Fully Homomorphic Encryption for Cyclotomic Prime Moduli

Robin Geelen [ID] and Frederik Vercauteren [ID]

COSIC, ESAT, KU Leuven, Belgium
`firstname.lastname@esat.kuleuven.be`

Abstract. This paper presents a Generalized BFV (GBFV) fully homomorphic encryption scheme that encrypts plaintext spaces of the form $\mathbb{Z}[x]/(\Phi_m(x), t(x))$ with $\Phi_m(x)$ the m-th cyclotomic polynomial and $t(x)$ an arbitrary polynomial. GBFV encompasses both BFV where $t(x) = p$ is a constant, and the CLPX scheme (CT-RSA 2018) where $m = 2^k$ and $t(x) = x - b$ is a linear polynomial. The latter can encrypt a single huge integer modulo $\Phi_m(b)$, has much lower noise growth than BFV, but it is not known to be efficiently bootstrappable.

We show that by a clever choice of m and higher degree polynomial $t(x)$, our scheme combines the SIMD capabilities of BFV with the low noise growth of CLPX, whilst still being efficiently bootstrappable. Moreover, we present parameter families that natively accommodate packed plaintext spaces defined by a large cyclotomic prime, such as the Fermat prime $\Phi_2(2^{16}) = 2^{16}+1$ and the Goldilocks prime $\Phi_6(2^{32}) = 2^{64}-2^{32}+1$. These primes are often used in homomorphic encryption applications and zero-knowledge proof systems.

Due to the lower noise growth, GBFV can evaluate much deeper circuits compared to native BFV in the same ring dimension. As a result, we can evaluate either larger circuits or work with smaller ring dimensions. In particular, we can natively bootstrap GBFV at 128-bit security already at ring dimension $n = 2^{14}$, which was impossible before. We implemented the GBFV scheme on top of the SEAL library and achieve a latency of only 2 seconds to bootstrap a ciphertext encrypting up to 8192 elements modulo $2^{16} + 1$.

Keywords: Fully homomorphic encryption · Bootstrapping · GBFV · BFV · CLPX · Cyclotomic prime · Fermat prime · Goldilocks prime.

1 Introduction

Homomorphic encryption (HE) schemes are commonly divided into two categories: on the one hand, there exist schemes that can evaluate "single instruction, multiple data" operations on a batch encryption of multiple elements (a.k.a. SIMD schemes). Examples of this first category are BGV [12], BFV [11,23] and CKKS [17]. On the other hand, some schemes do not have the SIMD option, but have faster execution times, an easier programming model and smaller parameters. This second category includes FHEW [22] and TFHE [18].

All previously mentioned homomorphic encryption schemes are noise-based. Encryption adds a small "noise" or "error" to the ciphertext, and this noise is removed during decryption. This approach is necessary for security, but it also comes with a major limitation: homomorphic evaluation of a circuit causes the noise to grow. As such, the noise must stay below a given threshold for the ciphertext to remain decryptable.

For current SIMD schemes, the noise-based methodology imposes one more restriction: let p be the "precision" of the encoding (i.e. the plaintext modulus in BGV/BFV), then the noise growth of multiplication obeys

$$n_{\text{out}} \propto p \cdot (n_1 + n_2),$$

where n_1 and n_2 are upper bounds on the input noise, and n_{out} is an upper bound on the output noise. Observe that n_{out} is directly proportional to p. This linear relation is an unfavorable property that results in more noise for larger p. As such, current SIMD schemes are rather impractical for high-precision arithmetic, which is required in many useful HE applications (see Section 1.2).

A less well-studied scheme, which does not belong to either of the categories discussed above, is the CLPX scheme due to Chen et al. [15]. The idea is to define the plaintext ring modulo a linear polynomial $t(x) = x - b$, instead of an integer p as in BGV and BFV. As such, it can encode a single integer defined modulo $\Phi_m(b)$ (which typically supports thousands of bits), but has relatively slow execution time and large parameters. Unfortunately, this scheme is still rather impractical as it supports only one number per ciphertext and is only a leveled scheme because no efficient bootstrapping method is known.[1] However, on the positive side, the noise growth under multiplication is only sublinear in the desired precision: it depends on b rather than $\Phi_m(b)$. Whereas BGV and BFV are very limited in multiplicative capacity, the CLPX scheme creates a true "gap" between precision and noise growth. Consequently, the CLPX scheme is currently the best choice for implementing extremely high-precision exact arithmetic in homomorphic encryption.

1.1   Contributions

It is an open problem to design an HE scheme that supports both high-precision arithmetic and SIMD capabilities. We solve this problem by generalizing and simplifying BFV and CLPX to arbitrary cyclotomic polynomials $\Phi_m(x)$ and arbitrary plaintext polynomials $t(x)$. We also give a detailed noise analysis of the various homomorphic operations.

We then instantiate the scheme by a clever choice of m and $t(x)$, allowing us to compute with vectors of elements in finite fields defined by a cyclotomic prime, i.e. a prime obtained as the evaluation of a cyclotomic polynomial in an integer. We also show how to deal with extensions of such finite fields. We give

---

[1] The scheme may be bootstrapped via reduction to binary circuits, but we conjecture this to be rather inefficient.

several parameter families including the Fermat prime $\Phi_2(2^{16}) = 2^{16} + 1$ and the Goldilocks prime $\Phi_6(2^{32}) = 2^{64} - 2^{32} + 1$ (both of which are examples of so-called generalized Mersenne primes).

Our construction is a trade-off between standard BFV and CLPX: similar to BFV, our scheme offers packing capabilities, though with somewhat fewer slots; and similar to CLPX, our scheme encrypts large (but not huge) integers with sublinear noise growth. We call the new scheme Generalized BFV (GBFV).

Finally, we show for the first time how an encryption scheme with polynomial (i.e. non-scalar) plaintext modulus can be bootstrapped for appropriate parameters. This is based on novel GBFV-to-BFV conversion and packing algorithms, which may be of independent interest. We implement our bootstrapping on top of Microsoft SEAL [58] and compare it to regular BFV bootstrapping.[2]

1.2   Motivation

Various FHE applications require high-precision plaintext spaces. For example, state-of-the-art private set intersection protocols [19] use a plaintext modulus of 16 up to 26 bits, which is already significant in terms of noise growth for standard BFV. Privacy preserving machine learning [33] uses even larger moduli of up to 80 bits. Other applications of high-precision FHE include rational number encoding [20,15] and p-adic encoding [37,5].

Outside the FHE domain, many zero-knowledge proof systems also use large values of p [39,6]. For example, the FRI-based systems known as Plonky2 [55], Miden-VM [54], Era-Boojum [51] and Risc Zero [57] use the popular Goldilocks prime $p = 2^{64} - 2^{32} + 1$.[3] A follow-up work shows how GBFV instantiated with the Goldilocks prime can securely and efficiently delegate proof generation of a zkSNARK to an untrusted server [24].

Another reason to use large values of p is packing density. It is well known that BFV's packing density (i.e. the number of slots divided by the ring dimension) is equal to $1/d$, where d is the multiplicative order of p modulo the cyclotomic index m. As such, we need $p > m$ for full packing. And in the specific case of power-of-two cyclotomics, the number of slots is upper bounded by $(p+1)/2$ [25]. This is one of the motivations to use the popular prime $p = 2^{16} + 1$ [10], which achieves full packing density up to index $m = 2^{16}$. The large-p restriction is even more apparent during bootstrapping, where p^2 is used as an intermediate modulus. This results in a precision of 32 bits for the 16-bit Fermat prime.

1.3   Related Work

The idea behind the CLPX scheme originates from the NTRU scheme. Hoffstein and Silverman [38] noticed that the integer modulus in NTRU encoding can be replaced with a small polynomial modulus. The CLPX scheme uses this trick in

---

[2] See https://github.com/KULeuven-COSIC/Bootstrapping_BGV_BFV/tree/traces.

[3] Note that the name "Goldilocks prime" is a slight abuse of terminology here, because the original prime was of the shape $\varphi^2 - \varphi - 1$ rather than $\varphi^2 - \varphi + 1$ [36,7].

combination with the BFV scheme to construct leveled homomorphic encryption for large integers [15]. Later research has shown how the same trick can be used to encode complex numbers more efficiently [8,14]. To some extent, these works on complex number encoding already generalize CLPX to moduli of the shape $x^k - b$, but this is still not general enough for our use case. Moreover, these prior works are tailored to complex numbers, do not have a mechanism to permute the encoded plaintext slots, and are not known to be bootstrappable. Another work constructs an alternative way to reduce the modulus consumption in complex number encoding by making the individual FHE operations more expensive [16]. However, this strategy seems inapplicable to exact schemes.

2   Preliminaries

2.1   Cyclotomic Fields and Rings

We will use the R-LWE problem, so we first introduce definitions and properties of cyclotomic polynomials. For an integer $m \geqslant 1$, we take a primitive m-th root of unity $\omega_m \in \mathbb{C}$. This means that $\omega_m^k = 1$ if and only if m divides k. We call

$$\Phi_m(x) = \prod_{j \in \mathbb{Z}_m^\times} (x - \omega_m^j)$$

the m-th *cyclotomic polynomial*. Here we used $\mathbb{Z}_m^\times$ for the unit group of integers modulo m. The degree of the above polynomial is $n = \varphi(m)$, where $\varphi(\cdot)$ is Euler's totient function. A standard result states that all cyclotomic polynomials are monic, irreducible over $\mathbb{Q}$ and have integer coefficients [1]. For the R-LWE problem, we define the m-th cyclotomic number field $\mathcal{K} = \mathbb{Q}(\omega_m) = \mathbb{Q}[x]/(\Phi_m(x))$ and its ring of integers $\mathcal{R} = \mathbb{Z}[\omega_m] = \mathbb{Z}[x]/(\Phi_m(x))$. The Galois group of $\mathcal{K}/\mathbb{Q}$ is written as $\mathrm{Gal}(\mathcal{K}/\mathbb{Q})$. It consists of the automorphisms $\sigma_j \colon x \mapsto x^j$ for $j \in \mathbb{Z}_m^\times$. As such, it is a trivial result that this Galois group is isomorphic to $\mathbb{Z}_m^\times$. The multiplicative subgroup generated by $g_1, \ldots, g_s \in \mathbb{Z}_m^\times$ is denoted by $\langle g_1, \ldots, g_s \rangle$. An ideal in a ring is written with round parentheses, that is $(r_1, \ldots, r_s)$.

Embeddings and norms. For the purpose of noise analysis, we need to embed the cyclotomic number field into a real or complex vector space. Two common methods are the coefficient embedding and the canonical embedding. In our definition, the coefficient embedding uses the powerful basis of $\mathcal{K}$ [47].

Definition 1. *Let $m = m_1 \cdot \ldots \cdot m_s$ be the prime-power factorization of m. Let*

$$\boldsymbol{a} = \sum_{(i_1, \ldots, i_s) \in I} a_{i_1, \ldots, i_s} \cdot x_1^{i_1} \cdot \ldots \cdot x_s^{i_s},$$

where $x_j = x^{m/m_j}$ and I is the set of s-tuples with the j-th entry ranging from 0 to $\varphi(m_j) - 1$. Then the coefficient embedding *is defined by the map*

$$\iota \colon \mathcal{K} \hookrightarrow \mathbb{R}^n \colon \boldsymbol{a} \mapsto \{a_{i_1, \ldots, i_s}\}_{(i_1, \ldots, i_s) \in I}.$$

Definition 2. *The* canonical embedding *is defined by the map*

$$\tau \colon \mathcal{K} \hookrightarrow \mathbb{C}^n \colon \boldsymbol{a} = a(x) \mapsto \left\{ a(\omega_m^j) \right\}_{j \in \mathbb{Z}_m^\times}.$$

This map is well-defined because ω_m^j is a root of $\Phi_m(x)$ for each $j \in \mathbb{Z}_m^\times$.

The coefficient embedding preserves addition, and is therefore an additive group embedding. The canonical embeddings preserves addition and multiplication, and is therefore a ring embedding. Note that multiplication is defined component-wise in the embedding space $\mathbb{C}^n$.

