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

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

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

February 12, 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 scale only *quasi-linearly* (versus quadratically for prior 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 (hidden) 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, Kim25, BK25, BFG+25, CPL25], 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 bear out 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, often $\mathbb{Z}_2 = \{0, 1\}$. So, the complexity of a single homomorphic $\mathbb{Z}_p$-multiplication (counting just small-ring homomorphic multiplications) is still $\Omega(\log^2 p)$, 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" bits. This results in rather high concrete running times, even with optimized implementations for moderately large p (see, e.g., Figure 1).

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 "*fresh*" ciphertexts still needs to be inversely proportional to 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 the standard notion of negligible decryption error), it needs a costly transformation to prevent a *complete break* when the adversary gets decryptions of honestly generated ciphertexts [LM21, LMSS22].

For high-precision *exact* FHE, the landscape is much more fragmented. The known solutions either: are limited to 'rare' plaintext moduli of very 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 [Kim25, BK25, CPL25]; or work (only) for powers of two, but use specially crafted non-standard ciphertext rings, which require a "ground-up" implementation and lack other important FHE features [BFG+25]. 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 match those of prior schemes for much smaller 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]

- It 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 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 view 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.
- It is *flexibly parameterizable* and *practically efficient*. Specifically, it directly implements mod-p plaintext arithmetic via quasi-linear-time operations on d ordinary BFV ciphertexts, with error rates and growth at least as good as 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, this yields running times of $\tilde{O}(h^2 \log p)$ for depth-h computations, i.e., just *quasi-linear* in $\log p$, versus at least *quadratic* in $\log p$ for ordinary BGV/BFV, GSW/TFHE, and [CCKS23] (for approximate FHE).
- It 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 dBFV trivially reduces to bootstrapping its d ordinary BFV ciphertexts *in parallel*. This can be done via any previous approach, and can even be "hosted" by dBFV or dGSW—the natural decomposed analog of GSW—themselves to get bootstrapping with good error rates and growth. (See Sections 1.3.4 and 5 for details.)

Our central contribution is the *mathematical definition and analysis of dBFV*. However, as initial proofs of concept, we also wrote preliminary, unoptimized (single-threaded but AVX-enabled) CPU and (single-)GPU implementations of the scheme, and benchmarked them against ordinary BFV and TFHE for homomorphic

[1] Curiously, even though prior works seem to put BGV and BFV on essentially equal footing in terms of features and efficiency (and indeed, switching from one kind of ciphertext to the other is essentially free [AP13, Appendix A]), we have not seen a way to make our techniques work directly with BGV.

$\mathbb{Z}_{2^{64}}$-arithmetic with real-world parameters; see Figure 1.[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 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.

Our scheme'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. 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 suitably *reduced* modulo the "good" lattice encrypt noisy small plaintext decompositions, and arithmetic operations on reduced ciphertexts naturally induce the corresponding operations on the decompositions.

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 $R/\mathfrak{p}$.[3] 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]).

[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.

[3] 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	1	64	–	–	596.7	16 084	

Figure 1: 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* bit-level 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 $|e| < \frac{1}{2}$.[4] 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, because $\mathbf{s}$ is over $\mathbb{Z}$, we can "lift" the $(\mathbb{Q}/p\mathbb{Z})$-entries of $\mathbf{c}$ to their smallest (or nearly smallest) mod-p representatives in $\mathbb{Q}$, yielding some $\hat{\mathbf{c}}$ that by Equation (1.1)

[4]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.

satisfies

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

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 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.[5] Of particular interest are "small" decompositions, whose coefficients are proportional to b, but are not necessarily in $\{0, \ldots, b-1\}$.

[5] 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 computing a small decomposition means reducing to a small representative modulo $\mathcal{L}$.

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 $\boldsymbol{\mu} + \boldsymbol{\mu}', \boldsymbol{\mu} \cdot \boldsymbol{\mu}' \in \mathbb{Z}[B]$ are decompositions of $\mu + \mu', \mu \cdot \mu' \in \mathbb{Z}_p$ (respectively), simply because polynomial evaluation at b is a ring homomorphism.[6] 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 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, and this property can be maintained by a suitable kind of *reduction* on ciphertexts. 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 $\mathbb{Z}[B]/\mathcal{L}$.[7] 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 essential 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

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

[7] 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.

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.

To address this issue, in Section 4.3 we give a novel encryption method for dBFV in which error of the same *size* (over $\mathbb{Q}$) has much larger *rate* relative to the modulus lattice $\mathcal{L}$, proportional to $1/p^{1/d} \gg 1/p$. This matches the initial error rate of ordinary BFV with a modulus of about $p^{1/d}$, allowing for smaller LWE parameters at the same apparent security level—even ignoring the improved error *growth* under homomorphisms, which we describe next.

Homomorphisms and ciphertext 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 mod-$\mathcal{L}$ representatives* 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 of the mod-$\mathcal{L}$ representatives. 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.

Note that the product ciphertext will be quadratic in S, but as usual, it can be linearized by key-switching. More interesting is how we keep the ciphertext degree in B bounded, and also how we lift the ciphertext coefficients from $\mathbb{Q}[B]/\mathcal{L}$ to suitably *small* representatives in $\mathbb{Q}[B]$ (which is essential for the smaller error growth under multiplication). These are accomplished by what we call *ciphertext reduction*, which consists of *degree reduction* followed by a more standard reduction to small mod-$\mathcal{L}$ representatives of size $\approx d \cdot p^{1/d}$, using a "short" basis of $\mathcal{L}$. (See Section 4.6 for details.) 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 the underlying error polynomial) down to have bounded degree in B. For reduction modulo $\mathcal{L}$, we use the short basis of $\mathcal{L}$ given in [MP12], and its associated fast reduction algorithm for the special case $p = b^d$. For general p and b, we give a new, fast reduction algorithm that is a "hybrid" of the nearest-plane and round-off algorithms using the basis (see Section 4.6.2).

The notion of ciphertext reduction is more broadly useful—e.g., it lets 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 decomposed schemes as hosts. For a second-generation host we can immediately use dBFV itself, which (as we have already discussed) yields much better error growth and error rates in the bootstrapping key. For 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 A 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 the several prior works on FHE for high-precision plaintexts, and how our work compares to it.

