coset $\mu + \mathcal{L} \subseteq R[B]$. The coefficients of $\boldsymbol{\mu}(B)$ are called the *digits* of the decomposition.

Clearly, a digit decomposition is not unique, though we could define unique *small* decompositions, as the distinguished representatives of $R[B]/\mathcal{L}$ using a short R-basis of $\mathcal{L}$. However, to support natural and efficient homomorphic operations, in dBFV we will instead need to use more "relaxed" decompositions that are typically not distinguished representatives, but are still proportionately small.

Notice that digit decompositions are additively and multiplicatively homomorphic, simply by the evaluation-at-b ring isomorphism $R[B]/\mathcal{L} \cong R/\mathfrak{p}$. That is, if $\boldsymbol{\mu}(B), \boldsymbol{\mu}'(B)$ are decompositions of $\mu, \mu' \in R/\mathfrak{p}$ (respectively), then $\boldsymbol{\mu}_+(B) := \boldsymbol{\mu}(B) + \boldsymbol{\mu}'(B)$ and $\boldsymbol{\mu}_\times(B) := \boldsymbol{\mu}(B) \cdot \boldsymbol{\mu}'(B)$ are respectively decompositions of $\mu + \mu', \mu \cdot \mu' \in R/\mathfrak{p}$. More interestingly, in Section 4.4 below we will see a stronger property: that for *short* digit decompositions (having small coefficients), even "noisy" ones are additively and multiplicatively homomorphic, with some increase in the size of the noise.

Of course, the *degree* of $\boldsymbol{\mu}_\times(B)$ is typically larger than those of $\boldsymbol{\mu}(B), \boldsymbol{\mu}'(B)$, and similarly for the *sizes* of both $\boldsymbol{\mu}_+(B)$ and $\boldsymbol{\mu}_\times(B)$. At the ciphertext level, we address these issues using degree reduction to bound the degree and lifting to short representatives to control the size, both without changing the plaintext; see Section 4.6.

[14] We note that $\mathcal{L}_{\mathfrak{p}}^{\perp}(b)$ is essentially identical to the primitive lattice for the powers-of-b "gadget vector" in [MP12], but expressed in terms of polynomials rather than their coefficient vectors, and not limited to any particular degree.

[15] Similarly, evaluation at b induces an R-module homomorphism from $K[B]/\mathcal{L}$ (or $K^{<d}[B]/\mathcal{L}^{<d}$) to $K/\mathfrak{p}$. However, we caution that (except for $d = 1$) this is *not injective* and hence *not an isomorphism*, because $\mathcal{L} \subseteq R[B]$ (respectively, $\mathcal{L}^{<d}$) is not the entire kernel of the map. In the other direction, the natural map from K to $K[B]/\mathcal{L}$ (or $K^{<d}[B]/\mathcal{L}^{<d}$) trivially has kernel $\mathfrak{p}$, so it induces an injective R-module homomorphism from $K/\mathfrak{p}$.

Remark 4.2 (Decompositions and gadgets). The decomposition from Definition 4.1 corresponds to a gadget decomposition with respect to the powers-of-b gadget vector $\mathbf{b} = (1, b, b^2, \ldots)$ over $R/\mathfrak{p}$. As noted in [MP12] and elsewhere, there are other kinds of gadget vectors $\mathbf{g} \in (R/\mathfrak{p})^d$, which define R-module isomorphisms between $R/\mathfrak{p}$ and $R^d/\mathcal{L}^\perp_\mathfrak{p}(\mathbf{g})$ where $\mathcal{L}^\perp_\mathfrak{p}(\mathbf{g}) := \{\mathbf{z} \in R^d : \langle \mathbf{z}, \mathbf{g} \rangle = 0 \pmod{\mathfrak{p}}\}$, and which have efficient algorithms for lifting to short representatives. For example, when $\mathfrak{p}$ factors into "geometrically nice" pairwise coprime ideals, there is such a gadget based on the Chinese Remainder Theorem. This kind of CRT-gadget decomposition is homomorphic under *coordinate-wise* addition and multiplication, and hence the same goes for *short, "noisy"* CRT decompositions. The dBFV cryptosystem easily adapts to use this or any other suitable gadget decomposition, and can become even more efficient in such cases.

4.2 Ciphertexts

Essentially, a dBFV ciphertext can be seen as a *tuple* of BFV ciphertexts that encrypt the digits of a base-b digit decomposition of the plaintext $\mu \in R/\mathfrak{p}$. Formally, it is defined as follows.

Definition 4.3 (dBFV ciphertext). A *dBFV ciphertext* that encrypts a plaintext $\mu \in R/\mathfrak{p}$ under a secret key $\mathbf{s} \in R^k$ is a (typically affine linear in S) polynomial $\mathbf{c}(B, S) \in (K[B]/\mathcal{L})[S]$ that satisfies the (modular) decryption relation

$$\mathbf{c}(B, \mathbf{s}) = \boldsymbol{\mu}(B) + \mathbf{e}(B) = \mu + \mathbf{e}(B) \pmod{\mathcal{L}} \tag{4.2}$$

where $\boldsymbol{\mu}(B) \in \mu + \mathcal{L} \subseteq R[B]$ is any base-b decomposition of μ (Definition 4.1), and $\mathbf{e}(B) \in K[B]$ has coefficients that are decodable mod R.

Concretely, any specific dBFV ciphertext is actually over $K^{<d}[B]/\mathcal{L}^{<d}$ for some degree bound d, which may vary from one ciphertext to another; the special case $d = 1$ is ordinary BFV (at our "scaling"), because $K^{<1}/\mathcal{L}^{<1} = K/\mathfrak{p}$. Notice that we can increase the degree bound via the natural embedding into $K^{<d'}[B]/\mathcal{L}^{<d'}$ for any $d' \geq d$. For a convenient unified definition, we view a generic dBFV ciphertext as being over $K[B]/\mathcal{L}$ via the natural embedding.

Remark 4.4. Here are some useful observations about dBFV ciphertexts and their relationship to BFV ciphertexts.

1. If $\boldsymbol{\mu}(B)$ is a digit decomposition of a plaintext μ, then the constant-in-S polynomial $\mathbf{c}(B, S) = \boldsymbol{\mu}(B) \bmod \mathcal{L}$ is a dBFV ciphertext that (insecurely) encrypts μ, under *any* secret key, with zero error.
2. If $\mathbf{c}(S)$ over $K/\mathfrak{p}$ is a BFV ciphertext encrypting μ with error e under secret key $\mathbf{s}$, then $\mathbf{c}(B, S) := \mathbf{c}(S) \bmod \mathcal{L}$ is a dBFV ciphertext encrypting μ with error $\mathbf{e}(B) = e$ under $\mathbf{s}$. This is because $\mathbf{c}(B, \mathbf{s}) = \mathbf{c}(\mathbf{s}) = \mu + \mathbf{e}(B) \pmod{\mathcal{L}}$ and every coefficient of $\mathbf{e}(B)$ is decodable mod R. (Indeed, every non-constant coefficient of $\mathbf{e}(B)$ is zero.)
3. Because $\mathfrak{p}[B] \subseteq \mathcal{L}$, we can lift a dBFV ciphertext to some $\hat{\mathbf{c}}(B, S) = \sum_i \hat{\mathbf{c}}_i(S) \cdot B^i \in (K/\mathfrak{p})[B, S]$, and view it as a polynomial in B whose coefficients $\hat{\mathbf{c}}_i(S) \in (K/\mathfrak{p})[S]$ are polynomials in S. Then each $\hat{\mathbf{c}}_i(S)$ may be seen as a BFV encryption of the ith digit (modulo $\mathfrak{p}$) of some decomposition of μ.
4. A dBFV ciphertext is typically affine linear in S. In this case, it can be expressed as a pair $(\mathbf{c}_0(B), \mathbf{c}_1(B)) \in (K[B]/\mathcal{L})^{1+k}$ defining $\mathbf{c}(B, S) = \mathbf{c}_0(B) + \langle \mathbf{c}_1(B), S \rangle$.

Discretization lattices. As in BFV, the coefficients of a concrete dBFV ciphertext of degree $< d$ in B are taken from $\Gamma/\mathcal{L}^{<d}$ for some sufficiently fine R-lattice $\Gamma \supseteq \mathcal{L}^{<d}$ in $K^{<d}[B] \cong K^d$, again called the *discretization lattice*. Switching to a different discretization lattice generalizes in the natural way, with an additive error analogous to the one in Equation (3.3). Also as in BFV, preserving uniformity is required when switching is used in an LWE hardness reduction, but not inherently when reducing the precision of an existing ciphertext.

4.2.1 Decryption

To decrypt a dBFV ciphertext $\mathbf{c}(B, S)$ using the secret key $\mathbf{s}$, following Equation (4.2), simply evaluate

$$\mathbf{d}(B) := \mathbf{c}(B, \mathbf{s}) = \boldsymbol{\mu}(B) + \mathbf{e}(B) \in K[B]/\mathcal{L}\,,$$

decode $\mathbf{d}(B) = \mathbf{e}(B) \pmod{R[B]}$ coefficient-wise to recover $\mathbf{e}(B) \in K[B]$ and thereby $\boldsymbol{\mu}(B) = \mathbf{d}(B) - \mathbf{e}(B) \in R[B]/\mathcal{L}$, then output $\mu = \boldsymbol{\mu}(b) \bmod \mathfrak{p}$.

We caution that evaluating $\mathbf{c}(b, \mathbf{s}) = \boldsymbol{\mu}(b) + \mathbf{e}(b)$, and then attempting to decode it, typically *does not work*. This is because the evaluated noise polynomial $\mathbf{e}(b) \in K$ is too large to decode relative to R, since the small coefficients of $\mathbf{e}(B)$ are multiplied by large powers of b. Instead, it is essential to decode (i.e., error correct) the individual coefficients of $\mathbf{c}(B, \mathbf{s})$ to R first, before evaluating at $B = b$.

4.2.2 Lifting

Analogously to a BFV ciphertext over $K/\mathfrak{p}$, we can view a dBFV ciphertext $\mathbf{c}(B, S)$ as a (typically affine linear) polynomial in S with coefficients from the R-module $K[B]/\mathcal{L}$. Because $\mathbf{s}$ is over R, lifting the coefficients of $\mathbf{c}$ to short representatives in $K[B]$ preserves Equation (4.2) and yields a small $\hat{\mathbf{c}}(B, S) \in K[B, S]$. (The details of this lifting are given below in Section 4.6.) Because $\mathbf{s}$ is also small, $\hat{\mathbf{c}}(B, \mathbf{s})$ is proportionately small, so the decryption relation can be rewritten as the (non-modular) equation

$$\hat{\mathbf{c}}(B, \mathbf{s}) = \hat{\boldsymbol{\mu}}(B) + \mathbf{e}(B) \in K[B] \tag{4.3}$$

for some comparably small digit decomposition $\hat{\boldsymbol{\mu}}(B) \in R[B]$ of $\mu \in R/\mathfrak{p}$. More concretely, the size of $\hat{\boldsymbol{\mu}}(B)$ is proportional to the norms of $\mathbf{s}$ and of $\hat{\mathbf{c}}$ (see Lemma 4.5 below for a formalization). Similarly to BFV, the smallness of $\hat{\boldsymbol{\mu}}(B)$ is essential for bounding the error growth under homomorphic multiplication (see Section 4.4 below). But in contrast with BFV, the mod-$\mathcal{L}$ representatives used in dBFV can be *much smaller* than the typical mod-$\mathfrak{p}$ representatives used in BFV, which is the source of dBFV's much better error growth.

For now, we prove an elementary bound showing that the plaintext decomposition $\hat{\boldsymbol{\mu}}(B)$ has max norm proportional to those of $\mathbf{s}$ and of the lifted ciphertext $\hat{\mathbf{c}}(B, S)$. (Tighter bounds can be obtained using a finer-grained analysis, special properties of $\mathbf{s}$, or the distribution of $\hat{\mathbf{c}}$ arising from lifting.)

Lemma 4.5. *Let $\hat{\mathbf{c}}(B, S) \in K[B, S]$ be a lifted, affine linear (in S) dBFV ciphertext of degree less than D in B under secret key $\mathbf{s} \in R^k$, in which every coordinate has canonical max norm bounded by some $\beta > 0$. Then its noisy plaintext digit decomposition $\mathbf{d}(B) = \hat{\mathbf{c}}(B, \mathbf{s}) \in K[B]$ has max norm $\|\mathbf{d}(B)\|_\infty \le \beta \cdot (1 + \sum_{j=1}^k \|s_j\|_\infty)$, and operator norm $\|\mathbf{d}(B)\|_{op} \le D \cdot \|\mathbf{d}(B)\|_\infty$.*

Proof. Express $\hat{\mathbf{c}}(B, S) = \sum_i \hat{\mathbf{c}}_i(S) \cdot B^i$ where $\hat{\mathbf{c}}_i(S) = \hat{c}_{i,0} + \langle \hat{\mathbf{c}}_{i,1}, S \rangle \in K[S]$. Then the ith coefficient of $\mathbf{d}(B) = \hat{\mathbf{c}}(B, \mathbf{s}) \in K[B]$ is $\hat{\mathbf{c}}_i(\mathbf{s}) = \hat{c}_{i,0} + \langle \hat{\mathbf{c}}_{i,1}, \mathbf{s} \rangle \in K$. Recall from Section 2.2 that multiplication by $s_j \in R$ enlarges canonical norms by at most $\|s_j\|_\infty$, so by the triangle inequality, each coefficient of $\mathbf{d}(B)$ has canonical max norm at most $\beta \cdot (1 + \sum_j \|s_j\|_\infty)$. Summing over the D monomials in $\mathbf{d}(B)$ yields the claimed bound on $\|\mathbf{d}(B)\|_{\text{op}}$. □

4.3 Encryption

We describe two methods for symmetrically encrypting a plaintext $\mu \in R/\mathfrak{p}$ under a secret key $\mathbf{s} \in R^k$ for dBFV: the first method follows generically from BFV but requires a very small error rate; the second eliminates this drawback using a new LWE variant whose hardness we derive by a nearly lossless reduction from the corresponding standard (plain/Module/Ring) LWE problem. (Public-key encryption can be done analogously.) The first method (and its drawback) is closely analogous to what is done in [CCKS23]; the second method is novel.

Encryption via ordinary BFV. The first method simply encrypts μ under $\mathbf{s}$ as in the BFV scheme (see Section 3.3) to get a mod-$\mathfrak{p}$ BFV ciphertext $\mathbf{c}(S) \in (K/\mathfrak{p})[S]$, then reduces it (coefficient-wise) modulo $\mathcal{L}$ to the dBFV ciphertext $\mathbf{c}(B,S) = \mathbf{c}(S) \bmod \mathcal{L}$, which is in $(K[B]/\mathcal{L})[S]$. Concretely, we would reduce modulo $\mathcal{L}^{<d} \subseteq R^{<d}[B]$ for some arbitrary desired d.[16] This is correct by Item 2 of Remark 4.4, and security follows directly from that of the BFV scheme, and the fact that taking the quotient modulo $\mathcal{L}$ is a public operation that does not use the secret key.

The major drawback of this method is that it is quite wasteful in terms of concrete parameters and security: it relies on the hardness of LWE with a very sparse modulus $\mathfrak{p}$ and small error that must be decodable mod R. So, the error rate relative to the modulus is necessarily very small, hence the numerical precision and LWE dimension must be correspondingly large to ensure sufficient security; see below for further discussion.

Encryption using mod-$\mathcal{L}$-LWE. The second method addresses the above drawback by directly constructing a dBFV ciphertext of desired degree bound $< d$ in B using a mod-$\mathcal{L}$-LWE distribution with secret $\mathbf{s}$. Fix a sufficiently fine discretization lattice $\Gamma \supseteq R^{<d}[B]$ in $K^{<d}[B]$. For the reduction-backed instantiation with $\mathfrak{p} = pR$, take $\Gamma = h^{-1}\mathcal{L}^{<d}$ for a sufficiently large positive integer h divisible by p.

1. Sample an error $\mathbf{e}(B) \in \Gamma$ from an appropriate distribution, with coefficients that are decodable mod R.
2. Choose uniformly random $\mathbf{c}_1(B) \leftarrow (\Gamma/\mathcal{L}^{<d})^k$.
3. Let $\mathbf{c}_0(B) = \mu + \mathbf{e}(B) - \langle \mathbf{c}_1(B), \mathbf{s} \rangle \in \Gamma/\mathcal{L}^{<d}$.
4. Output the ciphertext $\mathbf{c}(B,S) = \mathbf{c}_0(B) + \langle \mathbf{c}_1(B), S \rangle \in (\Gamma/\mathcal{L}^{<d})[S]$.[17]

By inspection, the output satisfies the decryption relation (Equation (4.2)) for plaintext μ.

This encryption is CPA secure assuming the corresponding discretized decisional mod-$\mathcal{L}^{<d}$-LWE problem is hard. More specifically, we assume that for a random secret $\mathbf{s}$, independent samples

$$\big(\mathbf{a},\ \mathbf{b} = \langle \mathbf{a}, \mathbf{s} \rangle + \mathbf{e}\big) \in (\Gamma/\mathcal{L}^{<d})^{k+1},$$

are computationally indistinguishable from uniform, where $\mathbf{a} \leftarrow (\Gamma/\mathcal{L}^{<d})^k$ is uniform and $\mathbf{e}$ is drawn from the chosen error distribution. Each ciphertext corresponds to one such sample, with $\mathbf{c}_1 = -\mathbf{a}$ and $\mathbf{c}_0 = \mu + \mathbf{b}$. For the latter choice of Γ, in Appendix A we tightly reduce ordinary Module-LWE to this problem; under the natural normalizations, the reduction is nearly lossless. Its (invertible) packing step combines d ordinary samples into one K^d-valued sample, preserving their total dimension, and it expands the error by a factor of the spectral norm of a known short basis matrix of $\mathcal{L}^{<d}$. For arbitrary integer p, this factor matches the dimension-normalized sparsity of $\mathcal{L}^{<d}$ to within an $O(\sqrt{d})$ factor; for perfect-power $p = b^d$, it matches up to an additive 1. Thus, the reduction nearly preserves the error rate relative to the modulus lattice.