To analyze the noise in a ciphertext, we will study its norm through the coefficient or canonical embedding. The notations

$$||\boldsymbol{a}||_p = ||\iota(\boldsymbol{a})||_p \qquad \text{and} \qquad ||\boldsymbol{a}||_p^{\mathrm{can}} = ||\tau(\boldsymbol{a})||_p$$

denote the ℓ_p-norm on the coefficient embedding and the canonical embedding respectively. Since noise estimates are simpler for the canonical embedding, but decryption is done on the coefficient embedding, it can be useful to upper bound the coefficient norm in terms of the canonical norm. Fortunately, this is possible because any two norms on a finite-dimensional vector space are known to be equivalent. We refer to HElib for more information on how this can be done [34].

The norms satisfy the following lemma.

Lemma 1. *Let $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{K}$, then*

- $||\boldsymbol{a} + \boldsymbol{b}||_\infty^{\mathrm{can}} \le ||\boldsymbol{a}||_\infty^{\mathrm{can}} + ||\boldsymbol{b}||_\infty^{\mathrm{can}}$
- $||\boldsymbol{a} \cdot \boldsymbol{b}||_\infty^{\mathrm{can}} \le ||\boldsymbol{a}||_\infty^{\mathrm{can}} \cdot ||\boldsymbol{b}||_\infty^{\mathrm{can}}$
- $||\boldsymbol{a}||_\infty^{\mathrm{can}} \le ||\boldsymbol{a}||_1$.

The first property is the triangle inequality. The second and third property are given for example by Gentry et al. [32].

2.2   Additional Notations for R-LWE

Throughout this paper, we consider $t = t(x)$, which is either a polynomial in $\mathbb{Z}[x]$ or a non-zero element of $\mathcal{R}$, depending on the context. We write the quotient ring of $\mathcal{R}$ modulo t as $\mathcal{R}_t = \mathcal{R}/t\mathcal{R}$. All ring and field elements (except for the modulus t) are shown in bold lower case letters or explicitly as polynomials. For $\boldsymbol{a} \in \mathcal{K}$ (which can have non-integral coefficients) and a positive integer N, we denote the coefficient-wise centered reduction of $\boldsymbol{a}$ modulo N by $[\boldsymbol{a}]_N$. In other words, this gives the element in $N\mathcal{R} + \boldsymbol{a}$ which has coefficients in $[-N/2, N/2)$. We employ the standard notations $\lfloor \boldsymbol{a} \rfloor$, $\lceil \boldsymbol{a} \rceil$ and $\lfloor \boldsymbol{a} \rceil$ for coefficient-wise flooring, ceiling and rounding to the nearest integer, respectively. The result of rounding goes upwards if the input coefficient is in $\mathbb{Z} + 1/2$.

We will regularly use vectors and matrices over $\mathcal{R}$. Row vectors are written as $\mathsf{a} \in \mathcal{R}^{1 \times \ell}$, column vectors as $\overrightarrow{\boldsymbol{a}} \in \mathcal{R}^{\ell \times 1}$ and matrices as $\overrightarrow{\mathsf{a}} \in \mathcal{R}^{\ell_1 \times \ell_2}$. For the inner product between vectors of the same type, we use $\langle \cdot, \cdot \rangle$. Finally, we note

that the above notations for modular reduction, flooring, ceiling and rounding carry over component-wise to vectors and matrices.

We will require probability distributions to define the R-LWE problem and related homomorphic encryption schemes. The distribution $\mathcal{U}_q$ denotes the uniform distribution on $\mathcal{R}_q$. We also consider two distributions on $\mathcal{R}$, namely χ_{key} and χ_{err} for key and error generation respectively.

2.3   The Ideal Norm and the Resultant

To count the number of elements in a quotient ring, we can use the ideal norm.

Definition 3. *Let $\mathcal{I}$ be an ideal in $\mathcal{R}$, then the* absolute norm *of $\mathcal{I}$ is*

$$N(\mathcal{I}) = [\mathcal{R} : \mathcal{I}].$$

Lemma 2. *Let $t \in \mathcal{R}$ be non-zero, then $N(t\mathcal{R}) = |N_{\mathcal{K}/\mathbb{Q}}(t)|$ where $N_{\mathcal{K}/\mathbb{Q}}(\cdot)$ is the standard field norm.*

This lemma shows that the absolute norm and the field norm are compatible for principal ideals. For the proof, we refer to Marcus [50]. A direct corollary is that the ring $\mathcal{R}_t$ is finite for non-zero t.

Definition 4. *Let $\mathbb{P}_k \subseteq \mathbb{Z}[x]$ be the set of polynomials of degree at most k. Consider $f(x), g(x) \in \mathbb{Z}[x]$ of degree i and j respectively. The* Sylvester map *of $f(x)$ and $g(x)$ is the linear transformation*

$$\mathbb{P}_{j-1} \oplus \mathbb{P}_{i-1} \to \mathbb{P}_{i+j-1} \colon (r(x), s(x)) \mapsto r(x) \cdot f(x) + s(x) \cdot g(x).$$

If we use the power basis of x to express the Sylvester map as a matrix, then the determinant of this matrix is called the resultant $\text{Res}(f(x), g(x))$.

Observe that the image of the Sylvester map is a subset of the ideal $(f(x), g(x))$ in $\mathbb{Z}[x]$. The next lemma gives an alternative way to count the number of elements in the ring $\mathcal{R}_t$ based on the relation between norms and resultants [52].

Lemma 3. *Let $t(x) \in \mathbb{Z}[x]$, then $N_{\mathcal{K}/\mathbb{Q}}(t(x)) = \text{Res}(\Phi_m(x), t(x))$.*

The following lemma is a standard result (we refer to Knapp [42] for a proof).

Lemma 4. *For $f(x), g(x) \in \mathbb{Z}[x]$, it holds that*

- $\text{Res}(f(x), g(x))$ *is in the image of the Sylvester map.*
- $\text{Res}(f(x), g(x)) = 0$ *if and only if $f(x)$ and $g(x)$ have a common factor of degree at least one.*

The former statement generalizes Bézout's identity. Although the resultant is in the image of the Sylvester map, it is not necessarily the smallest positive integer with this property. As such, we use a definition of i Ventosa and Wiese [60,45].

Definition 5. *Let $f(x), g(x) \in \mathbb{Z}[x]$ have non-zero resultant. Then the* reduced resultant *or* congruence number $\text{Con}(f(x), g(x))$ *is the smallest positive integer in the image of the Sylvester map of $f(x)$ and $g(x)$.*

2.4   Ring Learning With Errors

The ring learning with errors problem [46] is an algebraic variant of the learning with errors problem [56]. Both are commonly used to construct homomorphic encryption schemes, but we will only need the variant over rings. The R-LWE problem is based on the R-LWE distribution for an integer $q \geq 2$ and a secret $\boldsymbol{s}$ sampled from χ_{key}.

Definition 6. *Fix a secret $\boldsymbol{s} \in \mathcal{R}_q$. The* R-LWE *distribution $A_{\boldsymbol{s}}^q$ is defined by first sampling $\boldsymbol{a} \leftarrow \mathcal{U}_q$, $\boldsymbol{e} \leftarrow \chi_{\text{err}}$ and then returning $(\boldsymbol{a}, [\boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}]_q)$.*

Definition 7. *Given access to polynomially many samples from $\mathcal{R}_q^2$, the* decision R-LWE problem *is to distinguish between the distributions $A_{\boldsymbol{s}}^q$ and $\mathcal{U}_q^2$.*

Definition 8. *Given access to polynomially many samples from $A_{\boldsymbol{s}}^q$, the* search R-LWE problem *is to find the underlying $\boldsymbol{s}$.*

Both variants of the R-LWE problem are conjectured to be hard for appropriately chosen parameters [46].

2.5   Basics of BFV and CLPX

This section introduces the secret key variants of BFV and CLPX. In fact, we describe the improved version of BFV encryption due to Kim et al. [40] where ring rounding is applied after multiplication by Δ. Details of the BGV scheme are omitted for conciseness, and because it is roughly equivalent to BFV.

BFV encryption. We fix a plaintext modulus $t = p$, a ciphertext modulus q and a scaling factor $\Delta = q/t$. Encryption of $\boldsymbol{m} \in \mathcal{R}_t$ is done via R-LWE:

$$\mathsf{ct} = \left([\lfloor \Delta \cdot \boldsymbol{m} \rceil + \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}]_q, -\boldsymbol{a}\right).$$

Decryption requires a ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ and the secret key $\boldsymbol{s} \in \mathcal{R}$:

$$\boldsymbol{m} = \lfloor (\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s})/\Delta \rceil.$$

One can homomorphically compute three types of operations over the plaintext space $\mathcal{R}_t$: addition, multiplication and automorphism [23]. The scheme can be made fully homomorphic by bootstrapping.

CLPX encryption. We fix a plaintext modulus $t = t(x) = x - b$, a ciphertext modulus q and a scaling factor $\Delta = \lfloor q/t \rceil$. The plaintext space corresponds to

$$\mathcal{R}_t = \mathbb{Z}[x]/(\Phi_m(x), x - b) = \mathbb{Z}[x]/(x - b, p) \cong \mathbb{Z}_p,$$

where $p = \Phi_m(b)$. Encryption of a single element $\mu \in \mathbb{Z}_p$ is done via R-LWE as follows. First, we compute a "hat encoding" $\hat{\boldsymbol{m}} = \mu \pmod{t\mathcal{R}}$ such that $\hat{\boldsymbol{m}}$ has small coefficients. Then the ciphertext is computed as

$$\mathsf{ct} = \left([\Delta \cdot \hat{\boldsymbol{m}} + \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}]_q, -\boldsymbol{a}\right).$$

Decryption requires a ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ and the secret key $\boldsymbol{s} \in \mathcal{R}$:

$$\hat{\boldsymbol{m}} = \lfloor (t/q) \cdot (\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) \rceil.$$

Finally, the original message is retrieved via $\hat{\boldsymbol{m}} = \hat{m}(x)$ and $\mu = \hat{m}(b)$. One can homomorphically compute two types of operations over the plaintext space $\mathbb{Z}_p$: addition and multiplication [15]. Since only a single element is encrypted, no SIMD operations are possible; and since the size of p is exponential in m, it is not known how to bootstrap efficiently for cryptographically secure parameters.

Observe that the hat encoder is redundant if we replace $\Delta \cdot \hat{\boldsymbol{m}}$ by $\lfloor (q/t) \cdot \mu \rceil$. This is done in our definition of the Generalized BFV scheme in Section 3, in accordance with the implementation of the BFV scheme due to Kim et al. [40]. However, the hat encoder is still required (generalized and renamed to Flatten) for plaintext-ciphertext multiplication.

SIMD operations. It was shown by Smart and Vercauteren [59] that one FHE plaintext can encode several independent numbers. Their idea is based on the Chinese remainder theorem (CRT). Specifically, let $t = p$ be a prime that does not divide m. Then it is a well-known fact that the m-th cyclotomic polynomial factors modulo p into $\ell = n/d$ distinct irreducible factors of degree d, where d is the order of p in $\mathbb{Z}_m^\times$. In other words, we have the CRT isomorphism

$$\mathcal{R}_p = \mathbb{Z}[x]/(\Phi_m(x), p) \to \mathbb{Z}[x]/(F_1(x), p) \times \ldots \times \mathbb{Z}[x]/(F_\ell(x), p) \tag{1}$$
$$\mu(x) \mapsto (\mu(x) \bmod F_1(x), \ldots, \mu(x) \bmod F_\ell(x)),$$

where $F_i(x)$ are the factors of $\Phi_m(x)$ modulo p. Consequently, the plaintext space is isomorphic to a direct product of ℓ copies of the finite field $\mathbb{F}_{p^d}$. In the case where $t = p^e$ is a prime power, one can apply Hensel lifting so that the plaintext space is given by ℓ copies of a Galois ring of characteristic p^e.