BGV/BFV-style exact FHE. One line of works [GC14, CLPX18, GV25b, BFG+25] 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, 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 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. In addition, it is not known whether these schemes can be efficiently bootstrapped.
- 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 practical bootstrapping procedure, by converting its ciphertexts to ordinary BFV ciphertexts with a (large) integer modulus, and bootstrapping them. (Very recently, this method was improved [GV25a] to keep more of the computation outside of ordinary BFV, thus improving the error growth in bootstrapping.)
- Very recently, the fourth work [BFG+25]—which is concurrent to and independent of ours—gets plaintext rings of the form $\mathbb{Z}_{2^n}$ by using a specially designed *non-cyclotomic* ring $R = \mathbb{Z}[X]/(X^n - X + 2)$ and the ideal $\mathfrak{p} = (X - 2)R$. This work also describes (but does not implement, owing to its complexity) a bootstrapping method that works by sequentially doing several TFHE-based bootstraps per bit of the plaintext. This method also extends to perform other homomorphic operations like comparisons of the underlying plaintexts.

  While this natively supports plaintext rings of the form $\mathbb{Z}_{2^n}$, it also has some drawbacks:

  1. First, because it uses a completely different kind of (non-cyclotomic) ring from almost all prior work, large portions of the algorithms, analysis, and implementation for rings must be re-done from scratch. Moreover, certain powerful FHE tools like fast number-theoretic transforms (NTTs) and automorphisms are simply not available, because the ring is not Galois.
  2. Second, for exponents of interest like $n = 64$, the ring dimension n is far too small for security. To address this, [BFG+25] suggests using Module-LWE of sufficient rank over R (e.g., 16 or 32) to get a large enough total dimension for security. However, this is significantly less efficient than using Ring-LWE over a ring of sufficient dimension—especially in the FHE context, where the tensoring and relinearization in homomorphic multiplication are *quadratic* in the rank.
  3. Lastly, the bootstrapping algorithm requires several TFHE-based programmable bootstraps (PBS) *per plaintext bit*, and these must be done *sequentially* over the bits. The analysis in [BFG+25] estimates that for $n = 64$, its bootstrapping method has about half the cost of the $\Theta(n^2)$ bitwise (and bootstrapped) homomorphic operations needed for $\mathbb{Z}_{2^n}$-multiplication in TFHE.

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 for our scheme very *simply* and *generically* reduces to a small number of *parallel* bootstrappings for BFV, of which there are a variety of existing solutions for various settings (e.g., packed versus unpacked ciphertexts, using BFV or GSW as the "host" scheme, etc.). Finally, we can also support PBS-based homomorphic comparisons (see Section 1.3.5), using *fewer* and *parallel* PBS calls: we can handle a small number of plaintext bits per PBS, whereas [BFG+25] needs several PBS per plaintext bit, run *sequentially* over the bits.

CKKS-style approximate FHE. We also know of a few works that use the CKKS "approximate" FHE scheme as a foundation for computing with high-precision plaintexts [CCKS23, Kim25, BK25, CPL25]. Before summarizing these, we note that for approximate FHE schemes, obtaining security in the presence of decryptions (of honestly generated ciphertexts) 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 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].

Three other recent works [Kim25, BK25, CPL25] (the last of which is concurrent to and independent of ours) also use CKKS, but in an *exact, discrete* mode that homomorphically does arithmetic on integers (up to some size) [DMPS24]. This mode also supports homomorphic modular reduction [KN25], but this is much more complex and expensive than native operations, because it uses (large-precision) *bootstrapping* as a subroutine; in practice, it has a latency of many seconds. Using sophisticated techniques for representing and operating on large integers, the works [Kim25, BK25, CPL25] support homomorphic arithmetic modulo integers of various forms—including powers of two and large primes—but they require many homomorphic modular reductions, even for addition. So, these schemes are quite complex and have very high latency—e.g., in [BK25, Table 1] one (single-threaded) homomorphic 255-bit multiplication takes 150 seconds.

By contrast, our solution *natively* supports arithmetic modulo any integer, via very simple and fast homomorphic operations—just ordinary polynomial addition and multiplication (with reduction). 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 and also helped inspire our techniques. 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 or get improved error growth, nor do they change the underlying mathematical definitions of ciphertexts and their homomorphic operations.

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 B 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

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

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, i.e., $\|a \cdot b\|_p \leq \|a\|_\infty \cdot \|b\|_p$ 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.

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 Reducing and Decoding

Under the canonical embedding, the ring of integers $R = \mathcal{O}_K$ embeds as a lattice of rank n (over $\mathbb{Z}$), as does any ideal of R; more generally, any additive subgroup of R^d embeds as a rank-nd lattice. In this work, we need to *reduce* elements of K to "small" distinguished representatives modulo certain ideals $\mathfrak{a}$ of R; more generally, we need to reduce elements of K^d modulo certain subgroups (lattices) $\mathcal{L} \subseteq R^d$. We also need to *decode* R under "tiny" error in K, i.e., given any $t = e \pmod R$ where $e \in K$ is sufficiently small, we need to efficiently recover e.[8] All 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 modulus lattice.

Reducing an input $\mathbf{a} \in K^d$ to a small representative modulo a subgroup $\mathcal{L} \subseteq R^d$ using a "short" $\mathbb{Z}$-basis (or if applicable, R-basis) of $\mathcal{L}$—or even just a full-rank subset—can be done using the following "round-off" method. Represent $\mathbf{a}$ as a $\mathbb{Q}$-linear (resp., K-linear) combination of the basis elements, then just drop the integer parts of the coefficients—i.e., reduce each coefficient to its unique mod-$\mathbb{Z}$ representative in $[-\frac{1}{2}, \frac{1}{2})$ (resp., to its small mod-R representative). The result is clearly congruent to $\mathbf{a}$ modulo $\mathcal{L}$, and by the triangle inequality, its canonical Euclidean or max norm is bounded by half the sum of the norms of the $\mathbb{Z}$-basis elements (and similarly for an R-basis). Somewhat tighter bounds, or distributional properties, can be obtained using special features of the basis and/or more sophisticated reductions, like the nearest-plane algorithm [Bab85] or a randomized variant as in [GPV08].

[8] Actually, to conform with the definition and recommended usage of Ring-LWE in general number fields [LPR10, LPR13], we would need to decode 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.