[16] In the resulting ciphertext, the coefficients of the non-constant-in-B terms are over R, i.e., they have no "fractional" parts and thus no intrinsic error terms. This phenomenon is closely tied to the suboptimal parameters of this first method.

[17] In an implementation, for security it is vital that the output ciphertext be represented by *distinguished representatives* modulo $\mathcal{L}^{<d}$.

Comparison. Compared to encryption via BFV, the mod-$\mathcal{L}$-LWE method provides a better tradeoff between the two main competing considerations for the initial ciphertext error:

- its *size relative to the modulus lattice* ($\mathfrak{p}$ or $\mathcal{L}^{<d}$), which we call the *error rate*, where larger rate improves security and efficiency; and
- its *size relative to the decoding lattice* (R or R^d, respectively), where smaller size improves homomorphic capacity, since the accumulated error must remain decodable modulo that lattice.

The mod-$\mathcal{L}$-LWE method brings these two quantities much closer together, because the modulus lattice $\mathcal{L}^{<d}$ for $d > 1$ is much finer than $\mathfrak{p}$, in a dimension-normalized sense.

More specifically: for the very sparse modulus $\mathfrak{p} = pR \subset R$, any error that is decodable mod R has error rate less than $1/p$ (and must be even smaller to ensure some homomorphic capacity). By comparison, $\mathcal{L}^{<d}$ has the same determinant as $\mathfrak{p}$ but in a d-factor larger dimension, so its dimension-normalized sparsity relative to the decoding lattice R^d is only about $p^{1/d} \approx b$. Thus, for errors of the same size relative to their respective decoding lattices (and thus supporting the same homomorphic capacity), the mod-$\mathcal{L}$-LWE method has a much larger error rate, by about a $p/b \approx p^{1-1/d}$ factor, resulting in significantly better security and efficiency. Alternatively, for the same error rate and LWE dimension (and thus comparable efficiency and security), the mod-$\mathcal{L}$-LWE method uses error that is about a $p^{1-1/d}$ factor smaller relative to its decoding lattice. Since dBFV's error growth under homomorphic multiplication is proportional to $p^{1/d}$, the mod-$\mathcal{L}$-LWE method can support roughly $d - 1$ more layers of multiplication than the BFV-based method.[18]

4.4 Homomorphic Operations

The dBFV cryptosystem supports the following homomorphic operations on ciphertexts. Throughout this subsection, suppose that dBFV ciphertexts $\mathbf{c}(B, S), \mathbf{c}'(B, S) \in (K[B]/\mathcal{L})[S]$ respectively encrypt plaintexts $\mu, \mu' \in R/\mathfrak{p}$ with error polynomials $\mathbf{e}(B), \mathbf{e}'(B)$ under the same secret key $\mathbf{s}$ (see Definition 4.3 and Equation (4.2) in particular).

4.4.1 Addition

The homomorphic sum of $\mathbf{c}, \mathbf{c}'$ is defined as

$$\mathbf{c}_+(B, S) := \mathbf{c}(B, S) + \mathbf{c}'(B, S) \in (K[B]/\mathcal{L})[S] \, .$$

As long as $\mathbf{e}_+(B) := \mathbf{e}(B) + \mathbf{e}'(B) \in K[B]$ is decodable mod R, this satisfies the decryption relation (Equation (4.2)) for plaintext $\mu + \mu' \in R/\mathfrak{p}$ because by linearity,

$$\mathbf{c}_+(B, \mathbf{s}) = \mathbf{c}(B, \mathbf{s}) + \mathbf{c}'(B, \mathbf{s}) = (\mu + \mu') + (\mathbf{e}(B) + \mathbf{e}'(B)) \pmod{\mathcal{L}} \, .$$

The above encompasses homomorphic addition with a *public* plaintext in $R/\mathfrak{p}$ as a special case, because as noted in Item 1 of Remark 4.4, any digit decomposition of a plaintext is a valid dBFV encryption of that plaintext, under *any* secret key (and with zero error).

4.4.2 Multiplication

To homomorphically multiply the ciphertexts $\mathbf{c}, \mathbf{c}'$, we do the following:

1. Lift $\mathbf{c}, \mathbf{c}'$ coefficient-wise to short representatives $\hat{\mathbf{c}}(B, S), \hat{\mathbf{c}}'(B, S) \in K[B, S]$.[19]

[18] This is merely a first-order approximation for large p. Because the error growth under multiplication has additional factors independent of p and d, the improvement will actually be somewhat smaller, and will see diminishing returns as d grows.

[19] Similarly to ordinary BFV, we need to lift the ciphertexts to be over $K[B]$ because $K[B]/\mathcal{L}$ is not closed under multiplication.

2. Define the *intermediate* homomorphic product as

$$\hat{\mathbf{c}}_\times(B,S) := \hat{\mathbf{c}}(B,S) \cdot \hat{\mathbf{c}}'(B,S) \in K[B,S] \ ,$$

and take its quotient modulo $\mathcal{L}$, obtaining the intermediate ciphertext $\mathbf{c}_\times(B,S) = \hat{\mathbf{c}}_\times(B,S) \bmod \mathcal{L}$.

The degrees in B and S of $\mathbf{c}_\times$ are typically larger than those of $\mathbf{c}, \mathbf{c}'$; we next reduce the degree in B and then linearize in S, as described below. (Just as with addition, the above encompasses homomorphic multiplication by a *public* plaintext in $R/\mathfrak{p}$ as a special case, by Item 1 of Remark 4.4.)

Lemma 4.6. *The intermediate ciphertext $\mathbf{c}_\times(B,S)$ encrypts $\mu \cdot \mu' \in R/\mathfrak{p}$ under $\mathbf{s}$, with error polynomial*

$$\mathbf{e}_\times(B) = \hat{\mathbf{c}}(B,\mathbf{s}) \cdot \mathbf{e}'(B) + \mathbf{e}(B) \cdot \hat{\mathbf{c}}'(B,\mathbf{s}) - \mathbf{e}(B) \cdot \mathbf{e}'(B) \in K[B] \ .$$

Proof. Let $\hat{\boldsymbol{\mu}}(B), \hat{\boldsymbol{\mu}}'(B) \in R[B]$ be the digit decompositions encrypted by $\hat{\mathbf{c}}, \hat{\mathbf{c}}'$ (respectively) according to Equation (4.3), so that $\hat{\mathbf{c}}(B,\mathbf{s}) = \hat{\boldsymbol{\mu}}(B) + \mathbf{e}(B)$ and similarly for $\hat{\boldsymbol{\mu}}'(B)$. Expanding the intermediate product,

$$\begin{aligned}\hat{\mathbf{c}}_\times(B,\mathbf{s}) &= \hat{\mathbf{c}}(B,\mathbf{s}) \cdot \hat{\mathbf{c}}'(B,\mathbf{s}) \\ &= \hat{\boldsymbol{\mu}}(B) \cdot \hat{\boldsymbol{\mu}}'(B) + \hat{\boldsymbol{\mu}}(B) \cdot \mathbf{e}'(B) + \mathbf{e}(B) \cdot \hat{\boldsymbol{\mu}}'(B) + \mathbf{e}(B) \cdot \mathbf{e}'(B) \in K[B] \ ,\end{aligned}$$

where the final three terms equal $\mathbf{e}_\times(B)$, by re-substituting $\hat{\boldsymbol{\mu}}(B) = \hat{\mathbf{c}}(B,\mathbf{s}) - \mathbf{e}(B)$ and similarly for $\hat{\boldsymbol{\mu}}'(B)$. Recall from Section 4.1 that $\hat{\boldsymbol{\mu}}_\times(B) = \hat{\boldsymbol{\mu}}(B) \cdot \hat{\boldsymbol{\mu}}'(B) \in R[B]$ is a digit decomposition of $\mu \cdot \mu'$, hence $\mathbf{c}_\times(B,\mathbf{s}) = \mu \cdot \mu' + \mathbf{e}_\times(B) \pmod{\mathcal{L}}$. □

Lemma 4.6 essentially says that homomorphic multiplication multiplies the error polynomials in $\mathbf{c}, \mathbf{c}'$ by the factors $\hat{\mathbf{c}}'(B,\mathbf{s}), \hat{\mathbf{c}}(B,\mathbf{s}) \in K[B]$, respectively. By Equation (2.1), this enlarges the canonical (max or Euclidean) norm of each error coefficient by at most the operator norm of the corresponding factor. (The additive term $\mathbf{e}(B) \cdot \mathbf{e}'(B)$ is insignificant, since both errors are relatively much smaller.) By Lemma 4.5, these operator norms are at most $D \cdot \beta \cdot (1 + \sum_j \|s_j\|_\infty)$, where D is the degree bound in B and β bounds the canonical max norm of small representatives modulo $\mathcal{L}$. For (say) integer modulus p, using integer base b and degree bound d that satisfy $b^d \geq p$, Section 4.6 shows that small representatives modulo $\mathcal{L}^{<d}$ have canonical max norms proportional to $p^{1/d}$, so the resulting operator-norm expansion factor is at most proportional to $d \cdot p^{1/d}$. (Heuristically, the expansion factor may even be proportional to $\sqrt{d} \cdot p^{1/d}$; see Section 2.2.) So, in terms of the plaintext modulus, the error growth in dBFV is at most proportional to this quantity, versus p in ordinary BFV.

As noted above, the degree in B of $\mathbf{c}_\times$ is typically larger than those of $\mathbf{c}, \mathbf{c}'$. This is resolved by applying a *degree-reduction* map ϕ to $\mathbf{c}_\times$; see Section 4.6. Similarly, if $\mathbf{c}, \mathbf{c}'$ are affine linear in S (as is typical), then $\mathbf{c}_\times$ is quadratic in S. Just as in ordinary BFV, we resolve this using key switching (see Section 3.4.2), by linearizing the quadratic-in-S coefficient of the ciphertext using an $(\mathbf{s} \otimes \mathbf{s})$-to-$\mathbf{s}$ hint made up of dBFV ciphertexts. More specifically, the degree-reduced quadratic coefficient $\phi(\mathbf{c}_{\times,2}(B)) \in K^{<d}[B]/\mathcal{L}^{<d}$ is decomposed to a short vector over $R^{<d}[B]$, which is multiplied by the key-switching hint.

4.4.3 Automorphisms

When $b, p \in \mathbb{Z}$ are integers and $\mathfrak{p} = pR$, the dBFV system supports the homomorphic application of any automorphism of K. More generally, it supports any automorphism τ that fixes b and $\mathfrak{p}$, as follows. First observe that $\tau(\mathcal{L}) = \mathcal{L}$, and similarly for $\mathcal{L}^{<d}$: because $\tau(b) = b$, we have $\tau(\mathbf{z})(b) = \tau(\mathbf{z}(b))$ for every $\mathbf{z}(B) \in R[B]$, and because $\tau(\mathfrak{p}) = \mathfrak{p}$, it follows that $\mathbf{z}(b) \in \mathfrak{p}$ if and only if $\tau(\mathbf{z})(b) \in \mathfrak{p}$. For a concrete

finite-precision representation, we likewise choose the discretization lattice to be preserved by τ, as holds for the usual choice $\Gamma = h^{-1}\mathcal{L}^{<d}$ with integer h.

Let $\mathbf{c}(B, S) \in (K[B]/\mathcal{L})[S]$ be a dBFV ciphertext encrypting plaintext $\mu \in R/\mathfrak{p}$ with error $\mathbf{e}(B) \in K[B]$ under secret key $\mathbf{s}$ (Definition 4.3). Because $\tau(\mathcal{L}) = \mathcal{L}$, we may apply τ coefficient-wise to the decryption relation. Since τ preserves sums and products,

$$\tau(\mathbf{c})(B, \tau(\mathbf{s})) = \tau(\mathbf{c}(B, \mathbf{s})) = \tau(\mu) + \tau(\mathbf{e}(B)) \pmod{\mathcal{L}} .$$

That is, $\tau(\mathbf{c})$ is a dBFV ciphertext that encrypts $\tau(\mu) \in R/\mathfrak{p}$ with error $\tau(\mathbf{e}(B)) \in K[B]$ under secret key $\tau(\mathbf{s})$. Because τ only permutes the canonical-embedding coordinates, it preserves any symmetric decoding condition on them, so $\tau(\mathbf{e}(B))$ remains decodable coefficient-wise modulo R. As usual, we can switch the secret key from $\tau(\mathbf{s})$ back to $\mathbf{s}$ (or some other secret key) using key-switching, incurring its usual small additive error.

4.4.4 Division by Divisors of $\mathfrak{p}$

Recalling that $\mathfrak{b} = bR$, suppose that $\mathfrak{b} \mid \mathfrak{p}$. Then when the plaintext is known to be divisible by b (i.e., an element of $\mathfrak{b}$), we can homomorphically divide both it and the plaintext modulus by b. (This is an important operation in bootstrapping, as detailed in Section 5.) This is somewhat more subtle than in ordinary BFV, where we simply divide the ciphertext by b, which also has the effect of dividing the error by b (see Section 3.4.3). Here, we instead divide the ciphertext's constant-in-B term by b, and add it to the rest of the polynomial divided by B (i.e., we decrease the degrees of all the remaining terms by one). This has the same effect on the error polynomial, while dividing the plaintext (and its modulus) by b.

To formalize this, denote $\mathcal{L}_\mathfrak{p} = \mathcal{L} = \mathcal{L}^\perp_\mathfrak{p}(b)$ (see Equation (4.1)), and more generally for any ideal $\mathfrak{r} \mid \mathfrak{p}$, define the $R[B]$-ideal

$$\mathcal{L}_\mathfrak{r} = \mathcal{L}^\perp_\mathfrak{r}(b) := \{\mathbf{z}(B) \in R[B] : \mathbf{z}(b) \in \mathfrak{r}\} \supseteq \mathcal{L}_\mathfrak{p} ,$$

so that $\mathcal{L}_\mathfrak{r}/\mathcal{L}_\mathfrak{p}$ is the submodule of $R[B]/\mathcal{L}_\mathfrak{p} \cong R/\mathfrak{p}$ corresponding to $\mathfrak{r}/\mathfrak{p}$. Then for any $\mathbf{z}(B) = z_0 + \tilde{\mathbf{z}}(B) \cdot B \in \mathcal{L}_\mathfrak{b}$, we have that $z_0 \in \mathfrak{b}$ because $z_0 = z_0 + \tilde{\mathbf{z}}(b) \cdot b = \mathbf{z}(b) = 0 \pmod{\mathfrak{b}}$.

Next, define the K-linear function $\phi_b \colon K[B] \to K[B]$ by

$$\phi_b(c_0 + \tilde{\mathbf{c}}(B) \cdot B) = c_0/b + \tilde{\mathbf{c}}(B) ,$$

and observe that $\phi_b(\mathbf{c})(b) = \mathbf{c}(b)/b$ for all $\mathbf{c}(B) \in K[B]$. So, by the above, $\phi_b(\mathcal{L}_\mathfrak{r}) \subseteq \mathcal{L}_{\mathfrak{b}^{-1}\mathfrak{r}}$ for any $\mathfrak{b} \mid \mathfrak{r} \mid \mathfrak{p}$. Therefore, ϕ_b induces R-module homomorphisms from $K[B]/\mathcal{L}_\mathfrak{p}$ to $K[B]/\mathcal{L}_{\mathfrak{b}^{-1}\mathfrak{p}}$, and from $\mathcal{L}_\mathfrak{b}/\mathcal{L}_\mathfrak{p}$ to $R[B]/\mathcal{L}_{\mathfrak{b}^{-1}\mathfrak{p}}$. Analogous statements hold for degree-bounded domains and ranges, where ϕ_b reduces the degree bound by 1. From all of this we immediately get the following.

Lemma 4.7. *Suppose that dBFV ciphertext $\mathbf{c}(B, S) \in (K[B]/\mathcal{L}_\mathfrak{p})[S]$ encrypts plaintext $\mu \in \mathfrak{b}/\mathfrak{p}$ with error polynomial $\mathbf{e}(B) \in K[B]$ under secret key $\mathbf{s}$. Then $\phi_b(\mathbf{c}) \in (K[B]/\mathcal{L}_{\mathfrak{b}^{-1}\mathfrak{p}})[S]$ encrypts $\mu/b \in R/(\mathfrak{b}^{-1}\mathfrak{p})$ with error polynomial $\phi_b(\mathbf{e}(B)) \in K[B]$ under $\mathbf{s}$.*

More generally (e.g., in bootstrapping), the plaintext may be known only to lie in some principal ideal $\mathfrak{a} = aR$ that divides $\mathfrak{b}$ (and hence divides $\mathfrak{p}$ as well). In this case, we can divide the plaintext and modulus by a, simply by homomorphically multiplying by the public value $b/a \in R$, then dividing by b as above. Notice that this first step expands the error polynomial by a factor of b/a; we do not see a direct way to avoid such an expansion.