Gentry et al. [31] showed that the plaintext slots can be arbitrarily permuted based on the group action of $\mathrm{Gal}(\mathcal{K}/\mathbb{Q})$. This automorphism group contains the subgroup $\langle \sigma_p \rangle$ generated by the *Frobenius automorphism* σ_p. The Frobenius automorphism itself acts on each slot independently as an automorphism on the underlying Galois ring. It becomes even more interesting when considering automorphisms outside of $\langle \sigma_p \rangle$. These automorphisms can be shown to induce inter-slot permutations of the plaintext data.[4]

3   Generalized BFV Scheme

This section describes our generalization of BGV/BFV and CLPX to arbitrary cyclotomic rings and non-linear polynomial plaintext moduli. Although we describe the secret key variant of the scheme, it can easily be turned into a public key encryption scheme using standard techniques [23].

---

[4] In some situations, two automorphisms are needed to implement one properly defined permutation. However, this detail is outside the scope of this exposition.

3.1   Choosing Small Representatives

Our generalization captures plaintext spaces modulo arbitrary non-zero principal ideals generated by $t = t(x)$. In some procedures, we require a representative with small coefficients in $\mathcal{R}$ from an element in $\mathcal{R}_t$. To achieve this, we define the function

$$\mathsf{Flatten}_t \colon \mathcal{R}_t \to \mathcal{R} \colon \boldsymbol{m} \mapsto t \cdot \left[\frac{\boldsymbol{m}}{t} \right]_1 .$$

Note that $\mathsf{Flatten}$ generalizes both the hat encoder from Chen et al. [15] and the notation $[\cdot]_N$ (since $\mathsf{Flatten}_t(\boldsymbol{m}) = [\boldsymbol{m}]_t$ for an integer t). Moreover, it filters out a unique canonical representative in $\mathcal{R}$: it satisfies $\mathsf{Flatten}_t(\boldsymbol{m}) = \boldsymbol{m} \pmod{t\mathcal{R}}$, and the output does not depend on the input representative. Also note the similarity to Babai rounding [3] for approximating the closest vector problem.

3.2   Gadget Decomposition

Two additional functions are required for decomposition and recombination of ring elements. These functions are defined with respect to integers $\omega, q \geq 2$ and $\ell_{\omega,q} = \lceil \log_\omega(q) \rceil$, and they will be used to control the noise growth during key switching (see later). Let $\boldsymbol{a}' = [\boldsymbol{a}]_q$ for $\boldsymbol{a} \in \mathcal{R}$, then we define

$$\mathcal{D}_{\omega,q}(\boldsymbol{a}) = \left([\boldsymbol{a}']_\omega, \left\lfloor \left[\frac{\boldsymbol{a}'}{\omega} \right]_\omega \right\rfloor, \dots, \left\lfloor \left[\frac{\boldsymbol{a}'}{\omega^{\ell_{\omega,q}-1}} \right]_\omega \right\rfloor \right)^\top$$

and

$$\mathcal{P}_{\omega,q}(\boldsymbol{a}) = \left([\boldsymbol{a}']_q, [\boldsymbol{a}' \cdot \omega]_q, \dots, \left[\boldsymbol{a}' \cdot \omega^{\ell_{\omega,q}-1} \right]_q \right)^\top .$$

The following essential lemma is proven by Brakerski et al. [12].

Lemma 5. *For all $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R}$, it holds that*

$$\langle \mathcal{D}_{\omega,q}(\boldsymbol{a}), \mathcal{P}_{\omega,q}(\boldsymbol{b}) \rangle = \boldsymbol{a} \cdot \boldsymbol{b} \pmod{q\mathcal{R}}.$$

We note that alternative methods have been proposed to define $\mathcal{D}$ and $\mathcal{P}$, which are more convenient for the actual implementation of HE schemes. We refer to Genise et al. [28] for an overview of the state-of-the-art techniques.

3.3   Scheme Definition

The FHE scheme has plaintext space $\mathcal{R}_t$ and ciphertext space $\mathcal{R}_q^2$ for an integer q. For correctness, we will require that $||t(x)||_\infty^{\mathrm{can}} \ll q$ (similarly to BGV and BFV, where we assume that $t \ll q$). We also define the "scaling factor" as $\Delta = q/t \in \mathcal{K}$. We do not round the scaling factor to $\mathcal{R}$, which results in a conceptually simpler scheme definition than the original BFV and CLPX. The scheme then consists of the following algorithms for key generation, encryption and decryption:

– $\mathsf{SecretKeyGen}$: sample $\boldsymbol{s} \leftarrow \chi_{\mathrm{key}}$ and return $\boldsymbol{s}$.

– EvalKeyGen(s, s'): given secret keys $s, s' \in \mathcal{R}$, sample $\overrightarrow{a} \leftarrow \mathcal{U}_q^{\ell_{\omega,q}}$ and $\overrightarrow{e} \leftarrow \chi_{\mathrm{err}}^{\ell_{\omega,q}}$, and compute

$$\overrightarrow{\mathsf{evk}} = \left([\mathcal{P}_{\omega,q}(s') + \overrightarrow{a} \cdot s + \overrightarrow{e}]_q, -\overrightarrow{a}\right).$$

Return $\overrightarrow{\mathsf{evk}}$.

– Encrypt(m, s): given message $m \in \mathcal{R}_t$ and secret key $s \in \mathcal{R}$, sample $a \leftarrow \mathcal{U}_q$ and $e \leftarrow \chi_{\mathrm{err}}$, and compute

$$\mathsf{ct} = \left([\lfloor \Delta \cdot m \rceil + a \cdot s + e]_q, -a\right).$$

Return ct. Observe that the computed ciphertext is independent of the chosen plaintext representative m due to the scaling by Δ.

– Decrypt(ct, s): given ciphertext $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ and secret key $s \in \mathcal{R}$, compute
$$m = \lfloor (c_0 + c_1 \cdot s)/\Delta \rceil.$$

Return m.

In a similar way as BFV, the IND-CPA security of the GBFV homomorphic encryption scheme (without any evaluation keys) can be reduced to the hardness of the decision R-LWE problem using a simple indistinguishability argument. The procedure EvalKeyGen, where s' depends on s, requires a circular security assumption on top of R-LWE.

The following algorithms are necessary to compute homomorphic operations on ciphertexts of the GBFV scheme:

– Add$(\mathsf{ct}, \mathsf{ct}')$: given ciphertexts $\mathsf{ct}, \mathsf{ct}' \in \mathcal{R}_q^2$, let $\mathsf{ct} = (c_0, c_1)$ and $\mathsf{ct}' = (c_0', c_1')$. Now compute
$$\mathsf{ct}_{\mathrm{add}} = ([c_0 + c_0']_q, [c_1 + c_1']_q)$$

and return $\mathsf{ct}_{\mathrm{add}}$.

– Add(ct, m): given ciphertext $\mathsf{ct} \in \mathcal{R}_q^2$ and message $m \in \mathcal{R}_t$, compute

$$\mathsf{ct}' = ([\lfloor \Delta \cdot m \rceil]_q, 0)$$

and return Add$(\mathsf{ct}, \mathsf{ct}')$.

– KeySwitch$(c, \overrightarrow{\mathsf{evk}})$: given partial ciphertext $c \in \mathcal{R}_q$ and evaluation key $\overrightarrow{\mathsf{evk}} = (\overrightarrow{r_0}, \overrightarrow{r_1}) \in \mathcal{R}_q^{\ell_{\omega,q} \times 2}$, compute

$$\overrightarrow{c} = \mathcal{D}_{\omega,q}(c), \quad \mathsf{ct}_{\mathrm{switch}} = \left([\langle \overrightarrow{c}, \overrightarrow{r_0} \rangle]_q, [\langle \overrightarrow{c}, \overrightarrow{r_1} \rangle]_q\right).$$

Return $\mathsf{ct}_{\mathrm{switch}}$.

- $\mathsf{Multiply}(\mathsf{ct}, \mathsf{ct}', \overrightarrow{\mathsf{evk}})$: given ciphertexts $\mathsf{ct}, \mathsf{ct}' \in \mathcal{R}_q^2$ and evaluation key $\overrightarrow{\mathsf{evk}} \in \mathcal{R}_q^{\ell_{\omega,q} \times 2}$ for $\boldsymbol{s}' = \boldsymbol{s}^2$, let $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ and $\mathsf{ct}' = (\boldsymbol{c}_0', \boldsymbol{c}_1')$. Now compute

$$\mathsf{ct}'' = ([\lfloor (\boldsymbol{c}_0 \cdot \boldsymbol{c}_0')/\Delta \rceil]_q, [\lfloor (\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0')/\Delta \rceil]_q), \quad \boldsymbol{c}_2'' = [\lfloor (\boldsymbol{c}_1 \cdot \boldsymbol{c}_1')/\Delta \rceil]_q.$$

  Compute $\mathsf{ct}''' = \mathsf{KeySwitch}(\boldsymbol{c}_2'', \overrightarrow{\mathsf{evk}})$ and return $\mathsf{Add}(\mathsf{ct}'', \mathsf{ct}''')$.

- $\mathsf{Multiply}(\mathsf{ct}, \boldsymbol{m})$: given ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$ and message $\boldsymbol{m} \in \mathcal{R}_t$, let $\hat{\boldsymbol{m}} = \mathsf{Flatten}_t(\boldsymbol{m})$. Now compute

$$\mathsf{ct}_{\mathrm{mult}} = ([\hat{\boldsymbol{m}} \cdot \boldsymbol{c}_0]_q, [\hat{\boldsymbol{m}} \cdot \boldsymbol{c}_1]_q)$$

  and return $\mathsf{ct}_{\mathrm{mult}}$. Note that $\mathsf{Flatten}$ is computed over $\mathcal{K}$, which may be more expensive than computations over $\mathcal{R}$. However, in many applications such as bootstrapping, this can be preprocessed if $\boldsymbol{m}$ is known in advance.

- $\mathsf{Automorphism}(\mathsf{ct}, \sigma, \overrightarrow{\mathsf{evk}})$: given ciphertext $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) \in \mathcal{R}_q^2$, automorphism $\sigma \in \mathcal{G}$ where

$$\mathcal{G} = \{ \sigma \in \mathrm{Gal}(\mathcal{K}/\mathbb{Q}) \mid \sigma(t) \in t\mathcal{R} \}$$

  and evaluation key $\overrightarrow{\mathsf{evk}} \in \mathcal{R}_q^{\ell_{\omega,q} \times 2}$ for $\boldsymbol{s}' = \sigma(\boldsymbol{s})$, compute

$$\mathsf{ct}' = ([(\sigma(t)/t) \cdot \sigma(\boldsymbol{c}_0)]_q, 0), \quad \boldsymbol{c}_1' = [(\sigma(t)/t) \cdot \sigma(\boldsymbol{c}_1)]_q.$$

  Compute $\mathsf{ct}'' = \mathsf{KeySwitch}(\boldsymbol{c}_1', \overrightarrow{\mathsf{evk}})$ and return $\mathsf{Add}(\mathsf{ct}', \mathsf{ct}'')$. Multiplication by $\sigma(t)/t$ is not required in the regular BFV scheme because it is equal to 1.

In the BGV and BFV schemes, all automorphisms of $\mathrm{Gal}(\mathcal{K}/\mathbb{Q})$ induce valid automorphisms on $\mathcal{R}_t$. This is different in the generalized scheme: for correctness, we impose that $\sigma(t) \in t\mathcal{R}$ (which is equivalent to $\sigma(t\mathcal{R}) = t\mathcal{R}$) such that σ is well-defined over $\mathcal{R}_t$.

Remark 1. Observe that the groups $\mathrm{Gal}(\mathcal{K}/\mathbb{Q}(t)) \subseteq \mathcal{G} \subseteq \mathrm{Gal}(\mathcal{K}/\mathbb{Q})$ are not equal in general. For example, let $m = 8$ and $t(x) = x^2 + 3x + 1$, then

- $\mathrm{Gal}(\mathcal{K}/\mathbb{Q}(t))$ contains only $x \mapsto x$.
- $\mathcal{G}$ contains $x \mapsto x^i$ for $i = 1, 7$. Note that $\sigma_7(t) = -x^2 t$ over $\mathcal{R}$ which shows that indeed $\sigma_7 \in \mathcal{G}$.
- $\mathrm{Gal}(\mathcal{K}/\mathbb{Q})$ contains $x \mapsto x^i$ for $i = 1, 3, 5, 7$.