Decoding $t = e \pmod R$ can be done in the same way, by reducing t (as described above) using a suitably "good" $\mathbb{Z}$-basis $\{b_j\}$ of R to get a representative $e' \in K$. 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 $\langle \overline{\boldsymbol{\sigma}(e)}, \boldsymbol{\sigma}(b_j^\vee) \rangle$, where $\{b_j^\vee\} \subset K$ is the *dual* of the basis $\{b_j\}$, which satisfies $\langle \overline{\boldsymbol{\sigma}(b_j)}, \boldsymbol{\sigma}(b_{j'}^\vee) \rangle = \delta_{j,j'}$. 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 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 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 bases with these same norms, but only "near" orthogonality, which complicates the geometry somewhat; see [LPR13] for details.

More generally, any ideal of the form $\mathfrak{a} = aR$ for some $a \in R$ (in particular, integer $a \in \mathbb{Z}$) 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 just use small integer scalings of R.

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 and ciphertext modulus from prior works, 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 $\mathbf{s}(\mathbf{s}) \in K/\mathfrak{p}$ 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 their "small" mod-$\mathfrak{p}$ 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.

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 sufficiently high precision in order to support a small enough error e. More specifically, if $\mathbf{c}$ is over $h^{-1}R \subset K$ (modulo $\mathfrak{p}$) for some $h \in \mathbb{Z}$, then e is as well, so the precision h must be large enough to support the desired error size. Also notice that as the error rate grows due to homomorphic operations, the ciphertext can be discretized (or "rounded off") to have a corresponding lesser precision; this is the analog of "modulus switching" in prior treatments of BFV/BGV.

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.[9] In what follows, every occurrence of K is implicitly restricted to some suitable precision, i.e., we actually use $h^{-1}R \subset K$ for some large enough $h \in R$.

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

1. Sample a (decodable mod R) error $e \in K$ from an appropriate (Module-)LWE error distribution.
2. Choose uniformly random $\mathbf{c}_1 \leftarrow (K/\mathfrak{p})^k$.
3. Let $c_0 = \mu + e - \langle \mathbf{c}_1, \mathbf{s} \rangle \in K/\mathfrak{p}$.
4. Output the ciphertext $\mathbf{c} = (c_0, \mathbf{c}_1) \in (K/\mathfrak{p})^{1+k}$.[10]

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 $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in (K/\mathfrak{p})^{k+1}$, where each $\mathbf{a} \leftarrow (K/\mathfrak{p})^k$ is uniform and each $e \in K$ 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 also pseudorandom.) Rescaling the samples by the precision h, this assumption is 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 their respective 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).[11]

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 $\mathbf{c}_\times(S) := \hat{\mathbf{c}}_\times(S) \bmod \mathfrak{p}$.

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

[11] 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 $\mathfrak{p}$.

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 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 seret 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 $\mathbf{s}$.

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 B, which we first briefly summarize.

Fix a "base" $g \in R$ and positive integer precision parameter ℓ, and everywhere restrict K to elements of this precision, i.e., to $g^{-\ell}R \subset K$. Then we can express any $a \in K/\mathfrak{p}$ as a (Laurent) polynomial $\mathbf{a}(G) \in R[G]$ of degree at most ℓ with small coefficients, such that $\mathbf{a}(g^{-1}) = a \pmod{\mathfrak{p}}$. Equivalently, $\langle \mathbf{g}^{-1}(\mathbf{a}), \mathbf{g} \rangle = a$, 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 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 $\mathfrak{p}$:[12]

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

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.[13] 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 "good" 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*"

[12] We note that $\mathcal{L}^{\perp}_{\mathfrak{p}}(b)$ 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.

[13] 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 $K/\mathfrak{p}$.

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 $\boldsymbol{\mu}_\times(B)$. We address this via *reduction*, which preserves the value of the plaintext while decreasing the decomposition's degree and size below some fixed bounds; see Section 4.6.

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 reduction algorithms that output short representatives in R^d. 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 $\mathbf{c}(B, S) = \mathbf{c}_0(B) + \langle \mathbf{c}_1(B), S \rangle$.

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" these coefficients of $\mathbf{c}$ to any mod-$\mathcal{L}$ representatives in $K[B]$ still satisfies Equation (4.2); in particular, they can be reduced to *small* representatives, yielding a small $\hat{\mathbf{c}}(B, S) \in K[B, S]$. (The details of this reduction 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 a typical mod-$\mathcal{L}$ representative (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 size of the plaintext decomposition $\hat{\boldsymbol{\mu}}(B)$ is proportional to the sizes 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 $\mathbf{s}$, and/or the distribution of $\hat{\mathbf{c}}$ arising from reduction.)

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

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 $\hat{\mathbf{c}}(B, \mathbf{s}) \in K[B]$ 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 $\|s_j\|_\infty$. The claim then follows by the triangle inequality. □

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: one that follows generically from BFV but requires a very small error rate, and a second one that eliminates this drawback using a specialized variant of LWE. (Public-key encryption can be done analogously.) We point out that the first method (and its drawback) is closely analogous to what is done in [CCKS23], but the second method appears 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.[14] This is correct by Item 2 of Remark 4.4, and security follows directly from that of the BFV scheme, and the fact that ciphertext reduction 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 Module-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 a variant LWE. The second method addresses the above drawback by directly constructing a dBFV ciphertext of desired degree bound $< d$ in B using a variant LWE distribution with secret $\mathbf{s}$. In what follows (and as in Section 3.3.2), every occurrence of K is implicitly restricted to $h^{-1}R \subset K$ for some large enough precision parameter $h \in R$.

1. Sample error $\mathbf{e}(B) \in K^{<d}[B]$ from an appropriate error distribution (where the coefficients of $\mathbf{e}(B)$ are decodable mod R).
2. Choose uniformly random $\mathbf{c}_1(B) \leftarrow (K^{<d}[B]/\mathcal{L}^{<d})^k$.
3. Let $\mathbf{c}_0(B) = \mu + \mathbf{e}(B) - \langle \mathbf{c}_1(B), \mathbf{s} \rangle \in K^{<d}[B]/\mathcal{L}^{<d}$.
4. Output the (affine linear in S) ciphertext $\mathbf{c}(B, S) = \mathbf{c}_0(B) + \langle \mathbf{c}_1(B), S \rangle \in (K^{<d}[B]/\mathcal{L}^{<d})[S]$.[15]

By inspection, the output ciphertext satisfies the decryption relation (Equation (4.2)) for plaintext μ. It is also immediate that this encryption is CPA secure, assuming the hardness of the following decisional Module-LWE variant: the secret remains over R, but the samples are over the R-module $K^{<d}[B]/\mathcal{L}^{<d}$ (instead of $K/\mathfrak{p}$), and the error distribution over $K^{<d}[B]$ has an error coefficient in *every* monomial of degree $< d$.