However, when $\mathfrak{b}$ is equal to (or just divisible by) a larger power of $\mathfrak{a}$, with more care it is possible to "lazily defer" divisions by a, incurring less error growth overall, by tracking the divisibility of plaintexts and their moduli. The basic idea is that a product of plaintexts in the R-module $\mathfrak{a}/\mathfrak{p} \cong R/(\mathfrak{a}^{-1}\mathfrak{p})$ is in $\mathfrak{a}^2/(\mathfrak{a}\mathfrak{p}) \cong R/(\mathfrak{a}^{-1}\mathfrak{p})$, and similarly for larger (and potentially different) powers of $\mathfrak{a}$. By keeping plaintexts in the left-hand-side quotients (which have "larger" ideals in the "denominator"), and dividing later by larger powers of a (or by b, when possible), we can avoid incurring the (b/a)-factor error growth of each individual division, and also get much smaller error growth when we do perform a division.

4.5 Homomorphic Change of Base

We can convert a dBFV ciphertext from base b to some other base $b' \in R$, i.e., from being modulo the lattice $\mathcal{L} = \mathcal{L}_{\mathfrak{p}}^{\perp}(b)$ to $\mathcal{L}' = \mathcal{L}_{\mathfrak{p}}^{\perp}(b')$ (see Equation (4.1)). This can be useful to get a different tradeoff between error growth and ciphertext sizes (and running times), or as preprocessing for homomorphic comparison (see Section 1.3.5).

As a simple warm-up, if $b = b'^c$ for a positive integer c, the conversion is analogous to the BFV-to-dBFV transform (Item 2 of Remark 4.4): we substitute $B = B'^c$ and reduce modulo $\mathcal{L}'$ to get the dBFV ciphertext $\mathbf{c}'(B', S) := \mathbf{c}(B'^c, S) \bmod \mathcal{L}'$. Because $b = b'^c$, this preserves the plaintext, and the error polynomial is simply $\mathbf{e}(B'^c)$, which has zero coefficients for the degrees not divisible by c.

The general transform can be formalized as follows; afterward we give some concrete examples.

Lemma 4.8. *Let $b' \in R$ and $\mathcal{L}' = \mathcal{L}_{\mathfrak{p}}^{\perp}(b')$, and let $\phi\colon K[B] \to K[B']$ be a K-linear function such that $\phi(\mathcal{L}) \subseteq \mathcal{L}'$ and $\phi(r) = r \pmod{\mathcal{L}'}$ for every $r \in R$. Then for any dBFV ciphertext $\mathbf{c}(B, S) \in (K[B]/\mathcal{L})[S]$ that encrypts plaintext $\mu \in R/\mathfrak{p}$ with error $\mathbf{e}(B) \in K[B]$ under secret key $\mathbf{s}$, the transformed ciphertext $\phi(\mathbf{c}) \in (K[B']/\mathcal{L}')[S]$ encrypts μ with error $\phi(\mathbf{e}(B))$ under $\mathbf{s}$.*

Proof. By the two hypotheses on ϕ, K-linearity and the decryption relation (Equation (4.2)) give

$$\phi(\mathbf{c}(B, S))(\mathbf{s}) = \phi(\mathbf{c}(B, \mathbf{s})) = \mu + \phi(\mathbf{e}(B)) \pmod{\mathcal{L}'} . \qquad \square$$

1. The above warm-up example of $b = b'^c$ is formalized by defining ϕ via the variable substitution $B = B'^c$. This fixes constants, and $\phi(\mathbf{z}(B))(b') = \mathbf{z}(b)$ for every $\mathbf{z}(B) \in R[B]$, so ϕ maps $\mathcal{L}$ into $\mathcal{L}'$.

2. In the reverse direction, if $b' = b^c$ for a positive integer c, we define ϕ to collapse each block of c coefficients using the powers of b, i.e., substitute $B^c = B'$ and evaluate the remaining powers $B^0, \ldots, B^{c-1}$ at $B = b$ to get a polynomial in $K[B']$. Again this map fixes constants, and $\phi(\mathbf{z}(B))(b') = \mathbf{z}(b)$ for every $\mathbf{z}(B) \in R[B]$, so ϕ maps $\mathcal{L}$ into $\mathcal{L}'$. The largest coefficient of the error polynomial is enlarged by a factor no larger than $b^c/(b-1)$.

3. For general b and b', as in general degree reduction (Section 4.6.2), for each relevant i choose a small polynomial $\mathbf{r}_i(B') \in R[B']$ such that $\mathbf{r}_i(b') = b^i \pmod{\mathfrak{p}}$, taking $\mathbf{r}_0(B') = 1$. Define the K-linear map ϕ by $\phi(B^i) = \mathbf{r}_i(B')$. Because $\phi(1) = \mathbf{r}_0(B') = 1$, K-linearity implies that ϕ fixes constants. Moreover, $\phi(\mathbf{z}(B))(b') = \mathbf{z}(b) \pmod{\mathfrak{p}}$ for every $\mathbf{z}(B) \in R[B]$, so $\phi(\mathcal{L}) \subseteq \mathcal{L}'$. In the matrix representation of Equation (4.4), the ith column of $\mathbf{P}$ is the coefficient vector of $\mathbf{r}_i(B')$, so the transformed error is $\mathbf{P}\mathbf{e}$, with expansion governed by $\|\mathbf{P}\|_{\mathrm{op}}$.

4.6 Lifting and Degree Reduction

Recall from Definition 4.3 and Equation (4.1) that dBFV ciphertexts are elements of $(K[B]/\mathcal{L})[S]$ of some degree in B, that homomorphic multiplication (Section 4.4) needs to lift their coefficients from $K[B]/\mathcal{L}$ to short representatives in $K[B]$, and that the intermediate product ciphertext typically has larger degree in B. So, we need ways to efficiently lift to short representatives, and to reduce the degree in B below some fixed bound. Both can be done using a short R-basis of a suitable degree-bounded restriction of $\mathcal{L}$ (or a full-rank sublattice thereof).

For lifting to short representatives, we use the basis with standard lattice algorithms (Section 2.3), or a hybrid of them.[20] (Throughout this subsection, asymptotic bounds on the sizes of lifted representatives may hide a factor that depends only on the fixed short $\mathbb{Z}$-basis of R used for lifting.) To reduce the degree in B, we apply a suitable projection that maps $K[B]$ and $\mathcal{L}$ to bounded-degree restrictions thereof; depending on the parameters, this may or may not enlarge the error somewhat. The following formally captures what we want and get from a degree-reduction map.

Lemma 4.9. *Let $\phi\colon K[B] \to K^{<d}[B]$ be a K-linear degree-reduction map satisfying $\phi(\mathcal{L}) \subseteq \mathcal{L}^{<d}$ and $\phi(\mu) = \mu \pmod{\mathcal{L}^{<d}}$ for all $\mu \in R$. Then when ϕ is applied (coefficient-wise) to any dBFV ciphertext in $(K[B]/\mathcal{L})[S]$ with error polynomial $\mathbf{e}(B) \in K[B]$, it yields one in $(K^{<d}[B]/\mathcal{L}^{<d})[S]$ that encrypts the same plaintext (under the same secret key) with error polynomial $\phi(\mathbf{e}(B)) \in K^{<d}[B]$.*

Proof. By the hypotheses, ϕ induces an R-module homomorphism from $K[B]/\mathcal{L}$ to $K^{<d}[B]/\mathcal{L}^{<d}$. Hence, when ϕ is applied to a dBFV ciphertext $\mathbf{c}(B,S) \in (K[B]/\mathcal{L})[S]$ that encrypts plaintext $\mu \in R/\mathfrak{p}$ under secret key $\mathbf{s} \in R^k$ with error $\mathbf{e}(B) \in K[B]$, by Equation (4.2),

$$\phi(\mathbf{c}(B,S))(\mathbf{s}) = \phi(\mathbf{c}(B,\mathbf{s})) = \phi(\mu + \mathbf{e}(B) + \mathcal{L}) = \mu + \phi(\mathbf{e}(B)) \pmod{\mathcal{L}^{<d}}\,. \qquad \square$$

In the rest of this subsection we construct and analyze degree-reduction projections and bases of the (suitably degree-restricted) lattice $\mathcal{L}$. When $\mathfrak{p}$ is a power of $\mathfrak{b}$, everything works out especially simply and nicely: lifting yields particularly short outputs, and degree reduction does not enlarge the error (indeed, it can even reduce it). We handle this special case first in Section 4.6.1, then the general case of arbitrary $\mathfrak{p}$ and b in Section 4.6.2.

4.6.1 Simple Lifting and Degree Reduction for $\mathfrak{p} = \mathfrak{b}^d$

For integer $d \geq 1$, the case $\mathfrak{p} = \mathfrak{b}^d = b^d R$ (e.g., integer plaintext modulus $p = b^d$) yields particularly simple degree-reduction and lifting algorithms: the former has no error growth, and the latter has especially short outputs whose coefficients are distinguished representatives of $K/\mathfrak{b}$.

Degree reduction. We define the K-linear degree-reduction map $\phi\colon K[B] \to K^{<d}[B]$ to simply discard any monomial of degree $\geq d$, i.e., reduction modulo B^d in $K[B]$. Then $\phi(\mathcal{L}) = \mathcal{L}^{<d}$ because $b^j = 0 \pmod{\mathfrak{p}}$ for every $j \geq d$, and trivially $\phi(\mu) = \mu$ for all $\mu \in R$. So, ϕ satisfies the hypotheses of Lemma 4.9, and moreover, it is non-expanding (i.e., it does not enlarge the error when applied to a ciphertext).

[20] As already noted, $\mathcal{L} = \mathcal{L}^{\perp}_{\mathfrak{p}}(b)$ is essentially the primitive powers-of-b gadget lattice from [MP12]. The associated short bases and lifting algorithms we describe here are also essentially the same as those considered in [MP12], though our degree-reduction and quasi-linear-time hybrid lifting algorithms are novel.

Lifting for the lattice $\mathcal{L}^{<d}$. Next, an R-basis for the lattice $\mathcal{L}^{<d}$ consists of the polynomials

$$(b - B) \cdot B^i \quad , \quad b \cdot B^{d-1}$$

for $i = 0, \ldots, d-2$. (Apply Lemma 4.10 below with $\mathbf{r}(B) = 0$.) This basis corresponds to a lower-triangular matrix, where each polynomial corresponds to its column vector of coefficients (starting with the constant term). So, the Gram-Schmidt orthogonalization (from right to left) corresponds to the scaled identity matrix $b \cdot \mathbf{I}_d$. This yields an especially simple nearest-plane lifting algorithm for the lattice $\mathcal{L}^{<d}$.

Given an input $\mathbf{a}(B) = \sum_{i<d} a_i \cdot B^i \in K^{<d}[B]/\mathcal{L}^{<d}$, do the following for $i = 0$ to $d-1$:

- express $a_i = q_i \cdot b + a'_i$, where a'_i is the distinguished representative of $a_i \bmod \mathfrak{b}$, and $q_i \in R$;
- let $a_{i+1} = a_{i+1} + q_i$. (For $i = d-1$, skip this step.)

Finally, output $\mathbf{a}'(B) = \sum_{i<d} a'_i \cdot B^i \in K^{<d}[B]$.

By inspection, the output's coefficients are distinguished representatives modulo $\mathfrak{b}$. Iteration $i < d-1$ of the loop has the effect of subtracting by $q_i \cdot (b - B) \cdot B^i \in \mathcal{L}^{<d}$, and by $q_i \cdot b \cdot B^{d-1} \in \mathcal{L}^{<d}$ for $i = d-1$, so the final output is congruent modulo $\mathcal{L}^{<d}$ to the input, as needed.

4.6.2 General Lifting and Degree Reduction for any $\mathfrak{p}, b$

For arbitrary $\mathfrak{p}$ and b, we construct degree-reduction projections and a short basis of the degree-bounded lattice as follows.

Degree reduction. Let d be a sufficiently large integer so that any $B^j \in R[B]/\mathcal{L}$ has an efficiently computable small enough representative $\mathbf{r}_j(B) \in R^{<d}[B]$, i.e., $\mathbf{r}_j(B) = B^j \pmod{\mathcal{L}}$. As we will see below, the particular choices of $\mathbf{r}_j(B)$ can be derived in various ways, trading error expansion against algorithmic efficiency.

Any K-linear $\phi\colon K[B] \to K^{<d}[B]$ is uniquely defined by its outputs $\phi(B^j) \in K^{<d}[B]$ for $j \geq 0$. Defining $\phi(B^j) = \mathbf{r}_j(B) \in R^{<d}[B]$, we get that $\phi(\mathcal{L}) \subseteq \mathcal{L}^{<d}$ and $\phi(\mu) = \mu \cdot \mathbf{r}_0(B) = \mu \pmod{\mathcal{L}^{<d}}$ for any $\mu \in R$. So, ϕ satisfies the hypotheses of Lemma 4.9.

We can view this map in matrix form as follows: identify any element of $K[B]$ with the column vector over K of its coefficients, starting from the constant term. Then ϕ corresponds to left-multiplication by the matrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{r}_0 & \cdots & \mathbf{r}_{d-1} & \mathbf{r}_d & \cdots \end{pmatrix} , \tag{4.4}$$

where we truncate the number of columns to match the degree bound of the input polynomial (for dBFV, the bound $< 2d$ suffices), and the first d columns can form the identity matrix without loss of generality.

The choice of ϕ (equivalently, the $\mathbf{r}_j$) presents a tradeoff between the operator norm $\|\mathbf{P}\|_{\text{op}} := \max_{\mathbf{x} \neq 0} \|\mathbf{P}\mathbf{x}\|_2 / \|\mathbf{x}\|_2$ (which bounds the multiplicative error blowup) and the cost of computing $\mathbf{P}\mathbf{x}$. Two representative choices, at opposite ends of a spectrum trading off operator norm and algorithmic running time, are:

- *Independent small representatives.* Choose each $\mathbf{r}_j(B)$ separately as a small representative of $B^j \bmod \mathcal{L}$—e.g., for integer modulus $\mathfrak{p} = pR$ with $d = \lceil \log_b p \rceil$, use the polynomial whose coefficients are a base-b representation of $b^j \bmod p$, with canonical max norm $O(b)$. This yields $\|\mathbf{P}\|_{\text{op}} = O(d \cdot b)$ in the worst case. However, the $\mathbf{r}_j$ typically have no shared algebraic structure, so computing $\mathbf{P}\mathbf{x}$ requires $\Theta(d^2)$ K-operations in general.

- *Polynomial remainders.* Define ϕ to be the K-linear map $\phi(\mathbf{u}(B)) = \mathbf{u}(B) \bmod (B^d - \mathbf{f}(B))$ in $K[B]$ for some small $\mathbf{f}(B) \in R^{<d}[B]$ with $\mathbf{f}(B) = B^d \pmod{\mathcal{L}}$ (e.g., a base-b digit decomposition of $b^d \bmod \mathfrak{p}$), which admits $O(d \log d)$-time evaluation via standard fast polynomial division. This is equivalent to taking $\mathbf{r}_j(B) := B^j \bmod (B^d - \mathbf{f}(B))$; note that here the modulus is just a single element in (not all of) $\mathcal{L}$. However, the iterative substitution $B^d \to \mathbf{f}(B)$ can compound, making $\|\mathbf{P}\|_{\text{op}}$ proportional to $\|\mathbf{f}\|_{\text{op}}^d$ in the worst case. Still, this approach is viable for structured moduli such as Mersenne-style $p = b^d - c$ for small constant $c \in R$, with $\mathbf{P} = [\mathbf{I}_d \mid c\mathbf{I}_d]$ and thus $\|\mathbf{P}\|_{\text{op}} \le 1 + \|c\|_\infty$ (for input degree bound $< 2d$).

More generally, as an intermediate option we can define degree reduction in a multi-stage way, as reduction modulo a sequence of polynomials $B^k - \mathbf{f}_k(B) \in \mathcal{L}$ with $\deg(\mathbf{f}_k) < k$ (or even $\deg(\mathbf{f}_k) < d$) for decreasing values of k ending with $k = d$. Each stage can be evaluated in $O(d \log d)$ operations or less, and the total operator norm depends on the specific forms of the $\mathbf{f}_k$. Whether this can yield a useful degree-reduction map depends on b and the specific structure of $\mathfrak{p}$.

Lifting for the lattice $\mathcal{L}^{<d}$. We next address lifting for the lattice $\mathcal{L}^{<d}$ by constructing a short full-rank set of lattice vectors. This relies on a small polynomial $\mathbf{r}(B) \in R^{<d}[B]$ that is a representative of $B^d \bmod \mathcal{L}$ (i.e., $\mathbf{r}(b) = b^d \pmod{\mathfrak{p}}$) and additionally satisfies $\mathbf{r}(b) \neq b^d$ in R (which holds, e.g., for $\mathfrak{p} = pR$ with $b^d \ge p$), ensuring that the construction below yields a non-degenerate set. Note that $\mathbf{r}(B)$ can (but need not) be the polynomial $\mathbf{r}_d(B)$ used for degree reduction above, since the two serve distinct purposes and may be chosen independently.

Lemma 4.10. *Let $\mathbf{r}(B) \in R^{<d}[B]$ satisfy $\mathbf{r}(b) = b^d \pmod{\mathfrak{p}}$ with $\mathbf{r}(b) \neq b^d$ in R. Then the small polynomials $(b - B) \cdot B^i$ for $i = 0, \ldots, d-2$ and $b \cdot B^{d-1} - \mathbf{r}(B)$ are a full-rank set of vectors in the lattice $\mathcal{L}^{<d}$, and form a basis if $b^d - \mathbf{r}(b)$ generates $\mathfrak{p}$.*

Proof. Writing $\mathbf{r}(B) = \sum_{i<d} r_i \cdot B^i$, and (as usual) treating each polynomial as its column vector of coefficients, we can arrange them in a matrix

$$\mathbf{R} = \left(\begin{array}{cccc|c} b & & & & -r_0 \\ -1 & b & & & -r_1 \\ & -1 & \ddots & & \\ & & & b & -r_{d-2} \\ & & & -1 & b - r_{d-1} \end{array}\right) \in R^{d \times d} . \tag{4.5}$$

By a standard expansion over the rightmost column, the determinant of this matrix is $b^d - \mathbf{r}(b) \in \mathfrak{p} \setminus \{0\}$, so the matrix is nonsingular (over R) and hence has full rank. Moreover, if this determinant generates $\mathfrak{p}$, then the index in R^d of the lattice generated by the polynomials equals that of $\mathcal{L}^{<d}$, so the polynomials indeed form a basis of $\mathcal{L}^{<d}$. □

As described in Section 2.3, using either the "round-off" or "nearest-plane" algorithms [Bab85] with the small polynomials from Lemma 4.10, we can lift to short representatives for the lattice $\mathcal{L}^{<d}$. However, neither option is especially attractive for practice, because the inverse and Gram–Schmidt orthogonalization of the matrix $\mathbf{R}$ from Equation (4.5) require high precision, and do not have such nice forms, so straightforward implementations run in time at least quadratic in d.