Remark 2. Note that the scheme described in this section is totally general, i.e. it works for any non-zero plaintext modulus polynomial $t(x)$, and we have not imposed any restriction except that $\|t(x)\|_\infty^{\mathrm{can}} \ll q$. The above example already illustrates that a "compatible" choice of $\Phi_m(x)$ and $t(x)$ results in a non-trivial set of valid automorphisms. Similarly, such choice is also required to achieve non-trivial SIMD capabilities of the scheme.

3.4 Noise Analysis

This section provides a conservative worst-case noise analysis on the canonical embedding. As such, it demonstrates how the multiplication noise depends on the norm of t, and not directly on the precision parameter p studied in the next section. Heuristic average-case formulas are left as an open problem. We define the invariant noise of a ciphertext in the same way as CLPX [15].

Definition 9. *Let* $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ *be a ciphertext that encrypts* $\boldsymbol{m} \in \mathcal{R}_t$. *Its invariant noise is the field element* $\boldsymbol{v} \in \mathcal{K}$ *with smallest infinity norm on the coefficient embedding such that*

$$(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s})/\Delta = \boldsymbol{m} + \boldsymbol{v} \pmod{t\mathcal{R}}. \tag{2}$$

Observe that we can rewrite the above definition as

$$\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \Delta \cdot (\boldsymbol{m} + \boldsymbol{v}) \pmod{q\mathcal{R}}. \tag{3}$$

The following lemma gives a condition on the invariant noise for correctness of decryption, again similar to CLPX.

Lemma 6. *A ciphertext* $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ *that encrypts* $\boldsymbol{m} \in \mathcal{R}_t$ *decrypts correctly if the invariant noise* $\boldsymbol{v}$ *satisfies* $||\boldsymbol{v}||_\infty < 1/2$.

Proof. Let

$$(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s})/\Delta = \boldsymbol{m} + \boldsymbol{v} + t \cdot \boldsymbol{a}$$

for $\boldsymbol{a} \in \mathcal{R}$. Decryption computes

$$\lfloor (\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s})/\Delta \rceil = \lfloor \boldsymbol{m} + \boldsymbol{v} + t \cdot \boldsymbol{a} \rceil = \boldsymbol{m} + t \cdot \boldsymbol{a} = \boldsymbol{m} \pmod{t\mathcal{R}},$$

where the middle equality holds if $||\boldsymbol{v}||_\infty < 1/2$. $\qquad\square$

Additional symbols. We need to bound the ciphertext noise after encryption and all homomorphic operations. For this purpose, we assume that the key and error distributions, which were used earlier, are upper bounded. More specifically, we define three extra symbols:

- B_{key} is an upper bound on $||\boldsymbol{s}||_\infty^{\text{can}}$ for $\boldsymbol{s} \leftarrow \chi_{\text{key}}$.
- B_{err} is an upper bound on $||\boldsymbol{e}||_\infty^{\text{can}}$ for $\boldsymbol{e} \leftarrow \chi_{\text{err}}$.
- B_t is defined as $||t(x)||_\infty^{\text{can}}$.

The next lemma bounds the "ring rounding" error that occurs when rounding an element from $\mathcal{K}$ to $\mathcal{R}$.

Lemma 7. *Let* $\boldsymbol{a} \in \mathcal{K}$ *and* $\boldsymbol{b} = \lfloor \boldsymbol{a} \rceil \in \mathcal{R}$, *then* $||\boldsymbol{b} - \boldsymbol{a}||_\infty^{\text{can}} \leq n/2$.

Proof. Let $\boldsymbol{\epsilon} = \boldsymbol{b} - \boldsymbol{a}$. According to the third property of Lemma 1, we have

$$||\boldsymbol{\epsilon}||_\infty^{\text{can}} \leq ||\boldsymbol{\epsilon}||_1 \leq n/2.$$

The second inequality is obtained by bounding the coefficients of $\boldsymbol{\epsilon}$ by $1/2$. $\quad\square$

Initial noise. Let $\mathsf{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1)$ be a freshly encrypted ciphertext. It satisfies

$$\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \lfloor \Delta \cdot \boldsymbol{m} \rceil + \boldsymbol{e} = \Delta \cdot \boldsymbol{m} + \boldsymbol{\epsilon} + \boldsymbol{e} \pmod{q\mathcal{R}},$$

where $\boldsymbol{e}$ is sampled from χ_{err} and $\boldsymbol{\epsilon}$ is the ring rounding error. The invariant noise is given by $\boldsymbol{v} = (\boldsymbol{\epsilon} + \boldsymbol{e})/\Delta$. It can be bounded as

$$||\boldsymbol{v}||_\infty^{\mathrm{can}} \leq (n/2 + B_{\mathrm{err}}) \cdot B_t/q.$$

Ciphertext-ciphertext addition. The added ciphertext satisfies

$$\begin{aligned}
[\boldsymbol{c}_0 + \boldsymbol{c}_0']_q + [\boldsymbol{c}_1 + \boldsymbol{c}_1']_q \cdot \boldsymbol{s} &= (\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) + (\boldsymbol{c}_0' + \boldsymbol{c}_1' \cdot \boldsymbol{s}) \\
&= \Delta \cdot (\boldsymbol{m}_{\mathrm{add}} + \boldsymbol{v}_{\mathrm{add}}) \pmod{q\mathcal{R}},
\end{aligned}$$

where $\boldsymbol{m}_{\mathrm{add}} = \boldsymbol{m} + \boldsymbol{m}'$ and $\boldsymbol{v}_{\mathrm{add}} = \boldsymbol{v} + \boldsymbol{v}'$ are the added message and invariant noise respectively. The noise can be bounded as

$$||\boldsymbol{v}_{\mathrm{add}}||_\infty^{\mathrm{can}} \leq ||\boldsymbol{v}||_\infty^{\mathrm{can}} + ||\boldsymbol{v}'||_\infty^{\mathrm{can}}.$$

Plaintext-ciphertext addition. We replace the second term by a ring rounding error. Then the formula changes to $||\boldsymbol{v}_{\mathrm{add}}||_\infty^{\mathrm{can}} \leq ||\boldsymbol{v}||_\infty^{\mathrm{can}} + (n/2) \cdot B_t/q$.

Key switching. Suppose that we have $\boldsymbol{c} \cdot \boldsymbol{s}' = \Delta \cdot (\boldsymbol{m} + \boldsymbol{v}) \pmod{q\mathcal{R}}$. The key switched ciphertext satisfies

$$\begin{aligned}
\left[\langle \overrightarrow{\boldsymbol{c}}, \overrightarrow{\boldsymbol{r}_0} \rangle\right]_q + \left[\langle \overrightarrow{\boldsymbol{c}}, \overrightarrow{\boldsymbol{r}_1} \rangle\right]_q \cdot \boldsymbol{s} &= \left[\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}), \mathcal{P}_{\omega,q}(\boldsymbol{s}') + \overrightarrow{\boldsymbol{a}} \cdot \boldsymbol{s} + \overrightarrow{\boldsymbol{e}} \rangle\right]_q + \\
&\quad \left[\langle \mathcal{D}_{\omega,q}(\boldsymbol{c}), -\overrightarrow{\boldsymbol{a}} \rangle\right]_q \cdot \boldsymbol{s} \\
&= \langle \mathcal{D}_{\omega,q}(\boldsymbol{c}), \mathcal{P}_{\omega,q}(\boldsymbol{s}') \rangle + \langle \mathcal{D}_{\omega,q}(\boldsymbol{c}), \overrightarrow{\boldsymbol{a}} \cdot \boldsymbol{s} \rangle + \\
&\quad \langle \mathcal{D}_{\omega,q}(\boldsymbol{c}), \overrightarrow{\boldsymbol{e}} \rangle + \langle \mathcal{D}_{\omega,q}(\boldsymbol{c}), -\overrightarrow{\boldsymbol{a}} \rangle \cdot \boldsymbol{s} \\
&= \boldsymbol{c} \cdot \boldsymbol{s}' + \langle \mathcal{D}_{\omega,q}(\boldsymbol{c}), \overrightarrow{\boldsymbol{e}} \rangle \\
&= \Delta \cdot (\boldsymbol{m} + \boldsymbol{v}_{\mathrm{switch}}) \pmod{q\mathcal{R}},
\end{aligned}$$

where the third equality follows from Lemma 5 and

$$\boldsymbol{v}_{\mathrm{switch}} = \boldsymbol{v} + \langle \mathcal{D}_{\omega,q}(\boldsymbol{c}), \overrightarrow{\boldsymbol{e}} \rangle / \Delta.$$

Recall that $\overrightarrow{\boldsymbol{e}}$ is sampled from $\chi_{\mathrm{err}}^{\ell_{\omega,q}}$. The noise can be bounded as

$$||\boldsymbol{v}_{\mathrm{switch}}||_\infty^{\mathrm{can}} \leq ||\boldsymbol{v}||_\infty^{\mathrm{can}} + B_{\mathrm{switch}},$$

where $B_{\mathrm{switch}} = \ell_{\omega,q} \cdot (\omega \cdot n/2) \cdot B_{\mathrm{err}} \cdot B_t/q$. The factor $\omega \cdot n/2$ represents decomposition of $\boldsymbol{c}$ in base ω, which uses a similar observation as in Lemma 7.

Ciphertext-ciphertext multiplication. Before key switching, the multiplied ciphertext satisfies

$$
\begin{aligned}
\boldsymbol{c}_0'' + \boldsymbol{c}_1'' \cdot \boldsymbol{s} + \boldsymbol{c}_2'' \cdot \boldsymbol{s}^2 &= [\lfloor (\boldsymbol{c}_0 \cdot \boldsymbol{c}_0')/\Delta \rceil]_q + [\lfloor (\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0')/\Delta \rceil]_q \cdot \boldsymbol{s} + \\
&\qquad [\lfloor (\boldsymbol{c}_1 \cdot \boldsymbol{c}_1')/\Delta \rceil]_q \cdot \boldsymbol{s}^2 \\
&= (\boldsymbol{c}_0 \cdot \boldsymbol{c}_0')/\Delta + (\boldsymbol{c}_0 \cdot \boldsymbol{c}_1' + \boldsymbol{c}_1 \cdot \boldsymbol{c}_0')/\Delta \cdot \boldsymbol{s} + \\
&\qquad (\boldsymbol{c}_1 \cdot \boldsymbol{c}_1')/\Delta \cdot \boldsymbol{s}^2 + (\boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s} + \boldsymbol{\epsilon}_2 \cdot \boldsymbol{s}^2) \\
&= (\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) \cdot (\boldsymbol{c}_0' + \boldsymbol{c}_1' \cdot \boldsymbol{s})/\Delta + (\boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s} + \boldsymbol{\epsilon}_2 \cdot \boldsymbol{s}^2) \\
&= \Delta \cdot (\boldsymbol{m} \cdot \boldsymbol{m}' + \boldsymbol{m} \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot \boldsymbol{m}' + \boldsymbol{v} \cdot \boldsymbol{v}') + \\
&\qquad (\boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s} + \boldsymbol{\epsilon}_2 \cdot \boldsymbol{s}^2) \pmod{q\mathcal{R}}.
\end{aligned}
$$

Note that at this point, we have to fix two particular representatives $\boldsymbol{m}, \boldsymbol{m}' \in \mathcal{R}$ rather than $\boldsymbol{m}, \boldsymbol{m}' \in \mathcal{R}_t$. This is so that we can define the decryption formula from Equation (2) without reduction modulo $t\mathcal{R}$. The elements $\boldsymbol{\epsilon}_i$ are again ring rounding errors. Clearly, the intermediate noise is given by

$$
\begin{aligned}
\boldsymbol{v}_{\text{int}} &= \boldsymbol{m} \cdot \boldsymbol{v}' + \boldsymbol{v} \cdot \boldsymbol{m}' + \boldsymbol{v} \cdot \boldsymbol{v}' + \frac{\boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s} + \boldsymbol{\epsilon}_2 \cdot \boldsymbol{s}^2}{\Delta} \\
&= \boldsymbol{v}' \cdot (\boldsymbol{m} + \boldsymbol{v}) + \boldsymbol{v} \cdot (\boldsymbol{m}' + \boldsymbol{v}') - \boldsymbol{v} \cdot \boldsymbol{v}' + \frac{\boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s} + \boldsymbol{\epsilon}_2 \cdot \boldsymbol{s}^2}{\Delta} \\
&= \boldsymbol{v}' \cdot \frac{\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}}{\Delta} + \boldsymbol{v} \cdot \frac{\boldsymbol{c}_0' + \boldsymbol{c}_1' \cdot \boldsymbol{s}}{\Delta} - \boldsymbol{v} \cdot \boldsymbol{v}' + \frac{\boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s} + \boldsymbol{\epsilon}_2 \cdot \boldsymbol{s}^2}{\Delta}.
\end{aligned}
$$