We have not found an elementary hardness proof for this LWE variant based on the hardness of ordinary Module-LWE. However, we believe that by adapting the techniques of [Reg05, LPR10, PRS17], it should be possible to give a comparable worst-case hardness theorem for it, where the error coefficients in each monomial are independent, and the worst-case approximation factor is proportional to the inverse error rate relative to $\mathcal{L}^{<d}$, rather than $\mathfrak{p}$. (See below for further discussion.) We leave this for future work.

Comparison of the two methods. The second encryption method provides a better tradeoff between the error rate of ciphertexts—and thus efficiency and/or security—and their homomorphic capacity (before decryption or bootstrapping is needed). In short, this is because concrete security (for a given rank of the secret key) is primarily determined by the error rate *relative to the modulus lattice*, whereas homomorphic capacity is determined by the "absolute" size of the error *relative to* R (because the accumulated error must remain decodable modulo R). The second method brings these two rates much closer together, because the modulus lattice $\mathcal{L}^{<d}$ for $d > 1$ is much "denser" than $\mathfrak{p}$, in a dimension-normalized sense.

More specifically: for the "very sparse" modulus $\mathfrak{p} = pR$ (say), any error that is decodable mod R must have error rate less than $1/p$ (and even smaller to ensure some homomorphic capacity). By comparison, $\mathcal{L}$ has

[14] 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 is closely tied to the suboptimal parameters of this first method.

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

the same determinant as $\mathfrak{p}$ but in a d-factor larger dimension, so its "sparsity" is only about $p^{1/d} \approx b$. So, for error of the same size relative to R (and thus supporting the same homomorphic capacity), the second method uses a larger LWE error rate by about a $p/b = p^{1-1/d}$ factor. Alternatively, for the same LWE error rate and dimension (and thus security level), the second method uses error that is about a $p^{1-1/d}$ factor smaller relative to R. Since dBFV's error growth factor under homomorphic multiplication is proportional to $p^{1/d}$, the second method can support roughly $d-1$ more layers of multiplication than the first method.[16]

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 *small mod-$\mathcal{L}$ representatives* $\hat{\mathbf{c}}(B,S), \hat{\mathbf{c}}'(B,S) \in K[B,S]$.[17]
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 reduce this modulo $\mathcal{L}$, outputting the dBFV ciphertext $\mathbf{c}_\times(B,S) = \hat{\mathbf{c}}_\times(B,S) \bmod \mathcal{L}$.

Note that the degrees in B and S of $\mathbf{c}_\times$ are typically larger than those of $\mathbf{c}, \mathbf{c}'$; we address this 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 output ciphertext $\mathbf{c}_\times(B,S)$ encrypts $\mu \cdot \mu' \in R/\mathfrak{p}$ under $\mathbf{s}$, with error polynomial*

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

[16] 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.

[17] 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.

Proof. Let $\boldsymbol{\mu}(B), \boldsymbol{\mu}'(B) \in R[B]$ be the digit decompositions encrypted by $\hat{\mathbf{c}}, \hat{\mathbf{c}}'$ (respectively) according to Equation (4.3). For the intermediate product $\hat{\mathbf{c}}_\times$,

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

where in the last equality we have used the fact that $\hat{\boldsymbol{\mu}}(B) = \hat{\mathbf{c}}(B,\mathbf{s}) - \mathbf{e}(B)$, 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 enlarges the error polynomials in $\mathbf{c}, \mathbf{c}'$ by factors of $\hat{\mathbf{c}}'(B,\mathbf{s}), \hat{\mathbf{c}}(B,\mathbf{s}) \in K[B]$, respectively. (The additive term $\mathbf{e}(B) \cdot \mathbf{e}'(B)$ is insignificant, since both errors are relatively much smaller.) Recall from Lemma 4.5 that these factors are small, with coefficients whose norms are proportional to the norms of $\mathbf{s}$ and of small mod-$\mathcal{L}$ representatives. For the latter, Section 4.6 shows that for (say) integer modulus p, using integer base b and degree bound d that satisfy $b^d \geq p$ we can get mod-$\mathcal{L}^{<d}$ representatives whose norms are proportional to $d \cdot p^{1/d}$. So, in terms of the plaintext modulus, the error growth in dBFV is 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 the *degree-reduction* step of ciphertext reduction; 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 $\mathbf{c}_\times(B,S)$ using an $(\mathbf{s} \otimes \mathbf{s})$-to-$\mathbf{s}$ hint made up of dBFV ciphertexts. More specifically, the quadratic coefficient $\mathbf{c}_{\times,2}(B) \in K[B]/\mathcal{L}$ is decomposed to a short vector over $R[B]$, 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 that fixes b and $\mathfrak{p}$, as follows. First observe that $\tau(\mathcal{L}) = \mathcal{L}$ (and similarly for $\mathcal{L}^{<d}$) for any such automorphism τ, since $\tau(\mathbf{z}(b) + \mathfrak{p}) = \tau(\mathbf{z})(b) + \mathfrak{p}$ for any $\mathbf{z}(B) \in R[B]$, so $\mathbf{z}(B) \in \mathcal{L}$ if and only if $\tau(\mathbf{z}(B)) \in \mathcal{L}$. Therefore, τ is an R-module automorphism on $K[B]/\mathcal{L}$ (and on $K^{<d}[B]/\mathcal{L}^{<d}$).

Applying such τ to (the coefficients of) a 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 $\mathbf{s}$ (see the decryption relation Equation (4.2)), by the additivity and multiplicativity of τ we get that

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

That is, $\tau(\mathbf{c})(B,S)$ is a dBFV ciphertext the encrypts $\tau(\mu) \in R/\mathfrak{p}$ with error $\tau(\mathbf{e}(B)) \in K[B]$ under secret key $\tau(\mathbf{s})$. (The error $\tau(\mathbf{e}(B))$ is decodable mod R if and only if $\mathbf{e}(B)$ is.) As usual, we can switch the secret key from $\tau(\mathbf{s})$ back to $\mathbf{s}$ (or some other secret key) using key-switching.