Specialized hybrid lifting. Here we describe a practical hybrid of the two lifting algorithms, specialized to the matrix $\mathbf{R}$ from Equation (4.5), which uses only $O(d)$ K-operations and R-liftings, at relatively low precision. Essentially, this algorithm applies the nearest-plane strategy for *just the rightmost column* of $\mathbf{R}$, then the round-off strategy for the remaining columns.

Assume that K has an automorphism $\bar{\cdot}$ corresponding to complex conjugation, i.e., $\sigma(\bar{a}) = \overline{\sigma(a)}$ for every embedding $\sigma\colon K \to \mathbb{C}$ and $a \in K$, and equip K^d with the induced Hermitian form $\langle \mathbf{u}, \mathbf{v} \rangle := \sum_i \bar{u_i} \cdot v_i \in K$. Under every complex embedding this form is conjugate symmetric and positive definite, and it is K-linear in its second argument.

Let $\mathbf{B} \in R^{d \times (d-1)}$ be the matrix consisting of the first $d-1$ columns of $\mathbf{R}$, and let $\mathcal{V} = \mathrm{span}_K(\mathbf{B}) \subseteq K^d$. Observe that the column vector $\mathbf{b} \in R^d$ defined by its conjugate transpose $\mathbf{b}^* = (b^0, b^1, \ldots, b^{d-1})$ spans $\mathcal{V}^\perp$, i.e., $\mathbf{b}^* \cdot \mathbf{B} = \mathbf{0}$. Next, let $\mathbf{D} \in K^{d \times (d-1)}$ be dual to $\mathbf{B}$, i.e., $\mathbf{D}^* \cdot \mathbf{b} = \mathbf{0}$ and $\mathbf{D}^* \cdot \mathbf{B} = \mathbf{I}_{d-1}$. Define the prefix- and suffix-evaluation matrices $\mathbf{L}_b, \mathbf{U}_b \in R^{(d-1) \times d}$ by

$$\mathbf{L}_b := \begin{pmatrix} b^0 & & & & \\ b^0 & b^1 & & & \\ \vdots & \vdots & \ddots & & \\ b^0 & b^1 & \cdots & b^{d-2} & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{U}_b := \begin{pmatrix} 0 & b^1 & b^2 & \cdots & b^{d-1} \\ & & b^2 & \cdots & b^{d-1} \\ & & & \ddots & \vdots \\ & & & & b^{d-1} \end{pmatrix} .$$

Left-multiplication by $\mathbf{L}_b$ (respectively, $\mathbf{U}_b$) maps the coefficient vector of a polynomial to the evaluations at b of its successively longer prefixes (respectively, shorter suffixes). Both products can therefore be computed together in a single pass over the coefficients. Letting $s = \bar{b}b$, we claim that

$$\begin{aligned} (s^d - 1) \cdot \mathbf{D}^* = &\begin{pmatrix} \bar{b}^1(s^{d-1} - 1) & & & \\ & \bar{b}^2(s^{d-2} - 1) & & \\ & & \ddots & \\ & & & \bar{b}^{d-1}(s^1 - 1) \end{pmatrix} \mathbf{L}_b \\ &- \begin{pmatrix} b^{-1}(s^1 - 1) & & & \\ & b^{-2}(s^2 - 1) & & \\ & & \ddots & \\ & & & b^{-(d-1)}(s^{d-1} - 1) \end{pmatrix} \mathbf{U}_b . \end{aligned} \tag{4.6}$$

Indeed, multiplying the right-hand side by $\mathbf{B}$ and $\mathbf{b}$ gives $(s^d - 1)\mathbf{I}_{d-1}$ and $\mathbf{0}$, respectively, which verifies the defining conditions on $\mathbf{D}^*$ and hence the identity.

Given an input coset $\bar{\mathbf{v}} \in K^d / \mathcal{L}^{<d}$, first lift it arbitrarily to a representative $\mathbf{v} \in K^d$, viewed as a polynomial $\mathbf{v}(B) \in K^{<d}[B]$. Using the structured full-rank set represented by $\mathbf{R}$, the hybrid algorithm then shortens this representative as follows:

1. *Nearest-plane step:* letting $\mathbf{r}$ be the rightmost column of $\mathbf{R}$, compute

$$c := \frac{\langle \mathbf{b}, \mathbf{v} \rangle}{\langle \mathbf{b}, \mathbf{r} \rangle} = \frac{\mathbf{b}^* \cdot \mathbf{v}}{\mathbf{b}^* \cdot \mathbf{r}} = \frac{\mathbf{v}(b)}{b^d - \mathbf{r}(b)} \in K ,$$

round it to $z = \lfloor c \rceil \in R$ (using a known short $\mathbb{Z}$-basis of R), and let $\mathbf{v}' = \mathbf{v} - z \cdot \mathbf{r}$.

To efficiently compute z, let $g = b^d - \mathbf{r}(b) \in R$, compute $v = \mathbf{v}(b) \in K$, let $w \in K$ be its distinguished representative modulo gR (using g times the short $\mathbb{Z}$-basis of R), and let $z = (v - w)/g \in R$. In particular, if $b, p \in \mathbb{Z}$ then we can set d such that $g = b^d - \mathbf{r}(b) = p$, so this step simply computes the quotient and remainder of $\mathbf{v}(b)$ divided by p.

2. *Round-off step:* compute $\mathbf{z} = \lfloor \mathbf{D}^* \cdot \mathbf{v}' \rceil \in R^{d-1}$ and output $\mathbf{v}'' = \mathbf{v}' - \mathbf{B}\mathbf{z} \in K^d$.

   To efficiently compute $\mathbf{z} = (z_1, \ldots, z_{d-1})^t$ from $\mathbf{v}' = (v'_0, \ldots, v'_{d-1})^t$ following Equation (4.6), use the following loop, initialized with $v_{\text{pre}} = 0$ and $v_{\text{suf}} = \mathbf{v}'(b) \in K$. For $i = 1, \ldots, d-1$:

   (a) Let $v_{\text{pre}} = v_{\text{pre}} + v'_{i-1} \cdot b^{i-1}$ and $v_{\text{suf}} = v_{\text{suf}} - v'_{i-1} \cdot b^{i-1}$. (These are the current prefix and suffix evaluations of $\mathbf{v}'$ at b.)

   (b) Let $y_i = \bar{b}^i(s^{d-i} - 1) \cdot v_{\text{pre}} - b^{-i}(s^i - 1) \cdot v_{\text{suf}} \in K$, let $w_i \in K$ be its distinguished representative modulo $(s^d - 1)R$, and let $z_i = (y_i - w_i)/(s^d - 1) \in R$.[21]

Maintaining the required powers of b and s incrementally, the nearest-plane step uses $O(d)$ K-operations and one lifting for gR, while each iteration of the round-off step uses $O(1)$ K-operations and one lifting for $(s^d - 1)R$; hence, the algorithm uses $O(d)$ K-operations and $O(d)$ such liftings overall. Because $\mathbf{B}$ is bidiagonal, the product $\mathbf{B}\mathbf{z}$ can be accumulated during the loop without a separate matrix-vector multiplication or storage of $\mathbf{z}$.

Lemma 4.11. *The above hybrid algorithm outputs a representative of the input coset $\bar{\mathbf{v}}$. In particular, if $b, p \in \mathbb{N}$ and $b^d - \mathbf{r}(b) = p$ with $\mathbf{r}(b) \geq 0$, then every entry of the output has canonical max norm $O(b)$.*

Proof. First, the output $\mathbf{v}''$ represents $\bar{\mathbf{v}}$, because $\mathbf{v}'' - \mathbf{v}$ is an R-linear combination of the columns of $\mathbf{R} = (\mathbf{B} \mid \mathbf{r})$, all of which are in $\mathcal{L}^{<d}$. For the second claim, let $\tilde{\mathbf{r}}$ be the orthogonal projection of $\mathbf{r}$ onto $\mathcal{V}^\perp = \text{span}_K(\mathbf{b})$. The coordinates of $\mathbf{v}''$ with respect to the basis $(\mathbf{B} \mid \tilde{\mathbf{r}})$ are

$$\mathbf{D}^* \cdot \mathbf{v}'' = \mathbf{D}^* \cdot \mathbf{v}' - \mathbf{z} \quad \text{and} \quad \frac{\langle \tilde{\mathbf{r}}, \mathbf{v}'' \rangle}{\langle \tilde{\mathbf{r}}, \tilde{\mathbf{r}} \rangle} = \frac{\langle \mathbf{b}, \mathbf{v}'' \rangle}{\langle \mathbf{b}, \mathbf{r} \rangle} = c - z \ .$$

By the two rounding steps, these coordinates have canonical max norm $O(1)$. Moreover,

$$\|\tilde{\mathbf{r}}\|^2 = \frac{p^2 \cdot (b^2 - 1)}{b^{2d} - 1} \leq b^2 \ ,$$

where the inequality follows from $p = b^d - \mathbf{r}(b) \leq b^d$ and $d \geq 1$. Therefore, $\|\tilde{\mathbf{r}}\|_\infty \leq b$. By inspection of $(\mathbf{B} \mid \tilde{\mathbf{r}})$ (cf. Equation (4.5)), each row has at most three nonzero entries, each of canonical max norm at most b. It follows that every entry of $\mathbf{v}''$ has canonical max norm $O(b)$. □

5 Bootstrapping

In this section we consider bootstrapping for the dBFV scheme. In short, we show that bootstrapping for dBFV generically reduces to bootstrapping for BFV, which is a well-studied problem with various good approaches. Moreover, both main classes of solutions can be implemented by, and benefit from, using decomposed schemes for the bootstrapping itself, which brings the benefits of decomposition full circle.

Recall that bootstrapping [Gen09b, Gen09a] "refreshes" a high-error ciphertext into a lower-error one that encrypts the same plaintext, so that more homomorphic operations can be performed on the encrypted data. To accomplish this, bootstrapping *homomorphically evaluates the decryption function* for the given (high-error) ciphertext on a (low-error) *encryption of the secret key*, yielding a (fairly low-error) encryption of the same plaintext.

[21] Division by $s^d - 1$ can be computed or closely approximated using division by s^d. In the case of approximation, the final output of the algorithm is still a mod-$\mathcal{L}^{<d}$ representative of the input, but may be slightly larger.

We stress that the "target" encryption scheme (for the input ciphertext to be bootstrapped) can be different from the "host" homomorphic scheme (for the encrypted secret key and the homomorphic evaluation). As long as the resulting host-scheme ciphertext can be converted back to the (homomorphic) target scheme, the compute-then-bootstrap cycle can be repeated indefinitely. For bootstrapping dBFV, we consider two main approaches: one where the host scheme is BFV-style, and another where it is GSW-style [GSW13], like FHEW or TFHE [DM15, CGGI16] (which has a very different performance profile).

5.1 Generic Reduction to BFV Bootstrapping

Bootstrapping for dBFV easily reduces to bootstrapping for BFV: in essence, we simply bootstrap each of the "digit ciphertexts" in parallel, and linearly combine the results. More precisely: by Definition 4.3, let the input be a dBFV ciphertext $\mathbf{c}(B,S) \in (K^{<d}[B]/\mathcal{L}^{<d})[S]$ that encrypts plaintext $\mu \in R/\mathfrak{p}$ (under some secret key $\mathbf{s}$).

1. Following Item 3 of Remark 4.4, lift the ciphertext to some $\sum_{i<d} \mathbf{c}_i(S) \cdot B^i \in (K/\mathfrak{p})[B,S]$ where each $\mathbf{c}_i(S) \in (K/\mathfrak{p})[S]$ is a BFV ciphertext that encrypts (under the same secret key $\mathbf{s}$) the ith digit $\mu_i \in R/\mathfrak{p}$ of a (mod-$\mathfrak{p}$) digit decomposition $\boldsymbol{\mu}(B) = \sum_{i<d} \mu_i \cdot B^i \in (R/\mathfrak{p})[B]$ of μ.

2. Bootstrap (using a bootstrapping key that suitably encrypts $\mathbf{s}$) each BFV ciphertext $\mathbf{c}_i(S)$ to get a corresponding BFV ciphertext $\mathbf{c}'_i(S)$, which encrypts μ_i with small error e'_i.

3. Output the dBFV ciphertext $\mathbf{c}'(B,S) = \sum_{i<d} \mathbf{c}'_i(S) \cdot B^i \bmod \mathcal{L}$, which by definition encrypts μ with small error polynomial $\mathbf{e}'(B) = \sum_{i<d} e'_i \cdot B^i$, as desired.

We emphasize that in the above reduction, it is important that the digits μ_i are viewed as plaintexts modulo $\mathfrak{p}$ (and not modulo some "finer" ideal), because in the final step their linear combination needs to yield the original mod-$\mathfrak{p}$ plaintext.[22] So, the BFV bootstrapping needs to work for mod-$\mathfrak{p}$ plaintexts. When $\mathfrak{p}$ is very "sparse," using BFV itself as the host scheme would incur very large error growth during bootstrapping, leading to poor parameters.

A much better option is to bootstrap BFV using a host scheme that is less sensitive to the plaintext precision, like dBFV or a GSW-style scheme like FHEW/TFHE (or a decomposed variant thereof).

- Using dBFV as the host scheme, the results of the BFV bootstrapping are dBFV ciphertexts $\mathbf{c}'_i(B,S) \in (K[B]/\mathcal{L})[S]$ that respectively encrypt the digits $\mu_i \in R/\mathfrak{p}$. We then just linearly combine these to get the dBFV ciphertext $\mathbf{c}'(B,S) = \sum_i \mathbf{c}'_i(B,S) \cdot B^i \bmod \mathcal{L}$. (Recall that B^i is a small digit decomposition of $b^i \in R/\mathfrak{p}$, so this $\mathbf{c}'$ does indeed encrypt the original plaintext $\mu = \boldsymbol{\mu}(b) \bmod \mathfrak{p}$ under small error, as required.) See Section 5.2 for details.

- Alternatively, following an approach of [KDE+24], we can use a GSW-style scheme like FHEW/TFHE as the host. The error *growth* of this approach is nearly independent of the plaintext modulus $\mathfrak{p}$. However, the *error rate* of the bootstrapping key must be inversely proportional to the "sparsity" of $\mathfrak{p}$, which leads to suboptimal parameters and efficiency. Instead, just as dBFV encryption can use a much larger error rate than BFV (see Section 4.3), we can get the same improvement here by using a natural *decomposed variant* of GSW (dGSW) for the bootstrapping key and bootstrapping process. See Section 5.3 for details of the bootstrapping, and Appendix B for a description of the dGSW scheme.

[22] As an optimization, if $b^i R \mid \mathfrak{p}$, then we can treat μ_i as a plaintext modulo $b^{-i}\mathfrak{p}$, because it is ultimately multiplied by b^i. This use of a "finer" plaintext modulus can yield somewhat better bootstrapping efficiency and error growth for such digits, but it does not help for $i = 0$.

Remark 5.1. We note that in the above generic reduction, the initial lifting step, which produces several BFV encryptions of the (mod-$\mathfrak{p}$) digits of a decomposition of μ, might be somewhat wasteful, because each digit of the decomposition belongs to a small fraction of the plaintext space $R/\mathfrak{p}$. Recall that the main step of BFV bootstrapping homomorphically evaluates, for the employed discretization lattice Γ, the "rounding" function from $\Gamma/\mathfrak{p}$ to $R/\mathfrak{p}$ via some carefully designed efficient arithmetic circuit. Analogously, for an employed dBFV discretization lattice Γ', the rounding function from $\Gamma'/\mathcal{L}^{<d}$ to $R^{<d}[B]/\mathcal{L}^{<d}$ might also have a specialized efficient arithmetic circuit, which could yield a direct and more efficient bootstrapping procedure for dBFV. We leave an investigation of this idea to future work.

5.2 Bootstrapping via dBFV

Here we recall some more details of how BFV can be bootstrapped using a BFV-style scheme as the host, and dBFV in particular. Following Definition 3.1, let $\mathbf{c} = (c_0, \mathbf{c}_1) \in (K/\mathfrak{p})^{1+k}$ be a BFV ciphertext that satisfies the decryption relation

$$c_0 + \langle \mathbf{c}_1, \mathbf{s} \rangle = \mu + e \pmod{\mathfrak{p}},$$

where $\mu \in R/\mathfrak{p}$ is the plaintext and the error $e \in K$ is decodable mod R. Also suppose (without loss of generality) that $\mathbf{c}$ has been discretized to have fairly low precision in its fractional part, i.e., it is over $h^{-1}R$ for some moderately small integer $h \in \mathbb{Z}$.