The noise can be bounded as

$$
\begin{aligned}
||\boldsymbol{v}_{\text{int}}||_\infty^{\text{can}} \leq\ & (n/2) \cdot (B_{\text{key}} + 1) \cdot B_t \cdot (||\boldsymbol{v}||_\infty^{\text{can}} + ||\boldsymbol{v}'||_\infty^{\text{can}}) + \\
& (||\boldsymbol{v}||_\infty^{\text{can}} \cdot ||\boldsymbol{v}'||_\infty^{\text{can}}) + (n/2) \cdot (1 + B_{\text{key}} + B_{\text{key}}^2) \cdot B_t/q.
\end{aligned}
$$

The final noise (after key switching) can be bounded as

$$
||\boldsymbol{v}_{\text{mult}}||_\infty^{\text{can}} \leq ||\boldsymbol{v}_{\text{int}}||_\infty^{\text{can}} + B_{\text{switch}}.
$$

Plaintext-ciphertext multiplication. Different from addition, the equations for plaintext-ciphertext multiplication deviate much from ciphertext-ciphertext multiplication. That is, the multiplied ciphertext satisfies

$$
\begin{aligned}
[\hat{\boldsymbol{m}} \cdot \boldsymbol{c}_0]_q + [\hat{\boldsymbol{m}} \cdot \boldsymbol{c}_1]_q \cdot \boldsymbol{s} &= \hat{\boldsymbol{m}} \cdot (\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s}) \\
&= \Delta \cdot (\boldsymbol{m}_{\text{mult}} + \boldsymbol{v}_{\text{mult}}) \pmod{q\mathcal{R}},
\end{aligned}
$$

where $\boldsymbol{m}_{\text{mult}} = \boldsymbol{m} \cdot \boldsymbol{m}'$ and $\boldsymbol{v}_{\text{mult}} = \hat{\boldsymbol{m}} \cdot \boldsymbol{v}$. Note that $\boldsymbol{m}'$ indicates the plaintext encrypted by the ciphertext $(\boldsymbol{c}_0, \boldsymbol{c}_1)$. Here we used the important property that $\mathsf{Flatten}_t(\boldsymbol{m}) = \boldsymbol{m} \pmod{t\mathcal{R}}$. The invariant noise can then be bounded as

$$
||\boldsymbol{v}_{\text{mult}}||_\infty^{\text{can}} \leq (n/2) \cdot B_t \cdot ||\boldsymbol{v}||_\infty^{\text{can}}.
$$

Automorphism. Before key switching, the computed ciphertext satisfies

$$[(\sigma(t)/t) \cdot \sigma(\boldsymbol{c}_0)]_q + [(\sigma(t)/t) \cdot \sigma(\boldsymbol{c}_1)]_q \cdot \sigma(\boldsymbol{s}) = (\sigma(t)/t) \cdot \sigma(\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s})$$
$$= \Delta \cdot (\sigma(\boldsymbol{m}) + \sigma(\boldsymbol{v})) \pmod{q\mathcal{R}},$$

where we used the fact that $(\sigma(t)/t) \cdot \sigma(\Delta) = \Delta$. Clearly, the intermediate noise is given by

$$\boldsymbol{v}_{\text{int}} = \sigma(\boldsymbol{v}), \qquad \text{so} \qquad ||\boldsymbol{v}_{\text{int}}||_\infty^{\text{can}} = ||\boldsymbol{v}||_\infty^{\text{can}}.$$

The final noise (after key switching) can be bounded as

$$||\boldsymbol{v}_{\text{auto}}||_\infty^{\text{can}} \leq ||\boldsymbol{v}_{\text{int}}||_\infty^{\text{can}} + B_{\text{switch}}.$$

4 Algebraic Structure of the Plaintext Space

This section studies the algebraic structure of the plaintext space. We start with the special case of binomial moduli (plus some additional assumptions on the exact shape of the binomial) and then we treat more general moduli.

4.1 Plaintext Space for a Binomial Modulus

We will use the following standard property of cyclotomic polynomials. We refer to the literature [1] for a proof of the lemma and for a more detailed discussion about the properties of cyclotomics.

Lemma 8. *Let $r = \text{rad}(m)$ denote the radical of a positive integer m, i.e. the product of its distinct prime factors. Then the following relation holds:*

$$\Phi_m(x) = \Phi_r(x^{m/r}).$$

We will now derive properties of the plaintext space when the modulus is of the special shape $t(x) = x^k - b$, where both k and b are integers. We will assume that $0 < k < n = \varphi(m)$ and $k \mid (m/r)$, where $r = \text{rad}(m)$ is the radical of m.

Our plaintext ring is $\mathcal{R}_t = \mathbb{Z}[x]/\mathcal{I}$, using the ideal $\mathcal{I} = (\Phi_m(x), t(x)) \subseteq \mathbb{Z}[x]$. In our special case, this can be simplified with Lemma 8 and Euclidean division (i.e. by substituting $\Phi_m(x)$ with its reduction modulo $t(x) = x^k - b$) to

$$\mathcal{I} = (\Phi_r(x^{m/r}), x^k - b) = (t(x), p),$$

where $p = \Phi_r(b^{m/(rk)})$. The next lemma shows that for some combinations, the splitting behaviour of $t(x)$ modulo p is extremely nice.

Lemma 9. *Let $m \geq 3$ be an integer and let $r = \text{rad}(m)$ be its radical. Consider $0 < k < n = \varphi(m)$ such that $k \mid (m/r)$. For an integer b, define $t(x) = x^k - b$ and $p = \Phi_r(b^{m/(rk)})$. If p is a prime number and does not divide m, then $t(x)$ splits over $\mathbb{F}_p$ into $\ell' = k/d$ distinct irreducible factors of identical degree d, where d is the multiplicative order of p modulo m. The subgroup $\mathcal{G}$ of valid automorphisms equals $\mathcal{G} = \text{Gal}(\mathcal{K}/\mathbb{Q}(t))$ and consists of the maps $x \mapsto x^i$ for $i = 1 \pmod{m/k}$.*

Proof. Observe that $\Phi_m(x) \in (t(x), p)$, and thus $t(x)$ divides $\Phi_m(x)$ over $\mathbb{F}_p[x]$. As such, the splitting behaviour of $t(x)$ over $\mathbb{F}_p$ follows directly from the splitting behaviour of $\Phi_m(x)$ over $\mathbb{F}_p$, which is well known to split into $\varphi(m)/d$ distinct irreducible factors of degree d, with d the multiplicative order of p modulo m. Since $\Phi_m(x)$ splits completely over $\mathbb{F}_{p^d}$, the same holds for $t(x)$. Moreover, its roots are primitive m-th roots of unity, so the order of b is exactly m/k.

We now analyze the subgroup of valid automorphisms. Recall that $\sigma_i \colon x \mapsto x^i$ is valid if and only if

$$\sigma_i(t) = x^{k \cdot i} - b \in t\mathcal{R} \iff b^i - b \in t\mathcal{R}.$$

This equivalence holds because $x^{k \cdot i} = b^i \pmod{t\mathcal{R}}$. As such, we need $p \mid b^i - b$, or even $p \mid b^{i-1} - 1$ since p and b are coprime by definition. This is true if $i - 1$ is divisible by the order of b modulo p, which was established to be m/k. Moreover, all these valid automorphisms satisfy $\sigma_i(t) = t$. □

SIMD operations. As a direct corollary of Lemma 9, we can pack multiple elements in one plaintext based on the splitting behaviour of $t(x)$ modulo p. In fact, the degree d of the factors of $t(x)$ is identical to the BFV scheme. We can also compute arbitrary permutations of the plaintext slots in a similar way as HElib [34]. That is, we first replace Equation (1) by the isomorphism

$$\mathcal{R}_t = \mathbb{Z}[x]/(t(x), p) \to \mathbb{Z}[x]/(T_1(x), p) \times \ldots \times \mathbb{Z}[x]/(T_{\ell'}(x), p)$$
$$\mu(x) \mapsto (\mu(x) \bmod T_1(x), \ldots, \mu(x) \bmod T_{\ell'}(x)). \tag{4}$$

Define the *slot algebra* $\mathbb{F}_{p^d} = \mathbb{F}_p(\zeta)$, where ζ is a formal root of $T_1(x)$ over $\mathbb{F}_p$. Then ζ is also a root of $\Phi_m(x)$, so it is a primitive m-th root of unity. The roots of $t(x)$ over $\mathbb{F}_{p^d}$ are simply obtained by twisting ζ with the k-th roots of unity. We therefore obtain them as

$$\zeta^{(m/k) \cdot i + 1} \quad \text{for} \quad 0 \le i < k.$$

In particular, the roots of $T_1(x)$ are the p-th power maps of ζ. Let $S \subseteq \mathbb{Z}$ be a full system of representatives for $\mathcal{H}/\langle p \rangle$, where $\mathcal{H} \cong \mathcal{G}$ is the subgroup of $\mathbb{Z}_m^\times$ whose elements are congruent to 1 modulo m/k. Equation (4) is updated to

$$\mu(x) \mapsto \left\{ \mu(\zeta^h) \right\}_{h \in S}.$$

This is possible because all ζ^h are roots of $t(x)$ belonging to different $T_i(x)$.

The so-called hypercube representatives are constructed as

$$S = \left\{ g_1^{e_1} \cdot \ldots \cdot g_s^{e_s} \mid 0 \le e_i < \ell_i' \right\},$$

where the number of slots is $\ell' = \ell_1' \cdot \ldots \cdot \ell_s'$ and s is the number of dimensions. As such, we can associate each slot with a tuple $(e_1, \ldots, e_s)$ or with $h = g_1^{e_1} \cdot \ldots \cdot g_s^{e_s}$. We use a procedure similar to the one in HElib [34] to compute g_i and ℓ_i'.

Rotations can be implemented by means of the automorphism group $\mathcal{G}$, in a similar way as for BFV [27]. Let $\boldsymbol{\alpha}$ be the mask obtained by embedding 0 in the

plaintext slots with indices $(e_1, \ldots, e_i, \ldots, e_t)$ where $e_i < v$, and embedding 1 in the other slots. Then the rotation with $0 \leq v < \ell'_i$ positions in dimension i for a plaintext $\boldsymbol{m}$ can be computed as

$$\boldsymbol{m} \mapsto \boldsymbol{\alpha} \cdot \sigma_j(\boldsymbol{m}) + (1 - \boldsymbol{\alpha}) \cdot \sigma_k(\boldsymbol{m}),$$

where $j = g_i^{-v} \pmod{m}$ and $k = g_i^{\ell'_i - v} \pmod{m}$. If the order of g_i in $\mathcal{H}$ is ℓ'_i, the equation collapses to $\boldsymbol{m} \mapsto \sigma_j(\boldsymbol{m})$ and we only need one automorphism. Finally, observe that the Frobenius automorphism σ_p acts on each slot separately as the p-th power map.

4.2   Plaintext Space for a More General Modulus

We slightly generalize the analysis of the previous sections to plaintext moduli where $p = \mathrm{Con}(\Phi_m(x), t(x))$ is prime and does not divide m (this means that p is unramified in $\mathcal{K}$). We remark that the scheme also works if p is not prime, but it would require a more careful analysis of the splitting behaviour for the different prime factors of p.

It is clear by definition that $p \in \mathcal{I} = (\Phi_m(x), t(x))$, so we may also write the ideal as $\mathcal{I} = (\Phi_m(x), t(x), p)$. Reducing to $\mathbb{F}_p[x]$, the ideal becomes principally generated by $t'(x) = \gcd(\Phi_m(x), t(x)) \in \mathbb{F}_p[x]$ so we conclude that $\mathcal{I} = (t'(x), p)$. A noteworthy difference with the binomial case is that when we consider $\mathcal{I}$ as an ideal in $\mathbb{Z}[x]$ (rather than in $\mathcal{R}$), it is not necessarily equal to $(t(x), p)$.