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

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

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 $\mathcal{L}' = \mathcal{L}^{\perp}_{\mathfrak{p}}(b')$, and let K-linear function $\phi\colon K[B] \to K[B']$ be such that $\phi(R[B]) \subseteq R[B']$ and $\phi(\mathbf{z}(B))(b') = \mathbf{z}(b) \pmod{\mathfrak{p}}$ for all $\mathbf{z}(B) \in R[B]$, so that $\phi(\mathcal{L}) \subseteq \mathcal{L}'$. 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 hypotheses, ϕ induces an R-module homomorphism from $K[B]/\mathcal{L}$ to $K[B']/\mathcal{L}'$, and a ring isomorphism from $R[B]/\mathcal{L}$ to $R[B']/\mathcal{L}'$ that corresponds to the identity function on the plaintext ring $R/\mathfrak{p}$ when composed with the latters' evaluation-at-b and at-b' (respectively) ring isomorphisms with $R/\mathfrak{p}$. By all this, the fact that the secret key $\mathbf{s}$ is over R, and the decryption relation (Equation (4.2)),

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

which proves the claim. □

1. The above warm-up example of $b = b'^c$ is formalized by defining ϕ via the variable substitution $B = B'^c$. Then $\phi(\mathbf{c}(B))(b') = \mathbf{c}(b) \in K$ for all $\mathbf{c}(B) \in K[B]$, so the hypotheses of Lemma 4.8 hold.

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 $K[B']$. Then $\phi(\mathbf{c})(b') = \mathbf{c}(b) \in K$ for all $\mathbf{c}(B) \in K[B]$, so the hypotheses of Lemma 4.8 hold, and 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', very similarly to what is done in general degree reduction (see Section 4.6.2), we can define ϕ by expressing each relevant power $b^i \bmod \mathfrak{p}$ as a "small" R-linear combination of sufficiently many powers $b'^j \bmod \mathfrak{p}$ for $j \geq 0$. That is, we express $\mathbf{b}^t = (\mathbf{b}')^t \cdot \mathbf{D} \pmod{\mathfrak{p}}$ where column vectors $\mathbf{b}, \mathbf{b}'$ consist of the relevant powers of b, b' (respectively), and matrix $\mathbf{D}$ over R is small (e.g., consists of small mod-$b'R$ representatives). Then we define $\phi(\mathbf{c}) := \mathbf{D}\mathbf{c}$ for any $\mathbf{c} \in K[B]$, where we identify polynomials with their coefficient (column) vectors.

   Since $\phi(\mathbf{z}(B))(b') = (\mathbf{b}')^t \cdot \mathbf{D} \cdot \mathbf{z} = \mathbf{b}^t \cdot \mathbf{z} = \mathbf{z}(b) \pmod{\mathfrak{p}}$ for any $\mathbf{z} \in R[B]$, this ϕ satisfies the hypotheses of Lemma 4.8. The error polynomial is enlarged by the small matrix $\mathbf{D}$, i.e., the error in the transformed ciphertext is $\mathbf{e}' = \mathbf{D} \cdot \mathbf{e}$.

4.6 Ciphertext 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 fairly small representatives in $K[B]$; and that the intermediate product ciphertext typically has larger degree in B. So, we need ways to efficiently perform the lifting, and to reduce the degree in B below some fixed bound; we call the combination of these operations *ciphertext reduction*.

To do the lifting, we simply use standard lattice algorithms (Section 2.3) to reduce the ciphertext coefficients in $K[B]/\mathcal{L}$ to small mod-$\mathcal{L}$ representatives, using a "short" R-basis (or full-rank subset) of the lattice $\mathcal{L}$ (restricted to suitable degree).[18] 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.

[18] 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 lattice-reduction algorithms we describe here are also essentially the same as those considered in [MP12], though our degree-reduction and quasi-linear-time "hybrid" reduction algorithms are novel.

Lemma 4.9. *Let $\phi\colon K[B] \to K^{<d}[B]$ be a K-linear degree-reduction map that is identity on $K^{<d}[B]$ and satisfies $\phi(\mathcal{L}) \subseteq \mathcal{L}^{<d}$. 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),

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

This proves the claim. □

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, and ciphertext reduction yields particularly short outputs and does not increase the error (indeed, it can actually decrease it). We start with this special case first in Section 4.6.1, then handle the general case of arbitrary $\mathfrak{p}$ and $\mathfrak{b}$ in Section 4.6.2.

4.6.1 Simple 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 a particularly simple ciphertext-reduction algorithm with no error growth and especially short outputs, whose coefficients are distinguished representatives of $K/\mathfrak{b}$.

Degree reduction map. We define the K-linear degree-reduction map $\phi\colon K[B] \to K^{<d}[B]$ to simply discard any monomial of degree $\geq d$. This is clearly the identity map on $K^{<d}[B]$, and $\phi(\mathcal{L}) = \mathcal{L}^{<d}$ because $b^j = 0 \pmod{\mathfrak{p}}$ for every $j \geq d$. 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).

Reduction modulo $\mathcal{L}^{<d}$. 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 $i = 0, \ldots, d-2$. 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 $b\mathbf{I}_d$. This yields an especially simple instantiation of the nearest-plane algorithm for reducing modulo $\mathcal{L}^{<d}$, as follows.

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 $i = d-1$, this step is skipped.)

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 mod-$\mathfrak{b}$ representatives. Iteration i of the loop has the effect of subtracting by $q_i \cdot (b - B) \cdot B^i \in \mathcal{L}$ (or by $q_i \cdot b \cdot B^{d-1} \in \mathcal{L}$, for $i = d-1$), so the final output is congruent modulo $\mathcal{L}$ to the input, as needed.

4.6.2 General Reduction for any $\mathfrak{p}, \mathfrak{b}$.

In general, we can construct a degree-reduction projection, and a basis of the corresponding degree-bounded lattice $\mathcal{L}$, as follows. First, let d be a sufficiently large integer so that:

- any $\mathbf{f}(B) \in R[B]/\mathcal{L} \cong R/\mathfrak{p}$ has an efficiently computable "small" representative $\mathbf{r}_{\mathbf{f}}(B) \in R^{<d}[B]$, i.e., $\mathbf{r}_{\mathbf{f}}(b) = \mathbf{f}(b) \pmod{p}$, and
- for the representative $\mathbf{r}_d(B) \in R^{<d}[B]$ of $B^d \bmod \mathcal{L}$, we have that $\mathbf{r}_d(b) \neq b^d$ (over R, not modulo anything).