1. First, "scale up" the ciphertext to $\mathbf{c}' = h \cdot \mathbf{c}$ over $R/(h\mathfrak{p})$, and view it as an "errorless" encryption of the plaintext $\mu' = h\mu + e' \in R/(h\mathfrak{p})$, where $e' = he \in R$ is decodable mod hR. That is, the scaled-up ciphertext $(c'_0, \mathbf{c}'_1)$ satisfies the zero-error decryption relation

$$c'_0 + \langle \mathbf{c}'_1, \mathbf{s} \rangle = \mu' \pmod{h\mathfrak{p}}.$$

2. Homomorphically decode (or "round") μ' to $h\mu \in hR/(h\mathfrak{p})$, while in the process dividing both the value and the modulus by h, to get an encryption of $\mu \in R/\mathfrak{p}$.

The actual work occurs in the second step, which homomorphically evaluates the nonlinear function that "rounds down" $\mu' = h\mu + e' \in R/(h\mathfrak{p})$ to $\mu \in R/\mathfrak{p}$. Such functions have been intensely studied in the literature, and their homomorphic evaluations using BFV-style schemes have been heavily optimized for a variety of forms of the ideals $\mathfrak{p}, h\mathfrak{p}$ (with or without additional constraints on μ); see, e.g., [GHS12a, AP13, HS15, CH18, OPP23, GV23, GIKV23, MHWW24, KSS24, GV25a]. These works express the rounding function algebraically, via a combination of automorphisms, polynomials modulo ideals ranging from $h\mathfrak{p}$ to $\mathfrak{p}$, and scalar division to "scale down" the plaintext and its modulus ideal. Since dBFV supports all these homomorphic operations, it can immediately be used as the host scheme for any of these approaches. This inherits all the efficiency and security advantages of dBFV over BFV, for the homomorphic computation of these particular functions.

5.3 Bootstrapping via (d)GSW

Here we describe some further details of how BFV can be bootstrapped using a GSW-style scheme like FHEW/TFHE [DM15, CGGI16]—or a novel *decomposed* GSW (dGSW) scheme we described in Appendix B—as the host. The main idea is due to [KDE+24]; we simplify the presentation using our "scale factor-free" form of BFV, extend it to dGSW, and also provide the details for efficiently handling plaintexts from subrings. (In [KDE+24], such details were given only for CKKS.)

The basic idea is as follows. Let $(c_0, \mathbf{c}_1)$ be the BFV ciphertext to be bootstrapped, and recall from Equation (3.1) that it satisfies $c_0 + \langle \mathbf{c}_1, \mathbf{s} \rangle = \mu + e \pmod{\mathfrak{p}}$, where $\mu \in R/\mathfrak{p}$ is the plaintext and $e \in K$ is decodable mod R (but is large enough to need bootstrapping). View the right-hand side as the "noisy plaintext," and suppose that $\mu = \mu_0 \in \mathbb{Z}/p\mathbb{Z}$ is just an *integer* mod $p\mathbb{Z} = \mathfrak{p} \cap \mathbb{Z}$, i.e., the ciphertext is "unpacked."[23] Then for the corresponding noisy coefficient $\mu_0 + e_0 \in \mathbb{Q}/p\mathbb{Z}$, we use FHEW/TFHE-style "programmable bootstrapping" to homomorphically compute a BFV ciphertext whose noisy plaintext is $\approx 0 + e_0 \in \mathbb{Q}/p\mathbb{Z}$, where $\approx$ hides some tiny error much smaller than e_0. We then subtract this ciphertext from the input ciphertext to obtain one with a low-noise plaintext coefficient $\approx \mu_0$, as desired. However, the *other* plaintext coefficients still have fairly large noise, which makes them unsuitable for further homomorphic computation. To address this, we homomorphically evaluate the (scaled) *trace* function, which preserves the low-noise coefficient $\approx \mu_0$ and zeroes out all the others (up to some tiny error coming from the key-switching hints). The result is a ciphertext whose entire noisy plaintext is $\approx \mu_0 = \mu$, as desired.

As mentioned above in the overview (Section 5.1), applying this approach with an ordinary GSW-style scheme like FHEW/TFHE requires a bootstrapping key with error rate $\ll 1/p$, which for large p significantly harms efficiency. Instead, we can use a *decomposed* variant of GSW (dGSW) analogous to dBFV for the bootstrapping key and homomorphic operations on it, which allows for a much larger error rate proportional to $1/p^{1/d}$ for any desired $d \geq 1$. The details of dGSW are given in Appendix B, and the following procedure adapts straightforwardly to use it, by just replacing all the mod-$\mathfrak{p}$ relations with mod-$\mathcal{L}^{<d}$ ones.

Bootstrapping procedure. For concreteness and simplicity, adopt the following setup:

- Suppose that the input BFV ciphertext is over a cyclotomic field $\mathbb{Q}(\zeta_m)$ for power-of-two $m = 2n$, and that the host FHEW/TFHE scheme is over the same field.[24] We can therefore view the field as $K := \mathbb{Q}[X]/(X^n + 1)$ and its ring of integers as $R := \mathbb{Z}[X]/(X^n + 1)$. Let the plaintext modulus be $\mathfrak{p} = pR$ for an integer $p \in \mathbb{Z}$.

- As above in Section 5.2, let $\mathbf{c} = (c_0, \mathbf{c}_1) \in (m^{-1}R/\mathfrak{p})^{1+k}$ be a BFV ciphertext that has been discretized to be over $m^{-1}R$, where each coefficient of the error $e = e(X) \in K$ is in $[-1/4, 1/4)$.

- Suppose that the plaintext $\mu(X) \in R/\mathfrak{p}$ is just an *integer* modulo p, i.e., $\mu(X) = \mu_0 \cdot X^0 = \mu_0 \in \mathbb{Z}/p\mathbb{Z}$ is merely a constant coefficient. So, the ciphertext satisfies

$$c_0 + \langle \mathbf{c}_1, \mathbf{s} \rangle = \mu_0 + e(X) \pmod{\mathfrak{p}} . \tag{5.1}$$

  For larger plaintext subrings of $R/\mathfrak{p}$, we just adapt and repeat the first three steps below for each coefficient of $\mu(X)$ that carries the plaintext. In addition, we apply the homomorphic trace on $\mathbf{c}$ and each $\tilde{\mathbf{c}}$ separately before subtracting them, so that the various polynomials $\tilde{e}(X)$ don't interfere with plaintext coefficients we want to preserve.

To bootstrap $\mathbf{c}$, we perform the following steps:

1. From the coefficients of the scaled ciphertext $m \cdot \mathbf{c} \bmod mR$, which is over R/mR, extract a vector $(c_{0,0}, \vec{c}) \in \mathbb{Z}_m^{1+kn}$ for which

$$c_{0,0} + \langle \vec{c}, \vec{s} \rangle = m \cdot e_0 \pmod{m} , \tag{5.2}$$

[23] In general, we can extract an unpacked ciphertext for each $\mathbb{Z}$-coefficient of μ that may carry meaningful data, and apply this procedure to each of them in parallel.

[24] These conditions can be relaxed using more general cyclotomics and ring switching, respectively.

where $\vec{s} \in \mathbb{Z}^{kn}$ is the coefficient vector of the R-entries of $\mathbf{s}$, and $m \cdot e_0 \in \mathbb{Z} \cap [-m/4, m/4) = \mathbb{Z} \cap [-n/2, n/2)$. The above equation holds because $m \cdot \mu_0 = 0 \pmod{m}$, and the extraction can be done efficiently because the constant coefficient $\mu_0 + e_0$ of $c_0 + \langle \mathbf{c}_1, \mathbf{s} \rangle$ is affine linear in $\vec{s}$.

2. Using $(c_{0,0}, \vec{c})$, homomorphically compute (using the bootstrapping key) a BFV ciphertext $\tilde{\mathbf{c}} = (\tilde{c}_0, \tilde{\mathbf{c}}_1)$ satisfying

$$\tilde{c}_0 + \langle \tilde{\mathbf{c}}_1, \mathbf{s} \rangle \approx \tilde{e}(X) \pmod{\mathfrak{p}}, \tag{5.3}$$

where $\tilde{e}(X) \in K$ has a constant coefficient of e_0, and $\approx$ hides a tiny error.[25] This can be done by FHEW/TFHE-style "blind rotation" with a suitable "test polynomial" [DM15, CGGI16], using a bootstrapping key that consists of tiny-error mod-$\mathfrak{p}$ GSW encryptions of the entries of $\vec{s}$, encoded "in the exponent" as mth roots of unity.

In a little more detail, the mod-m arithmetic from Equation (5.2) is naturally computed homomorphically by the blind rotation, via the order-m roots of unity in R. Because $m \cdot e_0$ belongs to the size-n interval $\mathbb{Z} \cap [-n/2, n/2)$, using a suitable (fractional, over K) test polynomial yields a BFV encryption of a polynomial with constant term $e_0 \in \mathbb{Q}$.

3. Compute the difference $\hat{\mathbf{c}} = (\hat{c}_0, \hat{\mathbf{c}}_1) = \mathbf{c} - \tilde{\mathbf{c}}$. By Equations (5.1) and (5.3),

$$\hat{c}_0 + \langle \hat{\mathbf{c}}_1, \mathbf{s} \rangle \approx \mu_0 + \hat{z}(X) \pmod{\mathfrak{p}},$$

where $\hat{z}(X) = e(X) - \tilde{e}(X) \in K$ has a constant coefficient of zero. So, the right-hand side's constant coefficient has the same integer part μ_0 as in the original noisy plaintext, but its error is tiny.

4. Finally, homomorphically evaluate on $\hat{\mathbf{c}}$ the (scaled) trace function $n^{-1} \cdot \mathrm{Tr}$ from $K/\mathfrak{p}$ to $\mathbb{Q}/p\mathbb{Z}$ using automorphisms (and scalar division), which maps the plaintext to just $\mu_0 = n^{-1} \cdot \mathrm{Tr}(\mu_0 + \hat{z}(X))$.

In more detail, this can be done most efficiently by tracing down the tower of two-power cyclotomics one step at a time; each relative trace is the sum of just two automorphisms, one of which is the identity. At the start we lift to plaintext modulus $n\mathfrak{p}$ (see Section 3.4.3). At each step down the tower we sum the ciphertext and the homomorphic application of the appropriate non-identity automorphism (see Section 3.4.4). At the end we homomorphically divide the plaintext and modulus by n, to get the scaled trace modulo p (see Section 3.4.3). All this results in a BFV ciphertext $\mathbf{c}' = (c'_0, \mathbf{c}'_1)$ for which, as desired,

$$c'_0 + \langle \mathbf{c}'_1, \mathbf{s} \rangle \approx \mu_0 \pmod{\mathfrak{p}}.$$

References

[AP13] J. Alperin-Sheriff and C. Peikert. Practical bootstrapping in quasilinear time. In *CRYPTO*, pages 1–20. 2013. Pages 1, 9, and 35.

[AP14] J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, pages 297–314. 2014. Pages 1, 9, and 46.

[Bab85] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. Preliminary version in STACS 1985. Pages 15 and 31.

[25] To be precise, this BFV ciphertext has a "fractional" plaintext that is not necessarily in $R/\mathfrak{p}$, but this is of no consequence.

[BCG+24] M. G. Belorgey, S. Carpov, N. Gama, S. Guasch, and D. Jetchev. Revisiting key decomposition techniques for FHE: simpler, faster and more generic. In *ASIACRYPT*, pages 176–207. 2024. Page 12.

[BFG+26] Z. Brakerski, O. Friedman, D. Golan, A. Gurny, D. Mutzari, and O. Sheinfeld. REFHE: fully homomorphic ALU. In *EUROCRYPT*, pages 183–213. 2026. Pages 2, 3, 5, and 11.

[BGG+18] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, pages 565–596. 2018. Page 2.

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *TOCT*, 6(3):13, 2014. Preliminary version in ITCS 2012. Pages 1, 2, 5, 13, and 47.

[BK25] D. Boneh and J. Kim. Homomorphic encryption for large integers from nested residue number systems. In *CRYPTO*, pages 338–370. 2025. Pages 2, 3, 5, and 12.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, pages 868–886. 2012. Pages 1, 2, 4, 5, 13, and 15.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524. 2011. Page 1.

[BV11b] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, 43(2):831–871, 2014. Preliminary version in FOCS 2011. Pages 1, 7, and 19.

[BV14] Z. Brakerski and V. Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12. 2014. Pages 9 and 46.

[CCKS23] J. H. Cheon, W. Cho, J. Kim, and D. Stehlé. Homomorphic multiple precision multiplication for CKKS and reduced modulus consumption. In *ACM CCS*, pages 696–710. 2023. Pages 2, 3, 4, 5, 7, 8, 12, and 24.

[CCS19] H. Chen, I. Chillotti, and Y. Song. Improved bootstrapping for approximate homomorphic encryption. In *EUROCRYPT*, pages 34–54. 2019. Page 1.

[CGGI16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: fast fully homomorphic encryption over the torus. *J. Cryptology*, 33(1):34–91, 2020. Preliminary versions in ASIACRYPT 2016 and 2017. Pages 1, 2, 5, 7, 9, 34, 35, 37, and 46.

[CH18] H. Chen and K. Han. Homomorphic lower digits removal and improved FHE bootstrapping. In *EUROCRYPT*, pages 315–337. 2018. Pages 9 and 35.

[CKKS17] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, pages 409–437. 2017. Pages 1, 2, 5, and 7.

[CLPX18] H. Chen, K. Laine, R. Player, and Y. Xia. High-precision arithmetic in homomorphic encryption. In *CT-RSA*, pages 116–136. 2018. Pages 2, 3, 4, 5, 11, 15, and 16.

[CPL26] G. Cha, D. Park, and J. Lee. Improved radix-based approximate homomorphic encryption for large integers via lightweight bootstrapped digit carry. In *EUROCRYPT*, pages 243–273. 2026. Pages 2, 3, and 12.

[DM15] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, pages 617–640. 2015. Pages 1, 2, 9, 34, 35, 37, and 46.

[DMPS24] N. Drucker, G. Moshkowich, T. Pelleg, and H. Shaul. BLEACH: Cleaning errors in discrete computations over CKKS. *J. Cryptol.*, 37(1):3, 2024. Page 12.

[FV12] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144. Pages 2, 4, 5, 13, and 15.

[GC14] M. Geihs and D. Cabarcas. Efficient integer encoding for homomorphic encryption via ring isomorphisms. In *LATINCRYPT*, pages 48–63. 2014. Pages 2, 3, 4, 5, 11, 15, and 16.

[Gen09a] C. Gentry. *A fully homomorphic encryption scheme*. Ph.D. thesis, Stanford University, 2009. http://crypto.stanford.edu/craig. Pages 1 and 33.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009. Pages 1 and 33.

[GHS12a] C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography*, pages 1–16. 2012. Pages 1, 9, and 35.

[GHS12b] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, pages 465–482. 2012. Page 1.

[GIKV23] R. Geelen, I. Iliashenko, J. Kang, and F. Vercauteren. On polynomial functions modulo p^e and faster bootstrapping for homomorphic encryption. In *EUROCRYPT*, pages 257–286. 2023. Pages 9 and 35.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008. Page 15.

[GSW13] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92. 2013. Pages 1, 2, 5, 34, and 44.

[GV23] R. Geelen and F. Vercauteren. Bootstrapping for BGV and BFV revisited. *J. Cryptol.*, 36(2):12, 2023. Pages 9 and 35.

[GV25a] R. Geelen and F. Vercauteren. Better GBFV bootstrapping and faster encrypted edit distance computation. Cryptology ePrint Archive, Paper 2025/1104, 2025. Pages 4, 9, 11, and 35.

[GV25b] R. Geelen and F. Vercauteren. Fully homomorphic encryption for cyclotomic prime moduli. In *EUROCRYPT*, pages 366–397. 2025. Pages 2, 3, 4, 5, 11, 15, and 16.

[HPS98] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288. 1998. Page 11.

[HS01] J. Hoffstein and J. Silverman. Optimizations for NTRU. In *Public-Key Cryptography and Computational Number Theory*, Proceedings of the International Conference organized by the Stefan Banach International Mathematical Center Warsaw, Poland, September 11-15, 2000, pages 77–88. De Gruyter, 2001. Page 11.

[HS15] S. Halevi and V. Shoup. Bootstrapping for HElib. *J. Cryptol.*, 34(1):7, 2021. Preliminary version in EUROCRYPT 2015. Pages 1, 9, and 35.

[KDE+24] A. Kim, M. Deryabin, J. Eom, R. Choi, Y. Lee, W. Ghang, and D. Yoo. General bootstrapping approach for RLWE-based homomorphic encryption. *IEEE Trans. Computers*, 73(1):86–96, 2024. Pages 9, 34, and 35.

[Kim25a] J. Kim. Bootstrapping GBFV with CKKS. Cryptology ePrint Archive, Paper 2025/888, 2025. Pages 4 and 11.

[Kim25b] J. Kim. Efficient homomorphic integer computer from CKKS. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(4):873–898, September 2025. Pages 2, 3, and 12.

[Kim26a] J. Kim. Faster homomorphic integer computer. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026. Pages 2, 3, 5, and 12.

[Kim26b] J. Kim. Faster logical operations from discrete CKKS. Cryptology ePrint Archive, Paper 2026/732, 2026. Page 12.

[KLSS23] M. Kim, D. Lee, J. Seo, and Y. Song. Accelerating HE operations from key decomposition technique. In *CRYPTO*, pages 70–92. 2023. Page 12.

[KN25] J. Kim and T. Noh. Modular reduction in CKKS. *IACR Commun. Cryptol.*, 2(2):17, 2025. Page 12.

[KSS24] J. Kim, J. Seo, and Y. Song. Simpler and faster BFV bootstrapping for arbitrary plaintext modulus from CKKS. In *ACM CCS*, pages 2535–2546. 2024. Pages 9 and 35.