Remark 3. In the case of binomial plaintext moduli, arbitrary permutations can be computed as linear combinations of valid automorphisms. This is because the GBFV slots form an orbit under the group action of $\mathcal{G}/\langle \sigma_p \rangle$ on the BFV slots. This is not true in general as shown by a simple counterexample: take $m = 8$ and $t(x) = x^3 - 16x^2 + 256x - 4096$, then $p = 2^{16} + 1$ and $d = 1$. Now we have $\mathrm{Res}(\Phi_m(x), t(x)) = p^3$, so there are 3 valid plaintext slots while $|\mathcal{G}| = 1$. Note that arbitrary permutations may still be computed via conversion to BFV, but we do not elaborate this idea further.

Interpretation as a subspace of BFV. As already mentioned previously, the extension degree d of the slot algebra is identical for GBFV and BFV, which is no coincidence. In fact, the GBFV plaintext space is a subspace of $\mathcal{R}_p$. To see this, note that the definition of p implies that it is an element of $(\Phi_m(x), t(x))$, so we can write $p = \beta \cdot t$ in $\mathcal{R}$. But we know that $p\mathcal{R}$ splits in distinct prime ideals, so the Chinese remainder theorem implies that $\mathcal{R}_p \cong \mathcal{R}_\beta \times \mathcal{R}_t$. In this equation, GBFV only uses the slots corresponding to $\mathcal{R}_t$.

4.3   Hensel Lifting to Prime Powers

Some applications (bootstrapping in particular) require a plaintext space defined modulo a prime power p^e rather than a prime p. The following analysis shows

that this can be achieved by changing the plaintext modulus from t to t^e. Again, we assume that $p = \mathrm{Con}(\Phi_m(x), t(x))$ is prime and does not divide m.

Our starting point is the observation that $\Phi_m(x)$ is in the ideal $(t'(x), p)$, where $t'(x) = \gcd(\Phi_m(x), t(x))$ over $\mathbb{F}_p[x]$. It follows immediately that

$$\Phi_m(x) = \beta'(x) \cdot t'(x) \pmod{p}$$

for some $\beta'(x) \in \mathbb{Z}[x]$. Through the process of Hensel lifting, this equation may also be defined modulo p^e, so we can write

$$\Phi_m(x) = \beta'(x) \cdot t'(x) \pmod{p^e}. \tag{5}$$

We claim that $(\Phi_m(x), t^e(x)) = (t'(x), p^e)$ as ideals in $\mathbb{Z}[x]$. To see this, observe that $\Phi_m(x) \in (t'(x), p^e)$ due to Equation (5). We also know that $t(x) \in (t'(x), p)$ because $t'(x)$ divides $t(x)$ over $\mathbb{F}_p$ by construction. A simple binomial expansion of $t(x) = \gamma(x) \cdot t'(x) + \delta(x) \cdot p$ shows that $t^e(x) \in (t'(x), p^e)$, so it follows that

$$(\Phi_m(x), t^e(x)) \subseteq (t'(x), p^e).$$

In the opposite direction, note that both ideals have an index equal to $(N(t\mathcal{R}))^e$ when seen as additive subgroups of $\mathbb{Z}[x]$, so they are identical. The interpretation as a subspace of BFV (now with plaintext modulus p^e) also remains valid because we have $\mathcal{R}_{p^e} \cong \mathcal{R}_{\beta^e} \times \mathcal{R}_{t^e}$ by the Chinese remainder theorem.

4.4   Parameter Sets

In this section, we propose families of parameter sets for 16-bit, 32-bit, 64-bit and 128-bit cyclotomic prime moduli. These families accommodate a range of security levels, mainly determined by the degree of $\Phi_m(x)$. Tables illustrating the packing capacity vs. noise growth are deferred to Appendix A. The following list is non-exhaustive and was obtained via a simple script:

1. Let $m = 2^j$ and $t(x) = x^k - b$, with $k = 2^{i+j-5}$ and $b = 2^{2^i}$ for some integers $0 \le i \le 3$ and $5 \le j \le 16$ such that $p = \Phi_2(2^{16}) = 2^{16} + 1$.
2. Let $m = 2^j$ and $t(x) = x^k - b$, with $k = 2^{i+j-3}$ and $b = 288^{2^i}$ for some integers $0 \le i \le 1$ and $3 \le j \le 16$ such that $p = \Phi_2(288^4) = 288^4 + 1$.
3. Let $m = 3 \cdot 2^j$ and $t(x) = x^k - b$, with $k = 2^{i+j-6}$ and $b = 2^{2^i}$ for some integers $0 \le i \le 5$ and $6 \le j \le 16$ such that $p = \Phi_6(2^{32}) = 2^{64} - 2^{32} + 1$.
4. Let $m = 3 \cdot 2^j$ and $t(x) = x^k - b$, with $k = 2^{i+j-4}$ and $b = 236^{2^i}$ for some integers $0 \le i \le 3$ and $4 \le j \le 16$ such that $p = \Phi_6(236^8) = 236^{16} - 236^8 + 1$.

The polynomial $t(x)$ splits completely modulo $p = 1 \pmod{m}$ for all parameter families, and our plaintext space is thus isomorphic to $\mathbb{F}_p^k$. The method is fully parameterizable and inherently provides a flexible trade-off between number of slots and noise growth: a larger value of i results in larger k and b. This gives more slots but also more multiplication noise. In the extreme case where $b = 2$, the multiplication noise is completely dominated by the contribution inherent to the cyclotomic ring and (the Hamming weight of) the secret key distribution. As such, reducing the number of slots too aggressively becomes almost useless at a certain point. Finally, note that the group $\mathcal{G}$ is cyclic unless $i = 5$ in the third family or $i = 3$ in the fourth family.

Adjoining roots of unity. In zero-knowledge applications, one typically works in a quadratic or cubic field extension for the Goldilocks prime (i.e. the third parameter family) to achieve sufficient soundness. While it is possible to mimic computations in such extension fields via $\mathbb{F}_p$-arithmetic, we show that with a small tweak, one can also support these extension fields natively. To achieve this, we need to look for roots of unity that live in $\mathbb{F}_{p^d}$ but not in a strict subfield. That is, we look for small factors of $p^d - 1$ that are not already factors of $p^{d'} - 1$ for some $d' < d$. More specifically, it suffices to consider small prime factors of $\Phi_d(p)$. We propose the following augmented parameter sets:

1. To obtain a quadratic extension, we can adjoin a primitive 7-th root of unity to the cyclotomic ring (which is contained in $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$). As such, we update the parameters to $m = 7 \cdot 3 \cdot 2^j$ and $k = 7 \cdot 2^{i+j-6}$. This augmented parameter set does not satisfy the restrictions from Section 4.1 because $k \nmid (m/r)$. The number of slots over $\mathbb{F}_{p^2}$ is therefore not $k/2$, but $3 \cdot 2^{i+j-6}$.
2. To obtain a cubic extension, we can adjoin a primitive 9-th root of unity to the cyclotomic ring (which is contained in $\mathbb{F}_{p^3} \setminus \mathbb{F}_p$). As such, we update the parameters to $m = 9 \cdot 2^j$ and $k = 3 \cdot 2^{i+j-6}$. The number of slots over $\mathbb{F}_{p^3}$ is given by 2^{i+j-6}.

Eliminating roots of unity. Conversely, we can descend the third and fourth parameter family to the standard R-LWE problem over power-of-two cyclotomic rings by eliminating the primitive 3-th roots of unity. Let

$$N_{\mathcal{K}/\mathcal{K}'} \colon \mathcal{K} \to \mathcal{K}'$$

be the standard field norm from the $3 \cdot 2^j$-th to the 2^j-th cyclotomic field. Then the plaintext modulus $t' = N_{\mathcal{K}/\mathcal{K}'}(t)$ produces the same prime p, while it lives in a power-of-two cyclotomic ring. As a special case, we obtain native arithmetic modulo the Goldilocks prime in power-of-two cyclotomic rings, but with slightly more noise growth since $B_{t'} \approx B_t^2$ and using a smaller ring dimension.

Ring switching. In certain applications such as the delegated zkSNARK prover, one wants to minimize the size of the ciphertext resulting from a computation. It can happen that one is only interested in a ciphertext that encrypts a subset of the plaintext vector. To select such a subset, note that the above families define a tower of cyclotomic fields. That is, we have the natural embedding

$$\rho \colon \mathcal{R}' = \mathbb{Z}[x]/(\Phi_m(x)) \to \mathcal{R} = \mathbb{Z}[y]/(\Phi_{2^a \cdot m}(y)) : x \mapsto y^{2^a}$$

whenever m is even. Furthermore, this embedding respects the definition of $t(x)$ for a fixed i when j is replaced by $j + a$ in the parameter sets above. As such, the families also define compatible plaintext spaces. This allows us to apply ring switching, in order to transform a ciphertext defined over $\mathcal{R}$ to a corresponding ciphertext over $\mathcal{R}'$ that encrypts part of the original plaintext vector. We refer to Gentry et al. [29] for more details, but in short, it suffices to perform key switching to a secret key that lives in $\rho(\mathcal{R}') \subset \mathcal{R}$, select the slots one is interested in using a linear transformation, and finally map to $\mathcal{R}'$ using the trace function.

5   Bootstrapping

This section proposes a novel GBFV bootstrapping method inspired by regular BFV bootstrapping. We first review BFV bootstrapping, and then propose novel methods to bootstrap single GBFV ciphertexts and batches of ciphertexts.

5.1   Reviewing BFV Bootstrapping

Historically, the "native" BGV/BFV bootstrapping approach was first studied theoretically [12,30,2] and then implemented in HElib [35]. Later research has shown how it can be improved by optimizing the involved polynomials [13,26,48] and linear transformations [25,49]. Below we describe the "thin" bootstrapping workflow due to Chen and Han [13], because it will be used in later sections:

1. Evaluate a homomorphic linear transformation to map the slots of the input ciphertext to the coefficients of a different ciphertext.
2. Evaluate a homomorphic inner product to convert a noisy encryption of $\boldsymbol{m}$ to a low-noise encryption of $p \cdot \boldsymbol{m} + \boldsymbol{e}$.
3. Evaluate a homomorphic linear transformation to map the coefficients of the ciphertext to the slots of a different ciphertext.
4. Evaluate a homomorphic digit removal polynomial to cancel the terms e_i.

These four steps are summarized in Figure 1, where $\mathsf{Enc}_p(\boldsymbol{m})$ denotes an encryption of $\boldsymbol{m}$ under plaintext modulus p. For simplicity, we assume that the used plaintext moduli are a prime p and its square. This is sufficient for our large-p use case, but it could also be generalized to higher powers of p.

Alternatives of the native approach. Recently, there were many alternative BFV bootstrapping proposals. This includes a method to use the slots more efficiently [53], functional bootstrapping [44,43], and even an algorithm that uses
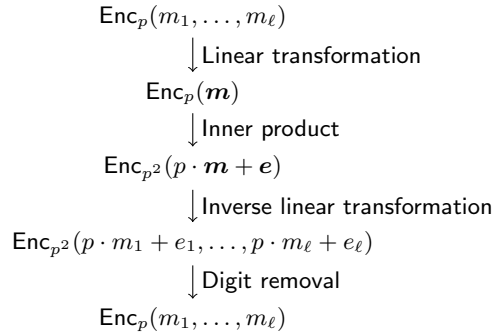
$$\mathsf{Enc}_p(m_1, \ldots, m_\ell)$$
$$\downarrow \text{Linear transformation}$$
$$\mathsf{Enc}_p(\boldsymbol{m})$$
$$\downarrow \text{Inner product}$$
$$\mathsf{Enc}_{p^2}(p \cdot \boldsymbol{m} + \boldsymbol{e})$$
$$\downarrow \text{Inverse linear transformation}$$
$$\mathsf{Enc}_{p^2}(p \cdot m_1 + e_1, \ldots, p \cdot m_\ell + e_\ell)$$
$$\downarrow \text{Digit removal}$$
$$\mathsf{Enc}_p(m_1, \ldots, m_\ell)$$

Fig. 1. Thin bootstrapping workflow, adapted from [13,27]

CKKS bootstrapping as a subroutine [41]. While finding the optimal method is an interesting research question, this paper does not intend to answer it. Instead, we use the native approach and improve it in Section 5.4 and Section 5.5.