As a primary example, if b, p are positive integers and $\mathfrak{p} = pR$, then we can let $d = \lceil \log_b p \rceil$: the small representative of $\mathbf{f}(B) \in R[B]/\mathcal{L}$ is the polynomial $\mathbf{r}(B) = \sum_{i<d} r_i \cdot B^i \in R^{<d}[B]$ whose coefficients correspond to the base-b representation of the distinguished representative of $\mathbf{f}(b) \in R/pR$; moreover, $b^d \geq p$, so b^d is not equal to its mod-pR representative. (This approach generalizes naturally to any modulus ideal $\mathfrak{p}$ for which we can efficiently compute suitably small representatives of $R/\mathfrak{p}$.)

Degree reduction. We define the K-linear degree-reduction map $\phi\colon K[B] \to K^{<d}[B]$ as follows. For each $j \geq d$, let $\mathbf{r}_j(B) \in R^{<d}[B]$ be a small representative of $B^j \bmod \mathcal{L}$, as above. Then ϕ is the K-linear map defined by replacing B^j with $\mathbf{r}_j(B)$ for each $j \geq d$. Because $\mathbf{r}_j(B) = B^j \pmod{\mathcal{L}}$ for all $j \geq d$, we have that $\phi(\mathcal{L}) = \mathcal{L}^{<d}$, so ϕ satisfies the hypotheses of Lemma 4.9.

We can view this 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

$$\mathbf{P}_d = \begin{pmatrix} \mathbf{I}_d & \mathbf{r}_d & \mathbf{r}_{d+1} & \cdots \end{pmatrix},$$

where we truncate the number of columns to match the degree bound of the input polynomial. (For our purposes in dBFV, degree bound $< 2d$ suffices.) This typically enlarges the error when applied to a ciphertext, but only by a fairly small factor since it has bounded degree and the entries of $\mathbf{P}_d$ are small, e.g., mod-$\mathfrak{b}$ representatives.

Reduction modulo $\mathcal{L}^{<d}$. We next address reduction modulo the lattice $\mathcal{L}^{<d}$ by constructing a short full-rank set of lattice vectors.

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

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

$$\mathbf{R} = \left(\begin{array}{cccc|c} b & & & & -r_0 \\ -1 & b & & & -r_1 \\ & -1 & \ddots & & \\ & & & b & -r_{d-2} \\ & & & -1 & b - r_{d-1} \end{array}\right) \in R^{d \times d}. \tag{4.4}$$

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 reduce modulo $\mathcal{L}^{<d}$ to get proportionately small representatives. However, neither option is especially attractive for practice, because the inverse and Gram–Schmidt orthogonalization of the matrix $\mathbf{R}$ from Equation (4.4) require high precision, and do not have such nice forms, so straightforward implementations run in time at least quadratic in d.

Specialized hybrid reduction. Here we describe a practical "hybrid" of the two reduction algorithms, specialized to the matrix $\mathbf{R}$ from Equation (4.4), which runs in time linear in d and uses 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.

In order to properly define and analyze this hybrid algorithm, which relies on notions of orthogonality and orthogonal projections, we define K^d as a "generalized" inner-product space over K. The usual definition of inner product requires a vector space over $\mathbb{R}$ or $\mathbb{C}$ in order to satisfy positive definiteness; we generalize this to allow base field K by considering all its complex embeddings. 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 define the inner product on K^d as $\langle \mathbf{u}, \mathbf{v} \rangle := \sum_i \overline{u_i} \cdot v_i \in K$. This has conjugate symmetry and is K-linear in the second argument, and thus is *conjugate* K-linear in the first argument, as one expects from a complex inner product. For positive definiteness, $\sigma(\bar{u} \cdot u) \geq 0$ for every $u \in K$ and embedding $\sigma\colon K \to \mathbb{C}$, with equality if and only if $u = 0$, so this inner product is positive definite with respect to every complex embedding of K.

Let $\mathbf{B} \in R^{d \times (d-1)}$ be the matrix consisting of the first $d-1$ columns of $\mathbf{R}$, and let $B = \mathrm{span}_K(\mathbf{B}) \subseteq K^d$. 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 the orthogonal subspace $B^\perp$, 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 $\mathbf{D}^* \cdot \mathbf{B} = \mathbf{I}_{d-1}$. Letting $s = \bar{b}b$, one may verify by inspection that

$$(s^d - 1) \cdot \mathbf{D}^* = \begin{pmatrix} \bar{b}^1(s^{d-1}-1) & & & \\ & \bar{b}^2(s^{d-2}-1) & & \\ & & \ddots & \\ & & & \bar{b}^{d-1}(s^1-1) \end{pmatrix} \begin{pmatrix} b^0 & & & & \\ b^0 & b^1 & & & \\ \vdots & \vdots & \ddots & & \\ b^0 & b^1 & \cdots & b^{d-2} & 0 \end{pmatrix} - \tag{4.5}$$
$$\begin{pmatrix} b^{-1}(s^1-1) & & & \\ & b^{-2}(s^2-1) & & \\ & & \ddots & \\ & & & b^{-(d-1)}(s^{d-1}-1) \end{pmatrix} \begin{pmatrix} 0 & b^1 & b^2 & \cdots & b^{d-1} \\ & & b^2 & \cdots & b^{d-1} \\ & & & \ddots & \vdots \\ & & & & b^{d-1} \end{pmatrix}.$$

Notice that the above lower- and upper-triangular matrices correspond to successive "partial" evaluations at b of a polynomial's lower- and higher-degree terms, respectively. As we will see next, this yields an efficient algorithm for multiplication by $\mathbf{D}^*$.

The hybrid algorithm reduces an input $\mathbf{v} \in K^d$ (representing a polynomial $\mathbf{v}(B) \in K^{<d}[B]$) to a small mod-$\mathcal{L}^{<d}$ 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 mod-gR representative (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 (4.5), use the following loop, initialized with $u' = 0, v' = \mathbf{v}'(b) \in K$. For $i = 1, \ldots, d-1$:

   (a) Let $u' = u' + v'_{i-1} \cdot b^{i-1}$ and $v' = v' - v'_{i-1} \cdot b^{i-1}$. (These are the "partial" evaluations of $\mathbf{v}'$ at b.)