[LM21] B. Li and D. Micciancio. On the security of homomorphic encryption on approximate numbers. In *EUROCRYPT*, pages 648–677. 2021. Pages 3 and 12.

[LMSS22] B. Li, D. Micciancio, M. Schultz, and J. Sorrell. Securing approximate homomorphic encryption using differential privacy. In *CRYPTO*, pages 560–589. 2022. Pages 3 and 12.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM*, 60(6):43:1–43:35, November 2013. Preliminary version in Eurocrypt 2010. Pages 1, 13, and 15.

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT*, pages 35–54. 2013. Pages 13, 15, and 48.

[LS15] A. Langlois and D. Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015. Pages 1 and 13.

[MHWW24] S. Ma, T. Huang, A. Wang, and X. Wang. Accelerating BGV bootstrapping for large p using null polynomials over $\mathbb{Z}_{p^e}$. In *EUROCRYPT*, pages 403–432. 2024. Pages 9 and 35.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. 2012. Pages 7, 9, 21, 22, and 29.

[NLV11] M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *CCSW*, pages 113–124. 2011. Page 2.

[OPP23] H. Okada, R. Player, and S. Pohmann. Homomorphic polynomial evaluation using Galois structure and applications to BFV bootstrapping. In *ASIACRYPT*, pages 69–100. 2023. Pages 9 and 35.

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97. 2010. Page 43.

[RAD78] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978. Page 1.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005. Pages 1 and 13.

[Rov26] L. Rovida. A flexible and polynomial framework for integer arithmetic in CKKS. Cryptology ePrint Archive, Paper 2026/450, 2026. Page 12.

[SV11] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014. Preliminary version in ePrint Report 2011/133. Pages 1, 3, and 12.

A Hardness of Mod-$\mathcal{L}$-LWE

In this section we give a tight reduction to mod-$\mathcal{L}$-LWE from a corresponding (plain/Module/Ring) LWE problem. Its concrete instantiation applies to the modulus lattices used in the preferred encryption algorithms for dBFV (Section 4.3), dGSW (Appendix B.1), and dBGV (Appendix C) with integer plaintext modulus. The proof follows from a simple, reversible sample-packing transform in Lemma A.2 below, followed by an (optional, non-reversible) error-shaping step. Our main focus is continuous Gaussian error, for which the error shaping is simple and exact; discrete Gaussian error follows separately by randomized rounding.

Under the natural normalizations, the full reduction is nearly lossless. The packing step combines d ordinary LWE samples into one d-dimensional sample, preserving their total dimension. And although the reduction expands the *absolute* error by the spectral norm of a known short basis of the target modulus lattice, for the gadget lattices used in our schemes this factor nearly matches the increase in modulus sparsity, so the error *rate* is nearly preserved.

Mod-$\mathcal{L}$-LWE. We first define the general form of mod-$\mathcal{L}$-LWE. As before, let K be a number field with ring of integers $R = \mathcal{O}_K$ (or, more generally, let R be any order of K). Let $\mathcal{L} \subseteq \Gamma$ be R-lattices in K^d for some positive integer d, called the *modulus lattice* and *discretization lattice*, respectively. Let $\mathcal{E}$ be either $K_{\mathbb{R}}^d$ (where $K_{\mathbb{R}} := K \otimes_{\mathbb{Q}} \mathbb{R}$), or an R-lattice in K^d containing Γ, according to whether the error is continuous or discrete, respectively. Finally, let ψ be an error distribution over $\mathcal{E}$.

Definition A.1 (Mod-$\mathcal{L}$-LWE). For $\mathbf{s} \in R^k$, the distribution $A_{\Gamma/\mathcal{L},\psi}(\mathbf{s})$ is over the domain

$$(\Gamma/\mathcal{L})^k \times (\mathcal{E}/\mathcal{L}) ;$$

a sample is generated by choosing $\mathbf{a} \leftarrow (\Gamma/\mathcal{L})^k$ uniformly at random, choosing $\mathbf{e} \leftarrow \psi$, and outputting

$$(\mathbf{a} \ , \ \mathbf{b} = \langle \mathbf{a}, \mathbf{s} \rangle + \mathbf{e} \bmod \mathcal{L}) .$$

Here $\langle \mathbf{a}, \mathbf{s} \rangle := \sum_{i=1}^k a_i s_i \in \Gamma/\mathcal{L}$ is the R-linear combination of the entries of $\mathbf{a}$, viewed naturally in $\mathcal{E}/\mathcal{L}$. For the chosen $\mathcal{E}$, define $U_{\Gamma/\mathcal{L}}$ to be the uniform distribution over $(\Gamma/\mathcal{L})^k \times (\mathcal{E}/\mathcal{L})$.

For any specified secret distribution χ over R^k, the search-$\mathsf{LWE}_{\Gamma/\mathcal{L},\psi,\chi,\ell}$ problem is to recover $\mathbf{s} \leftarrow \chi$ given ℓ independent samples from $A_{\Gamma/\mathcal{L},\psi}(\mathbf{s})$. The decision-$\mathsf{LWE}_{\Gamma/\mathcal{L},\psi,\chi,\ell}$ problem is to distinguish ℓ such samples (again for $\mathbf{s} \leftarrow \chi$) from ℓ independent samples from the corresponding $U_{\Gamma/\mathcal{L}}$.

When $d = 1$, this is simply rank-k Module-LWE with generalized modulus and discretization lattices. The usual "torus" normalization takes $\mathcal{L} = R$ and $\Gamma = \mathfrak{a}$, where $\mathfrak{a} \supseteq R$ is a fractional ideal; a common choice is $\mathfrak{a} = q^{-1}R$ for an integer $q \geq 2$. This specializes further to plain LWE when $K = \mathbb{Q}$, and to Ring-LWE when $k = 1$.

Sample packing. Here we give the heart of the hardness proof for mod-$\mathcal{L}$-LWE: an invertible transform that packs d Module-LWE samples into one d-dimensional sample, while linearly transforming their errors.

Lemma A.2 (Sample packing). *Let $\mathcal{L} \subseteq \Gamma$ be R-lattices in K, let $\mathcal{E}$ be either $K_{\mathbb{R}}$ or an R-lattice in K containing Γ, and let ψ be a distribution over $\mathcal{E}$. Let $\mathbf{B} \in K^{d \times d}$ be invertible, and define*

$$\mathcal{L}' := \mathbf{B} \cdot \mathcal{L}^{\oplus d}, \qquad \Gamma' := \mathbf{B} \cdot \Gamma^{\oplus d}, \qquad \mathcal{E}' := \mathbf{B} \cdot \mathcal{E}^{\oplus d}, \qquad \psi' := \mathbf{B} \cdot \psi^d ,$$

where ψ^d is the product distribution over $\mathcal{E}^{\oplus d}$. (Notice that $\mathcal{E}'$ is an R-lattice containing Γ' in the discrete setting, and $\mathcal{E}' = K_{\mathbb{R}}^d$ in the continuous setting.) There is a polynomial-time computable and invertible deterministic transform that:

1. *maps d independent samples from $U_{\Gamma/\mathcal{L}}$ to one sample from $U_{\Gamma'/\mathcal{L}'}$, and*
2. *maps d independent samples from $A_{\Gamma/\mathcal{L},\psi}(\mathbf{s})$ to one sample from $A_{\Gamma'/\mathcal{L}',\psi'}(\mathbf{s})$, for any $\mathbf{s} \in R^k$.*

Proof. Arrange the left components of the d input samples as the rows of $\mathbf{A} \in (\Gamma/\mathcal{L})^{d \times k}$, and their right components as a column vector $\mathbf{b} \in (\mathcal{E}/\mathcal{L})^d$. The transform applies $\mathbf{B}$ to each column of $\mathbf{A}$ and to $\mathbf{b}$, and outputs

$$\left(\mathbf{a}' = \mathbf{BA} \ , \ \mathbf{b}' = \mathbf{Bb}\right) .$$

Here the k columns of $\mathbf{BA}$ are the entries of $\mathbf{a}'$. Multiplication by $\mathbf{B}$ induces the additive-group isomorphisms

$$\Gamma^{\oplus d}/\mathcal{L}^{\oplus d} \longrightarrow \Gamma'/\mathcal{L}' \quad \text{and} \quad \mathcal{E}^{\oplus d}/\mathcal{L}^{\oplus d} \longrightarrow \mathcal{E}'/\mathcal{L}' .$$

Therefore, the transform is efficiently invertible via multiplication by $\mathbf{B}^{-1}$, and it maps the relevant uniform distributions to each other. For LWE samples, we have that $\mathbf{b} = \mathbf{As} + \mathbf{e} \bmod \mathcal{L}^{\oplus d}$, where $\mathbf{e}$ is drawn from ψ^d. By linearity,

$$\mathbf{b}' = \mathbf{Bb} = \langle \mathbf{a}', \mathbf{s} \rangle + \mathbf{Be} \pmod{\mathcal{L}'} .$$

The last error term has distribution $\psi' = \mathbf{B} \cdot \psi^d$, proving both distributional claims. □

Reshaping skewed Gaussian error. The error distortion in Lemma A.2 is exactly the linear transform of $\mathbf{B}$, which is why a small spectral norm for $\mathbf{B}$ is desired. Moreover, a distorted continuous Gaussian error can be transformed to a spherical one of essentially the same width by adding a compensating Gaussian, as follows. Apply the canonical embedding to each coordinate of $K_{\mathbb{R}}^d$, with the n embeddings of each coordinate grouped consecutively, and let

$$\mathbf{B}_{\boldsymbol{\sigma}} := (\mathrm{diag} \circ \boldsymbol{\sigma})(\mathbf{B}) \in \mathbb{C}^{dn \times dn}$$

denote the resulting $d \times d$ block matrix, where $\mathrm{diag} \circ \boldsymbol{\sigma}$ is applied entry-wise to $\mathbf{B}$ to get diagonal blocks. Write $\mathbf{B}_{\boldsymbol{\sigma}}^*$ for its conjugate transpose. For any $r > 0$, let D_r denote the spherical Gaussian on $K_{\mathbb{R}}$ with covariance parameter $r^2\mathbf{I}$. Then $\mathbf{B} \cdot D_r^d$ is an elliptical Gaussian of (positive definite) covariance parameter

$$\Sigma = r^2 \mathbf{B}_{\boldsymbol{\sigma}} \mathbf{B}_{\boldsymbol{\sigma}}^* \, .$$

Writing $\sigma_1(\cdot)$ for the largest singular value, we have $\lambda_{\max}(\Sigma) = r^2 \sigma_1(\mathbf{B}_{\boldsymbol{\sigma}})^2$. Therefore, for any $s \geq r\sigma_1(\mathbf{B}_{\boldsymbol{\sigma}})$, adding an independent Gaussian of covariance parameter $s^2\mathbf{I} - \Sigma$ yields a spherical Gaussian of parameter s. For the minimal choice $s = r\sigma_1(\mathbf{B}_{\boldsymbol{\sigma}})$, this does not increase the maximum directional standard deviation of the error. In the reduction, we add independent samples from this compensating Gaussian to the right components $\mathbf{b}'$ of the packed samples; this replaces the transformed error distribution by the desired spherical Gaussian, while preserving uniformity in the uniform case.

Discretizing the error. Any continuous-error mod-$\mathcal{L}$-LWE problem reduces to a corresponding discrete-error version over any desired superlattice of the discretization lattice, via a simple rounding step on the second components of the samples. Let $\mathcal{L} \subseteq \Gamma \subseteq \Gamma'$ be R-lattices in K^d. Given a continuous-error mod-$\mathcal{L}$-LWE sample $(\mathbf{a}, \mathbf{b})$ over $(\Gamma/\mathcal{L})^k \times (K_{\mathbb{R}}^d/\mathcal{L})$, lift the coset $\mathbf{b} + \Gamma'$ to some $\mathbf{r}$ as in Section 2.3, and replace $\mathbf{b}$ by $\mathbf{b} - \mathbf{r} \in \Gamma'/\mathcal{L}$. For an LWE sample $\mathbf{b} = \langle \mathbf{a}, \mathbf{s} \rangle + \mathbf{e} \bmod \mathcal{L}$, the coset that is lifted is merely $\mathbf{e} + \Gamma'$, because $\langle \mathbf{a}, \mathbf{s} \rangle \in \Gamma/\mathcal{L} \subseteq \Gamma'/\mathcal{L}$. Thus, the output error is $\mathbf{e} - \mathbf{r} \in \Gamma'$; in other words, the rounding passes through to the continuous error. If $\mathbf{b}$ is uniform over $K_{\mathbb{R}}^d/\mathcal{L}$, then $\mathbf{b} - \mathbf{r}$ is uniform over $\Gamma'/\mathcal{L}$: translating $\mathbf{b}$ by any element of $\Gamma'/\mathcal{L}$ does not change its coset modulo Γ', and translates $\mathbf{b} - \mathbf{r}$ by the same element. Therefore, hardness for any continuous error distribution immediately implies hardness for the induced discrete error distribution of $\mathbf{e} - \mathbf{r}$ over Γ'. In particular, to map spherical continuous Gaussian error to spherical discrete Gaussian error (up to negligible statistical distance), the lifting rule can be instantiated by randomized nearest-plane rounding as in [Pei10, Theorem 3.1 and Section 4.1].

It is also possible to switch the multiplier components $\mathbf{a}$ to a different discretization lattice, using the same kind of rounding procedure. However, this introduces a *secret-dependent* rounding error into the resulting sample, so we do not state a generic hardness reduction for that operation.

Putting everything together. Let $\mathcal{L}'$ be any target modulus lattice having a known R-basis matrix $\mathbf{B} \in K^{d \times d}$, so $\mathcal{L}' = \mathbf{B} \cdot R^{\oplus d}$, and let h be a positive integer. Taking $\mathcal{L} = R$ and $\Gamma = h^{-1}R$ in Lemma A.2 gives

$$\Gamma' = \mathbf{B} \cdot (h^{-1}R)^{\oplus d} = h^{-1}\mathcal{L}' \quad \text{and hence} \quad \Gamma'/\mathcal{L}' \cong (h^{-1}R/R)^d \, .$$

Therefore, setting $s = r\sigma_1(\mathbf{B}_{\boldsymbol{\sigma}})$, the lemma and continuous Gaussian compensation give, for every χ and ℓ, reductions from the search (respectively, decision) Module-LWE problem

$$\mathsf{LWE}_{h^{-1}R/R, D_r, \chi, d\ell} \quad \text{to} \quad \mathsf{LWE}_{h^{-1}\mathcal{L}'/\mathcal{L}', D_s^d, \chi, \ell} \, .$$

The reductions preserve search success probability and decision advantage; the packing step invertibly combines d input samples per output sample, preserving their total dimension, and the error parameter expands by the factor $\sigma_1(\mathbf{B}_{\boldsymbol{\sigma}})$.

For the application to dBFV, fix $d \geq 1$, let $\mathfrak{p} = pR$ for an integer $p \geq 2$, and take the target modulus lattice $\mathcal{L}' = \mathcal{L}^{<d}$. Because $pR^d \subseteq \mathcal{L}^{<d}$, choosing h to be a multiple of p ensures that the induced discretization lattice $h^{-1}\mathcal{L}^{<d}$ contains R^d, as required for dBFV encryption. Choose an integer $b \geq 2$ such that $b^d \geq p$. Let $\mathbf{r}(B) = \sum_{i<d} r_i B^i$ be the base-b expansion of $b^d - p$, with $0 \leq r_i < b$, and take $\mathbf{B}$ to be the basis matrix of $\mathcal{L}^{<d}$ constructed in Equation (4.5). It is the bidiagonal basis from the perfect-power case, with the rank-one correction that subtracts the coefficient vector of $\mathbf{r}$ from its last column. Because all its entries are integers, the triangle inequality gives

$$\sigma_1(\mathbf{B}_{\boldsymbol{\sigma}}) = \sigma_1(\mathbf{B}) \leq b + 1 + \|\mathbf{r}\|_2 \leq b + 1 + \sqrt{d}(b-1) \; .$$

Taking $b = \lceil p^{1/d} \rceil$, the spectral factor is within an $O(\sqrt{d})$ factor of the dimension-normalized sparsity $p^{1/d}$ of $\mathcal{L}^{<d}$ relative to R^d. In the simpler case $p = b^d$, we have $\mathbf{r} = \mathbf{0}$ and the sharper bound $b + 1 = p^{1/d} + 1$, which matches that sparsity up to an additive 1.

B Decomposed GSW

In this section we apply the ideas behind dBFV to give a "decomposed" variant of the FHE scheme of Gentry, Sahai, and Waters (GSW) [GSW13], which can handle high-precision plaintext rings with much larger error rates than ordinary GSW requires. Given the machinery of the dBFV scheme, this extension is fairly straightforward, using the fact that a GSW ciphertext essentially consists of multiple BFV ciphertexts encrypting various "scalings" of the plaintext. Analogously to our presentation of (d)BFV, we operate at an appropriate "scale," and essentially replace the plaintext ring $R/\mathfrak{p}$ with the isomorphic ring $R[B]/\mathcal{L}$, and replace the ciphertext R-module $K/\mathfrak{p}$ with the $R[B]$-module $K[B]/\mathcal{L}$.

Adopt the same mathematical setup as in Section 3.1. For notational simplicity, for polynomials in B we omit the dependence on this variable. Similarly to Section 3.4.2, let $\mathbf{g}$ be a "gadget" row vector over $K/\mathfrak{p} = K/\mathcal{L}^{<1}$ that (without loss of generality) has 1 as an entry, such that any element $a \in K[B]/\mathcal{L}$ arising from the employed discretization lattice can be efficiently expanded to a "short" column vector $\mathbf{g}^{-1}(a)$ over $R[B]$ such that $a = \mathbf{g} \cdot \mathbf{g}^{-1}(a) \pmod{\mathcal{L}}$. Apply $\mathbf{g}^{-1}$ entry-wise to any vector or matrix $\mathbf{A}$ over $K[B]/\mathcal{L}$, so that $(\mathbf{M} \otimes \mathbf{g}) \cdot \mathbf{g}^{-1}(\mathbf{A}) = \mathbf{MA} \pmod{\mathcal{L}}$ for any matrix $\mathbf{M}$ over $R[B]$ of compatible dimensions.