5.2   A First Attempt at GBFV Bootstrapping

Our first (and failed) idea to bootstrap the GBFV scheme was to work with a temporary plaintext modulus of t^2 (instead of p^2 in Figure 1). This approach required us to switch the *ciphertext modulus* from q to t^2 right before the inner product step. This is technically possible if we introduce a ring rounding error:

$$\left\lfloor \frac{t^2}{q} \cdot \boldsymbol{c}_0 \right\rceil + \left\lfloor \frac{t^2}{q} \cdot \boldsymbol{c}_1 \right\rceil \cdot \boldsymbol{s} = t \cdot \left(\boldsymbol{m} + \boldsymbol{v} + \frac{\boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s}}{t} \right) \pmod{t^2 \mathcal{R}}. \qquad (6)$$

Note that this works correctly if the norm of $1/t$ is small enough (and otherwise, we can switch to a higher power of t). Let the newly obtained ciphertext from Equation (6) be denoted by $(\boldsymbol{c}'_0, \boldsymbol{c}'_1)$. We can extract its regular (i.e. non-invariant) noise $\boldsymbol{e} = t \cdot \boldsymbol{v} + \boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s}$ as

$$\mathsf{Flatten}_t(\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s}) = t \cdot \left[\frac{\boldsymbol{c}'_0 + \boldsymbol{c}'_1 \cdot \boldsymbol{s}}{t} \right]_1.$$

To finish the bootstrapping, we need to extract the noise homomorphically, so we need to implement $\mathsf{Flatten}_t$ as an arithmetic circuit.

We note that $\mathsf{Flatten}_t$ has a period of t, and when translated to the isomorphic space modulo p^2, this period becomes p. As a result, there exists a polynomial representation of the required functionality if p is prime [26]. However, since the interpolation space is $\mathbb{Z}_{p^2}$ for a possibly very large number p, the polynomial may have a huge degree of up to $2p - 1$. Moreover, the large-p bootstrapping trick from Ma et al. [48] does not seem to help here, because multiple small error coefficients are "spread" over one element of $\mathbb{Z}_{p^2}$.

5.3   Conversion and Packing of GBFV Ciphertexts

To overcome the previous obstacle, we use BFV bootstrapping as a subroutine. For this purpose, we propose an almost noise-free conversion method between GBFV and BFV, and a packing method that assembles multiple GBFV ciphertexts in a BFV ciphertext. These methods can bootstrap GBFV via a black-box call to BFV bootstrapping, but they may also be of independent interest.

GBFV-to-BFV conversion. Suppose we have a GBFV ciphertext as in Equation (3). We make the reduction modulo t explicit by introducing an extra term:

$$\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \Delta \cdot (\boldsymbol{m} + t \cdot \boldsymbol{a} + \boldsymbol{v}).$$

If we multiply both sides by t/p and round, we get

$$\left\lfloor \frac{t}{p} \cdot \boldsymbol{c}_0 \right\rceil + \left\lfloor \frac{t}{p} \cdot \boldsymbol{c}_1 \right\rceil \cdot \boldsymbol{s} = \frac{q}{p} \cdot (\boldsymbol{m} + t \cdot \boldsymbol{a} + \boldsymbol{v} + p \cdot (\boldsymbol{\epsilon}_0 + \boldsymbol{\epsilon}_1 \cdot \boldsymbol{s})/q).$$

The result is a BFV encryption of $\boldsymbol{m} + t \cdot \boldsymbol{a}$ under plaintext modulus p, where $\boldsymbol{a}$ is a random but irrelevant ring element.

In the other direction, suppose that we start from a BFV ciphertext

$$\boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \frac{q}{p} \cdot (\boldsymbol{m} + \boldsymbol{v}) \pmod{q\mathcal{R}}.$$

Then we simply multiply by p/t and get

$$\left(\frac{p}{t} \cdot \boldsymbol{c}_0\right) + \left(\frac{p}{t} \cdot \boldsymbol{c}_1\right) \cdot \boldsymbol{s} = \Delta \cdot (\boldsymbol{m} + \boldsymbol{v}) \pmod{q\mathcal{R}}.$$

Note that this only works correctly because p/t is an element of $\mathcal{R}$.

GBFV-to-BFV packing. When multiple GBFV ciphertexts are bootstrapped simultaneously, we can pack them together in one BFV ciphertext. To facilitate the packing step, we additionally assume a binomial modulus as in Lemma 9. Intuitively, the packing works because the group action of $\mathcal{G}/\langle \sigma_p \rangle$ partitions the BFV slots into n/k orbits, one of which corresponds to the GBFV slots.

To ease the notation, let us write $\beta = p/t \in \mathcal{R}$ as before. We are given a set of encryptions $\mathsf{Enc}_t(\boldsymbol{m}_i)$ for $0 \leq i < n/k$, where the subscript again denotes the plaintext modulus. These ciphertexts are equal to

$$\mathsf{Enc}_t(\boldsymbol{m}_i) = \mathsf{Enc}_p(\beta \cdot \boldsymbol{m}_i).$$

This identity can be seen by expanding the ciphertexts using Equation (2) and multiplying by β. Consider a subset of automorphisms $\{\sigma^{(0)}, \ldots, \sigma^{(n/k-1)}\}$ which forms a system of representatives for the quotient group $\mathrm{Gal}(\mathcal{K}/\mathbb{Q})/\mathcal{G}$. Packing homomorphically computes

$$\mathsf{ct} = \sum_{i=0}^{n/k-1} \sigma^{(i)}(\beta^{-1} \cdot \mathsf{Enc}_t(\boldsymbol{m}_i)) = \mathsf{Enc}_p\left(\sum_{i=0}^{n/k-1} \sigma^{(i)}(\beta^{-1}\beta \cdot \boldsymbol{m}_i)\right),$$

where the inverse of β is defined modulo t. To unpack the j-th message, we apply

$$\mathsf{ct}_j = \beta \cdot \sigma^{-(j)}(\mathsf{ct}) = \mathsf{Enc}_t\left(\sum_{i=0}^{n/k-1} \sigma^{-(j)}(\sigma^{(i)}(\beta^{-1}\beta \cdot \boldsymbol{m}_i))\right), \tag{7}$$

where $\sigma^{-(j)}$ denotes the inverse of $\sigma^{(j)}$. Packing costs n/k automorphisms and one multiplicative BFV level. Unpacking costs n/k automorphisms and no levels at all (multiplication by β implements the noiseless BFV-to-GBFV conversion routine and is hence not considered a level in terms of the invariant noise).

To prove correctness, we need to show that ct_j encrypts $\boldsymbol{m}_j$ in Equation (7). If i equals j, both automorphisms cancel and the remaining term $\beta^{-1}\beta \cdot \boldsymbol{m}_j$ is congruent to $\boldsymbol{m}_j$ modulo t. All other terms will disappear completely modulo t, because $\sigma^{(i)}$ and $\sigma^{(j)}$ are in different cosets of $\mathrm{Gal}(\mathcal{K}/\mathbb{Q})/\mathcal{G}$. More specifically, it can be seen from the following lemma that $\sigma^{-(j)}(\sigma^{(i)}(\beta))$ is divisible by t.

Lemma 10. *Let $\sigma \notin \mathcal{G}$ be an automorphism of the cyclotomic number field $\mathcal{K}$, then $\sigma(\beta)$ is divisible by t in $\mathcal{R}$ under the conditions of Lemma 9.*

Proof. The congruence number p is divisible by t and $\sigma(t)$ (because the division results are β and $\sigma(\beta)$ respectively). So if we can show that the ideals generated by t and $\sigma(t)$ are coprime, then $\sigma(\beta) = p/\sigma(t)$ is divisible by t and we are done.

Consider $\sigma \colon x \mapsto x^i$, then Lemma 9 implies that $i \neq 1 \pmod{m/k}$. Define the ideal $\mathcal{I} = (t, \sigma(t)) \subseteq \mathcal{R}$ and let $t(x) = x^k - b$. Clearly $p \in \mathcal{I}$ and also $b^i - b$ is in $\mathcal{I}$ because it is the reduction of $\sigma(t)$ modulo t. We proved in Lemma 9 that the multiplicative order of b modulo p is equal to m/k. In combination with the fact that $i \neq 1 \pmod{m/k}$, we find that $b^i - b$ is not divisible by p in $\mathbb{Z}$. But p is prime, so it must be coprime to $b^i - b$, thus $1 \in \mathcal{I}$ by Bézout's identity.   □

5.4   Single Bootstrapping via Conversion

Further improvements can be obtained by opening the black box from the previous section. In particular, BFV bootstrapping with a large value of p is somehow wasteful, because we do not fully use the available message space. Let us assume from now on that p is prime and congruent to 1 modulo m, so that $d = 1$. We propose an improved bootstrapping based on Section 5.1 as follows:
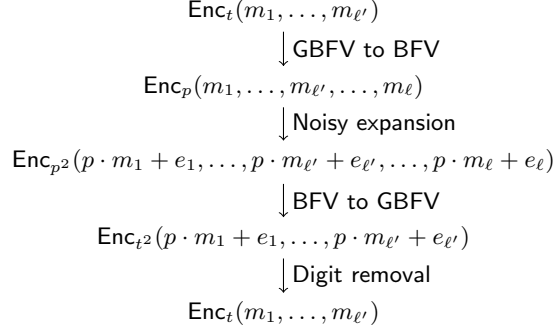
1. We start with an encryption $\mathsf{Enc}_t(m_1, \ldots, m_{\ell'})$, where ℓ' denotes the number of GBFV slots (to avoid confusion, we use a different symbol $\ell > \ell'$ to denote the number of BFV slots). In the first step, we convert this ciphertext from GBFV to BFV and get $\mathsf{Enc}_p(m_1, \ldots, m_{\ell'}, \ldots, m_\ell)$.
2. Evaluate the first three steps of thin bootstrapping in Section 5.1. We call this a slot-wise noisy expansion of the plaintext space.
3. Convert the obtained ciphertext from BFV modulus p^2 to GBFV modulus t^2. This step is a multiplication by β^2, which can be folded in the inverse linear transformation from the previous step, thereby saving a multiplicative level.
4. Evaluate an adapted digit removal polynomial. This is the polynomial from Ma et al. [48] (also used in [25,49]), multiplied by $\beta^{-1} \pmod{t}$. Division by t is automatic and is a simple reinterpretation of the result.

These four steps are summarized in Figure 2. They only cover the full splitting case, which suffices for many parameter sets. For $d > 1$, we need to include extra unpacking and repacking operations before and after digit removal [35,25].