   (b) Let $y_i = \bar{b}^i(s^{d-i} - 1) \cdot u' - b^{-i}(s^i - 1) \cdot v' \in K$, let $w_i \in K$ be its distinguished representative modulo $(s^d - 1)R$, and let $z_i = (y_i - w_i)/(s^d - 1) \in R$.[19]

Lemma 4.11. *The above hybrid algorithm outputs a mod-$\mathcal{L}^{<d}$ representative of the input in the parallelepiped $(\mathbf{B} \mid \tilde{\mathbf{r}}) \cdot \mathcal{P}(R)^d$, where $\tilde{\mathbf{r}}$ is the orthogonal projection of $\mathbf{r}$ onto $B^\perp = \mathrm{span}_K(\mathbf{B})^\perp = \mathrm{span}_K(\mathbf{b})$, and $\mathcal{P}(R)$ is the fundamental parallelepiped of the employed short $\mathbb{Z}$-basis of R.*

Before giving the proof, we analyze the size of $\tilde{\mathbf{r}}$, and thereby the size of the output. Letting $g = b^d - \mathbf{r}(b)$,

$$\|\tilde{\mathbf{r}}\|^2 = \frac{\overline{\langle \mathbf{b}, \mathbf{r} \rangle} \cdot \langle \mathbf{b}, \mathbf{r} \rangle}{\langle \mathbf{b}, \mathbf{b} \rangle} = \frac{\bar{g} \cdot g \cdot (s-1)}{(\bar{b} \cdot b)^d - 1} \in K \, .$$

In particular, if $b, p \in \mathbb{Z}$ and we set d such that $b^d - \mathbf{r}(b) = p$ for $\mathbf{r}(b) > 0$, we have that $\|\tilde{\mathbf{r}}\|^2 = p^2 \cdot (b^2 - 1)/(b^{2d} - 1) \leq b^2 - 1$. So, $\tilde{\mathbf{r}}$ is shorter than the columns of $\mathbf{B}$, and hence the algorithm's output has size proportional to b.

Proof. First, the output $\mathbf{v}'' = \mathbf{v} \pmod{\mathcal{L}^{<d}}$, 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 $\mathcal{L}^{<d}$. The second part of the claim is equivalent to showing that all the entries of $\mathbf{D}^* \cdot \mathbf{v}'' \in K^{d-1}$, and $\langle \tilde{\mathbf{r}}, \mathbf{v}'' \rangle / \langle \tilde{\mathbf{r}}, \tilde{\mathbf{r}} \rangle = \langle \mathbf{b}, \mathbf{v}'' \rangle / \langle \mathbf{b}, \mathbf{r} \rangle \in K$, are in $\mathcal{P}(R)$. For the latter, because $\mathbf{b}^* \cdot \mathbf{B} = \mathbf{0}$ and by definition of c,

$$\mathbf{b}^* \cdot \mathbf{v}'' = \mathbf{b}^* \cdot \mathbf{v}' = \mathbf{b}^*(\mathbf{v} - z \cdot \mathbf{r}) = (c - z) \cdot \mathbf{b}^* \cdot \mathbf{r} \, ,$$

and $c - z \in \mathcal{P}(R)$ by definition of $z = \lfloor c \rceil$. For the former, by definition of $\mathbf{D}^*$ we have that $\mathbf{D}^* \cdot \mathbf{v}'' = \mathbf{D}^* \cdot \mathbf{v}' - \mathbf{z}$, all of whose entries are in $\mathcal{P}(R)$ by definition of $\mathbf{z} = \lfloor \mathbf{D}^* \cdot \mathbf{v}' \rceil$. □

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

[19] 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.

(high-error) ciphertext on a (low-error) *encryption of the secret key*, yielding a (fairly low-error) encryption of the same plaintext.

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 "denser" ideal), because in the final step their linear combination needs to yield the original mod-$\mathfrak{p}$ plaintext.[20] 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*

[20] 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 "denser" plaintext modulus can yield somewhat better bootstrapping efficiency and error growth for such digits, but it does not help for $i = 0$.

variant of GSW (dGSW) for the bootstrapping key and bootstrapping process. See Section 5.3 for details of the bootstrapping, and Appendix A for a description of the dGSW scheme.

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 the "rounding" function from $K/\mathfrak{p}$ (with some suitable precision) to $R/\mathfrak{p}$, via some carefully designed efficient arithmetic circuit. Analogously, the dBFV rounding function from $K^{<d}[B]/\mathcal{L}^{<d}$ 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 A—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."[21] 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 $p^{1/d}$ for any desired $d \geq 1$. The details of dGSW are given in Appendix A, 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.[22] 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:

[21] 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.

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

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
$$c_{0,0} + \langle \vec{c}, \vec{s} \rangle = m \cdot e_0 \pmod{m}, \tag{5.2}$$
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 $\tilde{e}(X) \in K[X]$ has a constant coefficient of e_0, and $\approx$ hides a tiny error.[23] 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 $\hat{z}(X) = e(X) - \tilde{e}(X) \in K[X]$ 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 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, 3, 8, and 32.

[AP14] J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, pages 297–314. 2014. Pages 1, 8, and 40.

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

[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 13 and 29.

[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 11.

[BFG+25] Z. Brakerski, O. Friedman, D. Golan, A. Gurny, D. Mutzari, and O. Sheinfeld. REFHE: Fully homomorphic ALU. Cryptology ePrint Archive, Paper 2025/1449, 2025. Pages 2, 3, 10, 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. 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, and 12.

[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, and 11.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, pages 868–886. 2012. Pages 1, 2, 4, 12, and 14.

[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, 6, and 17.

[BV14] Z. Brakerski and V. Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12. 2014. Pages 8 and 40.

[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, 6, 8, 11, and 21.

[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, 6, 8, 31, 32, 34, and 40.

[CH18] H. Chen and K. Han. Homomorphic lower digits removal and improved FHE bootstrapping. In *EUROCRYPT*, pages 315–337. 2018. Pages 8 and 32.

[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, and 6.

[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, 10, and 14.

[CPL25] G. Cha, D. Park, and J.-W. Lee. Improved radix-based approximate homomorphic encryption for large integers via lightweight bootstrapped digit carry. Cryptology ePrint Archive, Paper 2025/1740, 2025. https://eprint.iacr.org/2025/1740. Pages 2, 3, and 11.

[DM15] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, pages 617–640. 2015. Pages 1, 2, 8, 31, 32, 34, and 40.

[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 11.

[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, 12, and 14.

[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, 10, and 14.

[Gen09a] C. Gentry. *A fully homomorphic encryption scheme*. Ph.D. thesis, Stanford University, 2009. http://crypto.stanford.edu/craig. Pages 1 and 30.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009. Pages 1 and 30.