Definition B.1 (dGSW ciphertext). A *dGSW ciphertext* that encrypts plaintext $\mu \in R/\mathfrak{p}$ under a secret key row vector $\mathbf{s} = (-\bar{\mathbf{s}}, 1) \in R^{k+1}$ is a matrix $\mathbf{C}$ over $K[B]/\mathcal{L}$ that satisfies the (modular) decryption relation

$$\mathbf{sC} = \bar{\mu} \cdot (\mathbf{s} \otimes \mathbf{g}) + \mathbf{e} \pmod{\mathcal{L}} \tag{B.1}$$

where $\bar{\mu} \in \mu + \mathcal{L} \subseteq R[B]$ is a (typically "small") digit decomposition of μ, and $\mathbf{e}$ is a row vector over $K[B]$ whose entries have coefficients that are decodable mod R.

As with dBFV, any specific dGSW ciphertext will be over $K^{<d}[B]/\mathcal{L}^{<d}$ for some degree bound d. Ordinary GSW (at our "scaling") is simply the special case $d = 1$, because $K^{<1}[B]/\mathcal{L}^{<1} = K/\mathfrak{p}$.

We emphasize that, unlike for dBFV ciphertexts, *the choice of digit decomposition $\bar{\mu} \in \mu + \mathcal{L}$ matters* in the above definition. This is because different choices of $\bar{\mu}$ typically yield different vectors $\bar{\mu} \cdot (\mathbf{s} \otimes \mathbf{g})$ over $K[B]/\mathcal{L}$, since $(\mathbf{s} \otimes \mathbf{g})$ is over $K/\mathfrak{p}$ and $\mathcal{L} \cdot K \not\subseteq \mathcal{L}$. Typically, we need to ensure that $\bar{\mu}$ is a "small" decomposition, because it expands errors under homomorphic multiplication; see Lemma B.2 below.

B.1 Basic Operations

We first describe several basic operations for the dGSW scheme.

Decryption via "downgrading" to dBFV. Since $\mathbf{s} \otimes \mathbf{g}$ has 1 as an entry, the corresponding column $\mathbf{c}$ of $\mathbf{C}$ can be seen as a dBFV ciphertext that encrypts μ with error equal to the corresponding entry $e \in K[B]$ of $\mathbf{e}$: we have that $\langle \mathbf{s}, \mathbf{c} \rangle = \bar{\mu} + e = \mu + e \pmod{\mathcal{L}}$. So, we can decrypt a dGSW ciphertext simply by "downgrading" to a dBFV ciphertext in this way, and decrypting it.

Keeping the degree in B bounded. As we will see below in Lemma B.2, the degree in B of dGSW ciphertexts typically grows with homomorphic multiplication. We can always reduce the degree below some fixed bound by applying a K-linear *degree reduction* map $\phi\colon K[B] \to K^{<d}[B]$ that satisfies the hypotheses of Lemma 4.9 to (each entry of) the ciphertext $\mathbf{C}$. This preserves the plaintext and applies ϕ to the entries of the error vector, because by the properties of ϕ and the fact that $\mathbf{s} \otimes \mathbf{g}$ is over $K/\mathfrak{p}$,

$$\mathbf{s} \cdot \phi(\mathbf{C}) = \phi(\mathbf{s}\mathbf{C}) = \phi\big(\bar{\mu} \cdot (\mathbf{s} \otimes \mathbf{g}) + \mathbf{e} + \mathcal{L}\big) = \phi(\bar{\mu}) \cdot (\mathbf{s} \otimes \mathbf{g}) + \phi(\mathbf{e}) \pmod{\mathcal{L}^{<d}} ,$$

and $\phi(\bar{\mu}) \in \phi(\mu + \mathcal{L}) \subseteq \mu + \mathcal{L}^{<d}$ is a digit decomposition of μ. Note that the new plaintext decomposition $\phi(\bar{\mu})$ and error $\phi(\mathbf{e})$ might or might not be larger than the originals (depending on the exact definition of ϕ), but the expansion is small for the ϕ defined in Section 4.6.

Encryption. As with dBFV in Section 4.3, we describe two ways to produce a "fresh" dGSW ciphertext that encrypts a plaintext $\mu \in R/\mathfrak{p}$. The first method simply encrypts a representative $\bar{\mu} \in \mu + \mathfrak{p}$ as in ordinary GSW to get a ciphertext over $K/\mathfrak{p}$, then reduces it entry-wise modulo $\mathcal{L}$. (Concretely, we would reduce modulo $\mathcal{L}^{<d}$ for some desired d.) Security follows directly from that of the GSW scheme, which is based on Module-LWE with modulus $\mathfrak{p}$ and an error distribution that is decodable mod R. As with dBFV, this is the primary drawback of this first method: for very "sparse" moduli $\mathfrak{p}$, the error rate relative to the modulus is very small, which requires correspondingly large numerical precision and LWE dimensions for security.

The second method, similarly to what is done for dBFV in Section 4.3, directly constructs a dGSW ciphertext using mod-$\mathcal{L}^{<d}$-LWE with secret $\bar{\mathbf{s}} \in R^k$. Fix a sufficiently fine discretization lattice $\Gamma \supseteq R^{<d}[B]$ in $K^{<d}[B]$ (for a reduction-backed instantiation with $\mathfrak{p} = pR$, take $\Gamma = h^{-1}\mathcal{L}^{<d}$ for a sufficiently large positive integer h divisible by p). For the plaintext μ, choose a (typically small) digit decomposition $\bar{\mu} \in \mu + \mathcal{L}^{<d}$. Let $\mathbf{G} := \mathbf{I}_{k+1} \otimes \mathbf{g}$, so that $\mathbf{s}\mathbf{G} = \mathbf{s} \otimes \mathbf{g}$.

1. Sample error vector $\mathbf{e}$ over Γ of the same dimension as $\mathbf{s} \otimes \mathbf{g}$, whose entries are drawn independently from a suitable error distribution (where the coefficients are decodable mod R).

2. Sample a matrix $\mathbf{A}$, having the same number of columns as $\mathbf{s} \otimes \mathbf{g}$, whose columns are uniformly random over $(\Gamma/\mathcal{L}^{<d})^k$.

3. Let $\mathbf{b} = \bar{\mathbf{s}}\mathbf{A} + \mathbf{e}$, which is over $\Gamma/\mathcal{L}^{<d}$, and output the dGSW ciphertext

$$\mathbf{C} = \begin{pmatrix} \mathbf{A} \\ \mathbf{b} \end{pmatrix} + \bar{\mu} \cdot \mathbf{G} .$$

(For security, the entries of $\mathbf{C}$ must be represented by *distinguished representatives* modulo $\mathcal{L}^{<d}$.)

By inspection, the output ciphertext satisfies Definition B.1. Under the decisional mod-$\mathcal{L}^{<d}$-LWE assumption (with discretization lattice Γ and the employed secret and error distributions), $\left(\begin{smallmatrix} \mathbf{A} \\ \mathbf{b} \end{smallmatrix}\right)$ is pseudorandom.[26] For any

[26] In particular, for the instantiation $\Gamma = h^{-1}\mathcal{L}^{<d}$ above with discretized Gaussian error as in Appendix A, this assumption follows by a nearly lossless reduction from the corresponding ordinary (plain/Module/Ring) LWE assumption.

plaintext, possibly chosen adaptively by the adversary, adding the efficiently computable plaintext-dependent offset $\bar{\mu}\mathbf{G}$ preserves pseudorandomness, so the resulting ciphertext is pseudorandom and the encryption is IND-CPA secure.

Just as with dBFV, this second method provides a better tradeoff between the error rate of fresh ciphertexts and their homomorphic capacity. Moreover, because (d)GSW is typically used for homomorphic computations that induce fairly small error growth, this advantage is much more pronounced: the *dominant* factor in the error rate of fresh ciphertexts is simply the "sparsity" of the modulus lattice, which this second method improves from p to only about $p^{1/d}$ (for integer modulus p). So, for a fixed security level, the main term in the LWE dimension and numerical precision is reduced by a factor of about d.

B.2 Homomorphic Operations

Let $\mathbf{C}, \mathbf{C}'$ be dGSW ciphertexts that respectively encrypt plaintexts $\mu, \mu' \in R/\mathfrak{p}$ via digit decompositions $\bar{\mu}, \bar{\mu}' \in R[B]$, with error vectors $\mathbf{e}, \mathbf{e}'$, under the same secret key $\mathbf{s}$. By linearity and the additive homomorphism of digit decompositions, it is immediate that $\mathbf{C}_+ := \mathbf{C} + \mathbf{C}'$ encrypts $\mu + \mu' \in R/\mathfrak{p}$ with error vector $\mathbf{e}_+ = \mathbf{e} + \mathbf{e}'$ under $\mathbf{s}$. For homomorphic multiplication, we have the following.

Lemma B.2. *The dGSW ciphertext* $\mathbf{C}_\times := \mathbf{C} \cdot \mathbf{g}^{-1}(\mathbf{C}')$ *encrypts* $\mu\mu' \in R/\mathfrak{p}$ *under* $\mathbf{s}$*, with error vector* $\mathbf{e}_\times = \bar{\mu} \cdot \mathbf{e}' + \mathbf{e} \cdot \mathbf{g}^{-1}(\mathbf{C}')$.

Before giving the (routine) proof, we point out a few important consequences of the lemma. First notice that, "downgrading" $\mathbf{C}'$ and $\mathbf{C}_\times$ to dBFV ciphertexts by selecting their appropriate column vectors $\mathbf{c}'$ and $\mathbf{c}_\times$ (respectively), we see that $\mathbf{c}_\times = \mathbf{C} \cdot \mathbf{g}^{-1}(\mathbf{c}')$ is a dBFV ciphertext that encrypts $\mu\mu'$ with error $\bar{\mu} \cdot e' + \mathbf{e} \cdot \mathbf{g}^{-1}(\mathbf{c}')$, where e' is the corresponding entry of $\mathbf{e}'$. In the literature this is often called a GSW-by-BFV "external product," which is more efficient than a GSW-by-GSW product, and can suffice for bootstrapping and other purposes.

Also observe that, as with ordinary GSW, the error growth is asymmetric: the error vector $\mathbf{e}$ from $\mathbf{C}$ is expanded by the fairly small matrix $\mathbf{g}^{-1}(\mathbf{C}')$, whereas the error vector $\mathbf{e}'$ from $\mathbf{C}'$ is expanded by the *plaintext decomposition* $\bar{\mu}$ in $\mathbf{C}$. Following [BV14, AP14, DM15, CGGI16], we can use "non-expanding" $\bar{\mu}$—e.g., roots of unity in R—to support a sequence of many multiplications that incurs just *additive* noise, enabling bootstrapping with only polynomial noise growth. In addition, digit decompositions can provide a larger set of non-expanding (or mildly expanding) elements in $R[B]$, which may enable new kinds of efficient homomorphic computations. We leave an exploration of this idea to future work.

Proof of Lemma B.2. We verify the decryption relation (Equation (B.1)) using the hypotheses:

$$\begin{aligned}
\mathbf{s} \cdot \mathbf{C}_\times &= \mathbf{s}\mathbf{C} \cdot \mathbf{g}^{-1}(\mathbf{C}') \\
&= (\bar{\mu} \cdot (\mathbf{s} \otimes \mathbf{g}) + \mathbf{e}) \cdot \mathbf{g}^{-1}(\mathbf{C}') \\
&= \bar{\mu} \cdot \mathbf{s}\mathbf{C}' + \mathbf{e} \cdot \mathbf{g}^{-1}(\mathbf{C}') \\
&= \bar{\mu} \cdot (\bar{\mu}' \cdot (\mathbf{s} \otimes \mathbf{g}) + \mathbf{e}') + \mathbf{e} \cdot \mathbf{g}^{-1}(\mathbf{C}') \\
&= (\bar{\mu}\bar{\mu}') \cdot (\mathbf{s} \otimes \mathbf{g}) + (\bar{\mu} \cdot \mathbf{e}' + \mathbf{e} \cdot \mathbf{g}^{-1}(\mathbf{C}')) \,,
\end{aligned}$$

and $\bar{\mu}\bar{\mu}' \in \mu\mu' + \mathcal{L}$ is a decomposition of $\mu\mu' \in R/\mathfrak{p}$, as needed. □

C Decomposed BGV

Here we apply the ideas underlying dBFV to construct a "decomposed" variant of the Brakerski–Gentry–Vaikuntanathan (BGV) FHE scheme [BGV12].[27] The central idea is to combine the plaintext and error into one short element of the plaintext coset of the gadget lattice. This makes homomorphic addition and multiplication especially transparent; the main points requiring care are fresh-encryption security and ciphertext-modulus reduction.

Noisy plaintext encodings. Recall the gadget-lattice ideal $\mathcal{L} = \mathcal{L}_{\mathfrak{p}}^{\perp}(b) \subseteq R[B]$ from Section 4.1. A *noisy encoding* of plaintext $\mu \in R/\mathfrak{p}$ is a short polynomial

$$\mathbf{e}(B) \in \mu + \mathcal{L}.$$

Equivalently, it is a short (not necessarily canonical) digit decomposition of μ. (For degree bound $d = 1$ and $\mathfrak{p} = pR$, this specializes to $e \in \mu + pR$, exactly the usual BGV notion of a short noise term.) Because $\mathcal{L}$ is an ideal, noisy encodings of μ and μ' add and multiply to noisy encodings of $\mu + \mu'$ and $\mu\mu'$, respectively.

Ciphertexts. For a positive integer ciphertext modulus q that is a unit modulo $\mathfrak{p}$, define the quotient ring $R_q := R/qR$. (When $\mathfrak{p} = pR$ for an integer p, this condition is equivalent to $\gcd(p, q) = 1$.) A ciphertext encrypting a plaintext $\mu \in R/\mathfrak{p}$ under a secret key $\mathbf{s} \in R^k$ is a polynomial

$$\mathbf{c}(B, S) \in R_q[B, S]$$

such that

$$\mathbf{c}(B, \mathbf{s}) = \mathbf{e}(B) \pmod{qR[B]}$$

for some noisy encoding $\mathbf{e}(B) \in \mu + \mathcal{L}$ whose coefficients are decodable mod qR. Decryption lifts $\mathbf{c}(B, \mathbf{s}) \in R_q[B]$ to the decodable representative $\mathbf{e}(B) \in R[B]$ and then recovers μ from its coset modulo $\mathcal{L}$, or equivalently by evaluating at $B = b$ modulo $\mathfrak{p}$.

Encryption. To encrypt, first fix a degree bound d and sample a noisy encoding $\mathbf{e}(B)$ from a suitable efficiently samplable probability distribution over $\mu + \mathcal{L}^{<d}$. With this noisy encoding, generate a fresh affine-linear ciphertext by choosing uniformly random $\mathbf{c}_1(B) \leftarrow (R_q^{<d}[B])^k$, letting

$$c_0(B) := \mathbf{e}(B) - \langle \mathbf{c}_1(B), \mathbf{s} \rangle \in R_q^{<d}[B],$$

and outputting

$$\mathbf{c}(B, S) := c_0(B) + \langle \mathbf{c}_1(B), S \rangle.$$

Below we establish IND-CPA security based on a mod-$\mathcal{L}^{<d}$-LWE problem with corresponding error distribution.

[27] The key observations underlying this dBGV system are due to Simon Pohmann, who identified them after our initial unsuccessful attempts, once the gadget lattice had been formalized as an ideal of $R[B]$.

Homomorphic operations. For ciphertexts with the same modulus q, homomorphic addition and multiplication are simply polynomial addition and multiplication. Indeed, their resulting noisy encodings are $\mathbf{e}(B) + \mathbf{e}'(B)$ and $\mathbf{e}(B)\mathbf{e}'(B)$, respectively, which encode the sum and product of the plaintexts. As in dBFV and ordinary BGV, these operations increase the size and degrees in B and S of the ciphertext and its noisy encoding. Degree in S is reduced by ordinary BGV key switching. Degree in B is reduced as in dBFV, by replacing each high power B^j with a short bounded-degree representative $\mathbf{r}_j(B) = B^j \pmod{\mathcal{L}}$. This changes the noisy encoding only by an element of $\mathcal{L}$, and hence preserves the plaintext. The absolute size of the noisy encoding is controlled by ciphertext-modulus reduction, described next.

Ciphertext-modulus reduction. The more delicate operation changes the ciphertext modulus from q to some smaller q', while shrinking the absolute size of the underlying noisy encoding by a factor of about q/q'. All of the following is the gadget-lattice analog of ordinary BGV modulus reduction.

For the cleanest form, take $q' = q \pmod{\mathfrak{p}}$, so q' is also a unit modulo $\mathfrak{p}$ and the plaintext can be preserved exactly. Set $\alpha = q'/q > 0$. Multiplication by α defines a map from $R_q[B]$ to $K[B]/q'R[B]$, because scaling by α maps $qR[B]$ to $q'R[B]$. Moreover, for any $c(B) \in R_q[B]$, the coset

$$(1-\alpha) \cdot c(B) + \mathcal{L}$$

is well defined, because $(1-\alpha) \cdot qR[B] = (q-q') \cdot R[B] \subseteq \mathfrak{p}[B] \subseteq \mathcal{L}$.

For every coefficient $c(B)$ of the ciphertext (as a polynomial in S), use a short basis of a suitable bounded-degree restriction of $\mathcal{L}$ to compute a short representative

$$f(B) \in (1-\alpha) \cdot c(B) + \mathcal{L},$$

and replace $c(B)$ by the rounded value

$$c'(B) := \alpha \cdot c(B) + f(B) \in R_{q'}[B] \; .$$

Collect the $f(B)$ into the polynomial $\mathbf{f}(B, S)$. Letting $\mathbf{e}(B)$ be the original noisy encoding, the new one has the form

$$\mathbf{e}'(B) := \alpha \cdot \mathbf{e}(B) + \mathbf{f}(B, \mathbf{s}) \; .$$

The choice of the $f(B)$ and the fact that $\mathbf{s}$ is over R imply that

$$\mathbf{f}(B, \mathbf{s}) = (1-\alpha) \cdot \mathbf{c}(B, \mathbf{s}) = (1-\alpha) \cdot \mathbf{e}(B) \pmod{\mathcal{L}} \; ,$$

and hence $\mathbf{e}'(B) \in \mathbf{e}(B) + \mathcal{L}$. Therefore, the plaintext is preserved, and the absolute error is scaled by the desired factor $\alpha = q'/q$, apart from an additive rounding term $\mathbf{f}(B, \mathbf{s})$ whose size is governed by the gadget lattice $\mathcal{L}$ and the secret (but is independent of q, q').

For a general q' that is also a unit modulo $\mathfrak{p}$, let $v := q'q^{-1} \in R/\mathfrak{p}$ and instead take $f(B) \in (v-\alpha) \cdot c(B) + \mathcal{L}$. This coset is well defined because $vq = q'$ in $R/\mathfrak{p}$ and $\mathfrak{p}[B] \subseteq \mathcal{L}$. The same calculation as above gives $\mathbf{e}'(B) \in v\mathbf{e}(B) + \mathcal{L}$. As in ordinary BGV, the resulting plaintext factor v can be tracked and removed upon decryption.

Security. IND-CPA security follows from the natural generalization of [LPR13, Lemma 2.23 and Section 8.3] to the gadget lattice, as we now describe. As in Section 2.3, discretizing a point $\mathbf{x} \in K^d_{\mathbb{R}}$ into a target coset $\mathbf{u} + \Lambda \subseteq K^d$ means lifting $(\mathbf{u} - \mathbf{x}) + \Lambda$ to $\mathbf{f}$, and outputting $\mathbf{x} + \mathbf{f} \in \mathbf{u} + \Lambda$.

Now define $\Lambda := \mathcal{L}^{<d}$, identify $R/\mathfrak{p}$ with $R^{<d}[B]/\Lambda$ via constant polynomials, and let χ be the secret-key distribution over R^k. Suppose that, uniformly over all plaintexts $\mu \in R/\mathfrak{p}$, the noisy-encoding distribution used in encryption is within negligible statistical distance of sampling an error from a continuous distribution ψ over $K_{\mathbb{R}}^d$ and discretizing it into $\mu + \Lambda$ using some efficient lifting rule for Λ. (No particular shape, such as spherical or even Gaussian, is required of ψ.) Assume that decision-$\mathsf{LWE}_{\Lambda/q\Lambda,\psi,\chi,\ell}$ (Definition A.1) is hard, where ℓ is a polynomial bound on the number of ciphertexts the adversary gets.[28] Because q is a unit modulo $\mathfrak{p}$, reduction modulo $qR^{<d}[B]$ induces, for every plaintext μ, a bijection

$$(\mu + \Lambda)/q\Lambda \longrightarrow R_q^{<d}[B] \ .$$

For any desired plaintext μ, we transform a given sample $(\mathbf{a}, \mathbf{b}) \in (\Lambda/q\Lambda)^k \times (K_{\mathbb{R}}^d/q\Lambda)$ as follows. The coset $(\mu - \mathbf{b}) + \Lambda$ is well defined because $q\Lambda \subseteq \Lambda$. Lift $(\mu - \mathbf{b}) + \Lambda$ to $\mathbf{f}$, and, using the bijections above, output the ciphertext whose coefficients are

$$\mathbf{c}_1(B) := -\mathbf{a}, \qquad c_0(B) := \mathbf{b} + \mathbf{f} \pmod{q\Lambda} \ .$$

If the samples are drawn from the LWE distribution $A_{\Lambda/q\Lambda,\psi}(\mathbf{s})$, then $\mathbf{b} = \langle \mathbf{a}, \mathbf{s} \rangle + \mathbf{e}$ modulo $q\Lambda$ for some $\mathbf{e} \leftarrow \psi$. Because $\langle \mathbf{a}, \mathbf{s} \rangle \in \Lambda$, the coset being lifted is $(\mu - \mathbf{e}) + \Lambda$, so $\mathbf{e} + \mathbf{f}$ has the noisy-encoding distribution and is independent of the uniform $\mathbf{c}_1(B) = -\mathbf{a}$. Together with the definition of $c_0(B)$, this is exactly the fresh-ciphertext distribution, up to the assumed negligible statistical distance. Therefore, samples from $A_{\Lambda/q\Lambda,\psi}(\mathbf{s})$ for $\mathbf{s} \leftarrow \chi$ map to fresh ciphertexts under $\mathbf{s}$ of the desired plaintext.

If the samples are drawn from the uniform distribution $U_{\Lambda/q\Lambda}$, then $\mathbf{a}$ and $\mathbf{b}$ are independent and uniform over their respective domains. The isomorphism above maps $-\mathbf{a}$ uniformly to $(R_q^{<d}[B])^k$. For any $\mathbf{z} \in \Lambda/q\Lambda$, replacing $\mathbf{b}$ by $\mathbf{b} + \mathbf{z}$ leaves the coset being lifted unchanged, and hence translates the distribution of $c_0(B)$ by $\mathbf{z}$. Therefore, all elements of $(\mu + \Lambda)/q\Lambda$ occur as $c_0(B)$ with equal probability, so $c_0(B)$ is uniform in $R_q^{<d}[B]$. It is also independent of $\mathbf{c}_1(B)$, so samples from $U_{\Lambda/q\Lambda}$ map to uniform ciphertexts independently of μ.

Applying the transformation independently to each given sample shows that fresh ciphertexts (even of plaintexts adaptively chosen by the adversary) are pseudorandom. The total statistical-distance loss is at most ℓ times the per-ciphertext loss, and hence is negligible. Hence encryption is IND-CPA secure under the stated decision-$\mathsf{LWE}_{\Lambda/q\Lambda,\psi,\chi,\ell}$ assumption.

D Representations and Efficient Operations

The homomorphic encryption schemes from Sections 3 and 4 treat the field K and ring R abstractly, and require representing high-precision elements of K. They also use a variety of operations on such elements, like addition and multiplication of polynomials over K, rounding to smaller precision, decoding modulo R, gadget/digit decompositions, generation of uniformly random elements and short random error terms, etc. In this section we give an overview of how all this can be implemented efficiently (in quasi-linear time) on a computer with fixed-precision registers.

Essentially, we use the standard technique of representing and operating on high-precision elements of K "positionally," as polynomials over R with "small," fixed-precision coefficients. This directly corresponds to a gadget/digit decomposition, and rounding and decoding correspond simply to truncation of less-significant digits. Since dBFV ciphertexts are bivariate polynomials over K, elements of K can be represented as

[28] For the error distributions considered there, the reduction in Appendix A in turn tightly bases this mod-$q\Lambda$-LWE problem on ordinary Module-LWE.

polynomials over R, and elements of R are typically represented as polynomials over $\mathbb{Z}$ (with some polynomial modulus), ciphertexts are ultimately represented as multivariate polynomials with small, fixed-precision integer coefficients. There are many techniques for efficient arithmetic on such polynomials, some of which we summarize here; we expect future work to find many specific optimizations.

D.1 Digit Representation

Fix an integer base $g \geq 2$ and a precision parameter $\ell \geq 0$. Essentially, a digit representation expresses a field element of K in "base g," using "digits" in R (which may or may not be "larger than g"; see Appendix D.1.1 for discussion). More general algebraic bases $g \in R$ may also be used when equipped with a suitable finite digit system, but the needed existence and termination properties are not automatic, so we focus on integer bases.

Definition D.1 (Digit representation). A *digit representation* of an element $a \in g^{-\ell}R \subset K$ is a *Laurent* polynomial[29]

$$\mathbf{a}(G) = \sum_{i \leq \ell} a_i \cdot G^i \in R[G, G^{-1}] \quad \text{such that} \quad a = \mathbf{a}(g^{-1}) = \sum_{i \leq \ell} a_i \cdot g^{-i} \,. \tag{D.1}$$

The coefficients $a_i \in R$ of $\mathbf{a}(G)$ are called the "digits" of this representation.

Notice that this is essentially a generalization of digit decomposition for plaintexts in the ring $R/\mathfrak{p}$ (Definition 4.1) to field elements. For convenience, we evaluate $\mathbf{a}(G)$ at $G = g^{-1}$ (instead of g) so that the digits go from most- to least-significant as the degrees of their corresponding monomials increase. The monomials of non-positive degree correspond to the "integer part," i.e., the digits before the "decimal point." Similarly, the monomials of positive degree correspond to the "fractional part," i.e., the digits after the decimal point. This convention yields a uniform upper bound on the degree for a given precision (i.e., number of digits) past the decimal point, and allows for rounding off an element by truncating its higher-degree monomials (see Appendix D.1.1).

Also observe that a digit representation with *small* digits is exactly what is needed to apply key-switching, as described in Section 3.4.2. That is, to apply key-switching to a linear form $\mathbf{u}(S') = \langle \mathbf{u}, S' \rangle$ over K, we need small digit representations of the entries of $\mathbf{u}$.

Remarks. For a Laurent polynomial $\mathbf{a}(G)$, its *degree* is the largest integer k for which $a_k \neq 0$, and its *order* is the smallest integer k such that $a_k \neq 0$. (By convention, the degree and order of the zero polynomial are defined to be $-\infty$ and zero, respectively.) So, for any $a \in g^{-\ell}R$, it has a representation $\mathbf{a}(G)$ of degree at most ℓ, whose order is non-positive if and only if it has a nonzero digit in its "integer part."

The *Euclidean degree* of $\mathbf{a}(G)$ is its degree minus its order. So, we can express $\mathbf{a}(G) = G^e \cdot \mathbf{p}(G)$ where e is (at most) the order and $\mathbf{p}(G) \in R[G]$ is an ordinary polynomial whose degree is (at least) the Euclidean degree. So, representing and operating on Laurent polynomials easily reduces to the same for ordinary polynomials, with some simple bookkeeping to maintain (lower bounds on) the orders.

Finally, in the (d)BFV scheme with plaintext modulus $\mathfrak{p}$ (see Sections 3 and 4), if $g^e \in \mathfrak{p}$ for some positive integer e, then it suffices to work with representatives of K/g^eR. We can represent such elements using Laurent polynomials of order greater than $-e$, i.e., any monomials of degree $\leq -e$ can be dropped. In other words, we store no more than e digits of the "integer part," before the "decimal point." When no such e exists, the coset of the integer part modulo $\mathfrak{p}$ can instead be lifted to a short representative using the methods of Section 2.3.

[29] A Laurent polynomial is like an ordinary polynomial, but allowing for monomials with negative-integer exponents.

D.1.1 Reduction

Any Laurent polynomial $\mathbf{a}(G)$ over R that satisfies Equation (D.1) is a valid digit representation of a, and this representation is not unique; in particular, it allows for "relaxed" R-coefficients (digits) that are not distinguished representatives of R/gR. However, for computational and cryptographic purposes we need to keep the digits within some bounded range. The following is the strictest possible such condition that can be imposed. (More generally, we can reduce modulo the lattice of digit representations of zero, similarly to Section 4.6.)

Definition D.2. A digit representation $\mathbf{a}(G)$ is *reduced* if all its coefficients $a_i \in R$ are from the set of (small) distinguished representatives of R/gR.

For any fixed $e \geq 0$, every residue in $g^{-\ell}R/g^eR$ has a unique representative of the form $\mathbf{a}(g^{-1})$, where $\mathbf{a}$ is reduced and supported on degrees $-e < i \leq \ell$. Therefore, we can generate a uniformly random representative of $g^{-\ell}R/g^eR$ simply by concatenating $\ell + e$ uniformly random and independent digits (distinguished representatives of R/gR). Also observe that a reduced representation can be "rounded off" to a nearby element of precision $\ell' < \ell$ simply by truncating its least-significant $\ell - \ell'$ digits (i.e., those whose monomials have degree $> \ell'$). As an optimization, note that only the truncated digits need to be reduced for this; more generally, they need only be small (but not fully reduced) in order to round off to a nearby element.

Choosing $g = 2^w$ for a machine-word-aligned value of w makes coefficient-wise digit extraction and carry propagation simple masking and shifting operations. Moreover, because relaxed digits are allowed, an implementation can often accumulate several operations before propagating carries or otherwise reducing the representation.

As we will see below, the basic addition and multiplication operations on field elements typically yield non-reduced representations. Fortunately, modulo g^eR, any valid representation can easily be converted to reduced form, essentially by propagating "carries" (or "overflows") to more-significant digits. More specifically, first discard any terms of degree at most $-e$, then do the following for $i = \ell, \ell - 1, \ldots, 1 - e$:

1. express $a_i = q_i \cdot g + a'_i$ where $a'_i \in R$ is the distinguished representative of $a_i \bmod gR$, and $q_i \in R$;
2. let $a_{i-1} = a_{i-1} + q_i \in R$;
3. if $a_j = 0$ for all $j < i$, set $a'_j = 0$ for all such j and break out of the loop. (This is a non-constant-time optimization.)

Finally, output $\mathbf{a}'(G) = \sum_{-e<i\leq\ell} a'_i \cdot G^i$, discarding the final carry into degree $-e$. It is clear by inspection that $\mathbf{a}'(G)$ is reduced and that $\mathbf{a}'(g^{-1}) = \mathbf{a}(g^{-1}) \pmod{g^eR}$, so the output and input represent the same residue modulo g^eR.

D.2 Arithmetic

As detailed above, elements of K can be represented as polynomials in $R[G]$ with "small" coefficients, typically of moderate degree (e.g., one or two dozen). Recalling that R is typically isomorphic to a polynomial quotient ring like $\mathbb{Z}[X]/(X^n + 1)$ (see Section 2), where n is typically in the thousands, we see that an element of K can be represented as a small-coefficient polynomial in a ring like $\mathbb{Z}[X, G]/(X^n + 1)$. Also recalling that a dBFV ciphertext lifted to short representatives is a polynomial in $K[B, S]$ (see Section 4.2.2 and Definition 4.3), of moderate degree in B and at most quadratic in S, such a ciphertext can be seen as a small-coefficient polynomial in a ring like $\mathbb{Z}[X, G, B, S]/(X^n + 1)$.

We summarize some standard techniques for doing arithmetic with such polynomials, especially multiplication. Our initial implementation uses a subset of these, focused mainly on the FFT/NTT techniques; we expect that significant additional speedups are possible by more fully exploring the space of optimizations.

1. To work efficiently modulo $X^n + 1$, we can use the Fast Fourier Transform (FFT) over $\mathbb{C}$, tweaked to evaluate at just the primitive $2n$th roots of unity, so that polynomial multiplication corresponds to coordinate-wise multiplication of the Fourier coefficients.

2. Alternatively, we can use analogous fast Number Theoretic Transforms (NTTs) over one or more prime fields $\mathbb{Z}_q$, where each prime $q = 1 \pmod{2n}$ provides primitive $2n$th roots of unity. With one prime, q must be large enough that the integer coefficients do not wrap around. More commonly, several machine-word primes can be used in a residue-number-system (RNS) representation whose product exceeds the required coefficient range; values can remain in RNS form across many operations and be reconstructed only when needed for carrying, rounding, or decoding.

3. Because the degrees in G and B are moderate, direct convolution may be preferable to another transform. Depending on their sizes, one can use schoolbook multiplication, Karatsuba or Toom–Cook multiplication, and compute only the coefficients that will survive any imminent truncation in G.

4. Values that participate in many operations can remain in the FFT/NTT domain in X, avoiding repeated transforms; inverse transforms are needed only when an operation must inspect the integer coefficients, as may occur in carrying, rounding, or decoding. Transforms shared by several products can likewise be reused (or "hoisted") or batched.

5. When transform multiplication in G is advantageous, we can work modulo (say) $G^{n'} + 1$ for a power-of-two n' that exceeds the degree in G of any polynomial that might occur. The same FFT/NTT techniques as above can be applied for G, independently of X. That is, we can apply the transforms and their inverses over G and X in any order, because they all commute with each other. For example, when truncating precision, we need the polynomial to be represented in the "coefficient domain" in G, but it can remain in the "Fourier domain" in X (as long as the truncated Fourier vectors represent small polynomials).

6. Because the degree in B is somewhat small, doing FFTs/NTTs over B is not likely to yield any speedup in practice. Instead, we can use *Kronecker substitution*, substituting $X = B^{n''}$ for a (power of two) n'' that exceeds the largest degree in B that would otherwise appear. Then mod-$(X^n + 1)$ polynomials in X, B correspond unambiguously to mod-$(B^{n \cdot n''} + 1)$ polynomials in B alone, and can be handled using an nn''-dimensional (tweaked) FFT/NTT. Similar Kronecker substitutions can also be done for G and/or S, which would reduce everything to very high-degree univariate polynomials and FFTs/NTTs.

7. Alternatively for S, when the secret rank is $k = 1$, the multiplied polynomials are merely linear in the single variable S, so Karatsuba's trick uses just three multiplications of polynomials in X, G, B. For $k > 1$, direct expansion instead produces pairwise quadratic terms in the components of S, though Kronecker substitution or batched multiplication can organize their computation.