[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, 8, and 32.

[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 8 and 32.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008. Page 13.

[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, 31, and 38.

[GV23] R. Geelen and F. Vercauteren. Bootstrapping for BGV and BFV revisited. *J. Cryptol.*, 36(2):12, 2023. Pages 8 and 32.

[GV25a] R. Geelen and F. Vercauteren. Better GBFV bootstrapping and faster encrypted edit distance computation. Cryptology ePrint Archive, Paper 2025/1104, 2025. Pages 8, 10, and 32.

[GV25b] R. Geelen and F. Vercauteren. Fully homomorphic encryption for cyclotomic prime moduli. In *EUROCRYPT*, pages 366–397. 2025. Pages 2, 3, 4, 10, and 14.

[HPS98] J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288. 1998. Page 10.

[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 10.

[HS15] S. Halevi and V. Shoup. Bootstrapping for HElib. In *EUROCRYPT*, pages 641–670. 2015. Pages 1, 8, and 32.

[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 8, 31, 32, and 33.

[Kim25] 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 11.

[KLSS23] M. Kim, D. Lee, J. Seo, and Y. Song. Accelerating HE operations from key decomposition technique. pages 70–92. 2023. Page 11.

[KN25] J. Kim and T. Noh. Modular reduction in CKKS. *IACR Commun. Cryptol.*, 2(2):17, 2025. Page 11.

[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 8 and 32.

[LM21] B. Li and D. Micciancio. On the security of homomorphic encryption on approximate numbers. pages 648–677. 2021. Pages 3 and 11.

[LMSS22] B. Li, D. Micciancio, M. Schultz, and J. Sorrell. Securing approximate homomorphic encryption using differential privacy. pages 560–589. 2022. Pages 3 and 11.

[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, 12, 13, and 22.

[LPR13] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT*, pages 35–54. 2013. Pages 12, 13, and 14.

[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 12.

[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 8 and 32.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. 2012. Pages 6, 8, 19, 20, and 26.

[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. pages 69–100. 2023. Pages 8 and 32.

[PRS17] C. Peikert, O. Regev, and N. Stephens-Davidowitz. Pseudorandomness of Ring-LWE for any ring and modulus. In *STOC*, pages 461–473. 2017. Page 22.

[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, 12, and 22.

[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 11.

A 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, and implicitly restrict elements of K to some suitable precision, i.e., use $h^{-1}R \subset K$ for a sufficiently large integer h. 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 $a \in K[B]/\mathcal{L}$ (of the given precision) 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 vector or matrix $\mathbf{M}$ over $R[B]$ that can be multiplied on the left with $\mathbf{A}$.

Definition A.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

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

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 A.2 below.

A.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 A.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 the Module-LWE variant over $K^{<d}[B]/\mathcal{L}^{<d}$ with secret $\bar{\mathbf{s}} \in R^k$. As usual, in what follows, every occurrence of K is actually restricted to $h^{-1}R \subset K$ for a suitable precision parameter $h \in R$.

1. Sample error vector $\mathbf{e}$ over $K^{<d}[B]$ 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 $(K^{<d}[B]/\mathcal{L}^{<d})^k$.

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

$$\mathbf{C} = \begin{pmatrix} \mathbf{A} \\ \mathbf{b} + \bar{\mu} \cdot (\mathbf{s} \otimes \mathbf{g}) \end{pmatrix}.$$

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

By inspection, the output ciphertext satisfies Definition A.1. It is also immediate that this encryption is CPA secure assuming the hardness of the decisional Module-LWE variant introduced in Section 4.3 (which says that each sampled $(\mathbf{A}, \mathbf{b})$ is pseudorandom).

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.

A.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 A.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 A.2. We verify the decryption relation (Equation (A.1)) using the hypotheses:

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

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

B 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-precisions 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 polynomials over R, and elements of R are typically represented as polynomials over $\mathbb{Z}$ (with some polynomial modulus), ciphertexts are ultimately represented as multi-variate 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.

B.1 Digit Representation

Fix a base $g \in R$ (typically, an integer greater than 1) 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 B.1.1 for discussion).

Definition B.1 (Digit representation). A *digit representation* of an element $a \in g^{-\ell}R \subset K$ is a *Laurent* polynomial[24]

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

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 B.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

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

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), it suffices to work with representatives of K/g^eR for a suitable positive integer e, namely, one for which $g^e \in \mathfrak{p}$. 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."

B.1.1 Reduction

Any Laurent polynomial $\mathbf{a}(G)$ over R that satisfies Equation (B.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 B.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.

The reduced representation of any element in $g^{-\ell}R$ is unique (up to padding by zero digits, in an implementation). Therefore, we can generate a uniformly random representative of $g^{-\ell}R/g^kR$ simply by concatenating $\ell + k$ uniformly random and independent digits (distinguished representatives of R/gR). Also observe that a reduced element can be "rounded off" to the nearest 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.

As we will see below, the basic addition and multiplication operations on field elements typically yield non-reduced representations. Fortunately, any valid representation can easily be converted to reduced form, essentially by propagating "carries" (or "overflows") to more-significant digits. More specifically, to reduce $\mathbf{a}(G)$ we do the following for $i = \ell, \ell - 1, \ldots$ (down to $-e$, if $\mathbf{a}(G)$ stands for a representative of K/g^eR):

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 $j < i$, break out of the loop. (This is a non-constant-time optimization.)

Finally, output $\mathbf{a}'(G) = \sum_i a'_i \cdot G^i$ (where the sum is over $i > -e$, if working with representatives of K/g^eR). It is clear by inspection that $\mathbf{a}'(G)$ is reduced, and that $\mathbf{a}'(g^{-1}) = \mathbf{a}(g^{-1})$, so the output and input and digit representations of the same element of K.

B.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 (lifted) dBFV ciphertext 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, by keeping the coefficients small enough and using a large enough prime $q = 1 \pmod{2n}$ (or a product of multiple such primes), we can use the analogous fast Number Theoretic Transform (NTT) over the field $\mathbb{Z}_q$, which has primitive $2n$th roots of unity by construction. We stress that q needs to be large enough so that the integer coefficients are uniquely determined by their mod-q reductions, i.e., they do not "wrap around" modulo q.

3. Since the degree in G is moderate, 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).

4. 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.

5. Alternatively for S, because any multiplied polynomials are merely linear in S, we can use Karatsuba's trick to do this using just three multiplications of polynomials in X,G,B.