This section improves over the black-box approach in terms of noise growth during digit extraction (which is typically the most depth-consuming step of bootstrapping). Remark that we now work with plaintext modulus t^2, which has much smaller norm than p^2 for typical parameter sets. Consequently, we get the beneficial multiplication noise growth from Section 3.4. This crucial improvement will allow us to use a smaller ring dimension of $n = 2^{14}$ than prior work.
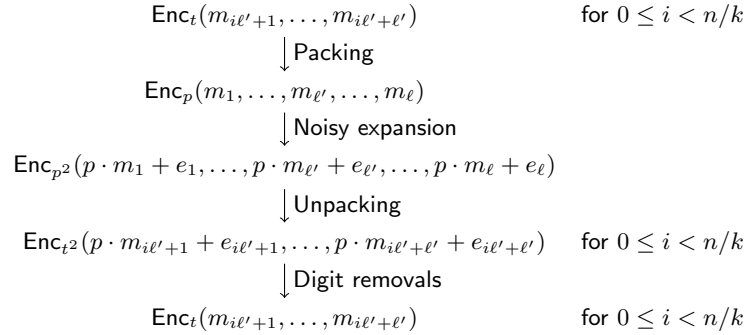
5.5   Batch Bootstrapping via Packing

To exploit the unused part of the message space, we can pack multiple GBFV ciphertexts together during the noisy expansion step. The packing limit is n/k GBFV ciphertexts per BFV ciphertext.

$$\mathsf{Enc}_t(m_1, \ldots, m_{\ell'})$$
$$\downarrow \text{GBFV to BFV}$$
$$\mathsf{Enc}_p(m_1, \ldots, m_{\ell'}, \ldots, m_\ell)$$
$$\downarrow \text{Noisy expansion}$$
$$\mathsf{Enc}_{p^2}(p \cdot m_1 + e_1, \ldots, p \cdot m_{\ell'} + e_{\ell'}, \ldots, p \cdot m_\ell + e_\ell)$$
$$\downarrow \text{BFV to GBFV}$$
$$\mathsf{Enc}_{t^2}(p \cdot m_1 + e_1, \ldots, p \cdot m_{\ell'} + e_{\ell'})$$
$$\downarrow \text{Digit removal}$$
$$\mathsf{Enc}_t(m_1, \ldots, m_{\ell'})$$

Fig. 2. GBFV single bootstrapping workflow

The steps for batch bootstrapping are displayed in Figure 3. The workflow is very similar to Figure 2, except that the conversion steps are replaced by packing and unpacking. Note that the packing multiplication by β^{-1} can be folded in the digit removal polynomial. The choice of doing unpacking before digit extraction, and not after digit extraction, is a design decision that reduces the noise growth.

6   Implementation and Results

We implemented the GBFV scheme on top of the Microsoft SEAL library [58]. Unfortunately, SEAL is restricted to power-of-two cyclotomic rings and moduli of 60 bits or less, which makes it impossible to implement all recommended parameter sets. However, there are two good reasons why we opted for SEAL: first, it supports the BFV scheme, which uses a GBFV-compatible "most significant bit" encoding. As a result, the implementation can be conveniently generalized

$$\mathsf{Enc}_t(m_{i\ell'+1}, \ldots, m_{i\ell'+\ell'}) \qquad\qquad \text{for } 0 \le i < n/k$$
$$\downarrow \text{Packing}$$
$$\mathsf{Enc}_p(m_1, \ldots, m_{\ell'}, \ldots, m_\ell)$$
$$\downarrow \text{Noisy expansion}$$
$$\mathsf{Enc}_{p^2}(p \cdot m_1 + e_1, \ldots, p \cdot m_{\ell'} + e_{\ell'}, \ldots, p \cdot m_\ell + e_\ell)$$
$$\downarrow \text{Unpacking}$$
$$\mathsf{Enc}_{t^2}(p \cdot m_{i\ell'+1} + e_{i\ell'+1}, \ldots, p \cdot m_{i\ell'+\ell'} + e_{i\ell'+\ell'}) \quad \text{for } 0 \le i < n/k$$
$$\downarrow \text{Digit removals}$$
$$\mathsf{Enc}_t(m_{i\ell'+1}, \ldots, m_{i\ell'+\ell'}) \qquad\qquad \text{for } 0 \le i < n/k$$

Fig. 3. GBFV batch bootstrapping workflow

to GBFV. Moreover, it allows us to extend the bootstrapping implementation due to Geelen [25] from BFV to GBFV. The recent work from Ma et al. [49] also implements bootstrapping, but in HElib and only for the BGV scheme. Note that these two works mark the state-of-the-art in native BGV/BFV bootstrapping.

In contrast to prior works about BGV/BFV, we are able to use a small ring dimension of $n = 2^{14}$ at 128-bit security, while still being able to pack a relatively large number of at least 1024 slots. This small ring dimension is enabled by the reduced multiplication noise of digit removal: since there is less noise growth, we can work with smaller q and n at constant security level. To the best of our knowledge, the only other method that can use ring dimension $n = 2^{14}$ (apart from the FHEW/TFHE branch) is BLEACH [21,4]. However, this scheme can only pack bits (whereas we pack 16-bit numbers), has a more expensive addition (i.e. XOR gate) and its bootstrapping has few remaining multiplicative levels.

We augmented the BFV bootstrapping implementation from Geelen [25] with sparse secret encapsulation [9]. The benchmarks in this section use similar parameters as BLEACH bootstrapping [4]: we take $n = 2^{14}$, a modulus $q \approx 2^{420}$ and a ternary secret key distribution with Hamming weight $h = 256$. The sparse key has Hamming weight $\tilde{h} = 32$ and we set the noise cut-off parameter [48,25] to $B = 15$ for failure rate less than 2^{-64}. We use the prime $p = 2^{16} + 1$ from the first recommended parameter family. Similarly to BFV bootstrapping [25], we also subtracted 15 bits from the initial and remaining noise budget to enable sub-domain interpolation of the noise over $[-B, B] \cap \mathbb{Z}$. The paragraphs below show experiments for single and batch bootstrapping operations. All experiments were conducted on a MacBook Pro (2021) equipped with an Apple M1 Max processor, 64 GB of RAM and running macOS Sonoma 14.7.1. The shown noise budget is the one reported by SEAL and is always normalized to p. As such, it gives the inherent noise rather than the invariant noise.

Experiments for single bootstrapping. Bootstrapping results for individual ciphertexts are given in Table 1. We applied 2-stage decomposition of the linear transformations in noisy expansion. We used the first recommended parameter family, which has $\ell' = k$ GBFV slots. The trade-off between number of slots and noise growth is clear from the table: the number of slots increases gradually from the left to the right column, while the remaining noise budget decreases. Increasing the number of slots to 16384 (which would coincide with regular BFV) is not possible for this parameter set, because the remaining noise budget would be negative. The total bootstrapping execution time is the lowest number ever demonstrated for BFV-like schemes at 128-bit security.

Experiments for batch bootstrapping. Similarly, we also generated results for batch bootstrapping in Table 2. The displayed number of slots reflects an individual ciphertext (the number of slots for a full batch is always 16384). The latency of single bootstrapping is better than batch bootstrapping, since in the latter multiple digit removal polynomials are evaluated. However, the through-put (number of bootstrapped slots times remaining capacity divided by total

Table 1. Results for GBFV single bootstrapping using $m = 2^{15}$ and $p = 2^{16} + 1$

Number of slots ℓ'		1024	2048	4096	8192
Bits per multiplicative level		11	12	14	18
Noise (bits)	Initial	317	317	317	317
	Noisy expansion	111	111	114	118
	Digit removal	82	91	113	161
	Remaining	124	115	90	38
Execution time (sec)	Noisy expansion	1.41	1.44	1.44	1.46
	Digit removal	0.53	0.54	0.54	0.55
	Total	1.94	1.98	1.98	2.01
Throughput	#slots · bits/sec	$65 \cdot 10^3$	$119 \cdot 10^3$	$186 \cdot 10^3$	$155 \cdot 10^3$
	#slots · #levels/sec	$5.8 \cdot 10^3$	$9.3 \cdot 10^3$	$12 \cdot 10^3$	$8.2 \cdot 10^3$

Table 2. Results for GBFV batch bootstrapping using $m = 2^{15}$ and $p = 2^{16} + 1$

Number of slots ℓ'		1024	2048	4096	8192
Bits per multiplicative level		11	12	14	18
Noise (bits)	Initial	317	317	317	317
	Packing	2	2	1	1
	Noisy expansion	126	125	126	126
	Unpacking	30	29	24	16
	Digit removals	82	91	113	161
	Remaining	77	70	53	13
Execution time (sec)	Packing	0.15	0.07	0.03	0.01
	Noisy expansion	1.52	1.46	1.43	1.43
	Unpacking	0.15	0.07	0.03	0.01
	Digit removals	8.56	4.23	2.14	1.06
	Total	10.38	5.83	3.63	2.51
Throughput	#slots · bits/sec	$122 \cdot 10^3$	$197 \cdot 10^3$	$239 \cdot 10^3$	$85 \cdot 10^3$
	#slots · #levels/sec	$11 \cdot 10^3$	$14 \cdot 10^3$	$14 \cdot 10^3$	0

execution time) of Table 2 is generally much higher than the corresponding column in Table 1 since noisy expansion is only evaluated once for the entire batch. The notable exception to this is the rightmost column, where the remaining noise budget of batch bootstrapping is less than a multiplicative level.

6.1 Comparison to Regular BFV Bootstrapping

This section compares the performance of BFV and GBFV bootstrapping. First, it should be noted that BFV bootstrapping is not possible with ring dimension $n = 2^{14}$ at a security level of 128 bits. As such, the results from Table 3 were generated using ring dimension $n = 2^{15}$ and ciphertext modulus $q \approx 2^{840}$. We applied 3-stage decomposition of the linear transformations for all benchmarks. The parameters in this section are optimized for throughput and hence result in higher latency than the previous section. The last row of Table 3 shows that the throughput of GBFV batch bootstrapping is roughly $1.3\times$ faster than BFV.

Table 3. Comparison to BFV bootstrapping using $m = 2^{16}$ and $p = 2^{16} + 1$

Bootstrapping algorithm		BFV	Single GBFV	Batch GBFV
Number of slots ℓ or ℓ'		32768	16384	16384
Total slot count		32768	16384	32768
Bits per multiplicative level		26	18	18
Noise (bits)	Initial	735	735	735
	Consumed	440	337	361
	Remaining	295	398	374
Execution time (sec)	Digit removal(s)	2.90	3.09	6.13
	Other operations	6.66	6.70	7.11
	Total	9.56	9.79	13.24
Throughput	#slots · bits/sec	$1011 \cdot 10^3$	$666 \cdot 10^3$	$926 \cdot 10^3$
	#slots · #levels/sec	$38 \cdot 10^3$	$37 \cdot 10^3$	$49 \cdot 10^3$

6.2   Limitations and Future Work

While the proposed algorithm is a significant improvement over well-known BFV bootstrapping in terms of noise growth, we stress that it cannot bootstrap all parameter sets. For example, CLPX uses a linear polynomial as the plaintext modulus, so the congruence number p is exponential in the ring dimension n. As a result, bootstrapping would require R-LWE with exponential modulus-to-noise ratio, because we need to compute the linear transformations modulo p. It is an open problem to achieve efficient bootstrapping for such parameters.

An important future work is implementing GBFV bootstrapping for the other parameter families (e.g. instantiated with the Goldilocks prime). This is currently not possible because SEAL only supports plaintext moduli of up to 60 bits. The implementation can also be further improved through modulus switching.

References

1. Al-Kateeb, A.Q.M., et al.: Structures and properties of cyclotomic polynomials. (2016)
2. Alperin-Sheriff, J., Peikert, C.: Practical bootstrapping in quasilinear time. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 8042, pp. 1–20. Springer (2013)
3. Babai, L.: On lovász' lattice reduction and the nearest lattice point problem. Comb. **6**(1), 1–13 (1986)
4. Bae, Y., Cheon, J.H., Kim, J., Stehlé, D.: Bootstrapping bits with CKKS. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 14652, pp. 94–123. Springer (2024)
5. Blindenbach, J.A., Cheon, J.H., Gürsoy, G., Kang, J.: On the overflow and p-adic theory applied to homomorphic encryption. In: CSCML. Lecture Notes in Computer Science, vol. 15349, pp. 268–279. Springer (2024)
6. Block, A.R., Tiwari, P.R.: On the concrete security of non-interactive FRI. Cryptology ePrint Archive, Paper 2024/1161 (2024)
7. Bloemen, R.: The goldilocks prime (2024), https://xn--2-umb.com/22/goldilocks
8. Bootland, C., Castryck, W., Iliashenko, I., Vercauteren, F.: Efficiently processing complex-valued data in homomorphic encryption. J. Math. Cryptol. **14**(1), 55–65 (2020)
9. Bossuat, J., Troncoso-Pastoriza, J.R., Hubaux, J.: Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation. In: ACNS. Lecture Notes in Computer Science, vol. 13269, pp. 521–541. Springer (2022)
10. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: CHIMERA: combining ring-lwe-based fully homomorphic encryption schemes. J. Math. Cryptol. **14**(1), 316–338 (2020)
11. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 868–886. Springer (2012)
12. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 13:1–13:36 (2014)
13. Chen, H., Han, K.: Homomorphic lower digits removal and improved FHE bootstrapping. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 10820, pp. 315–337. Springer (2018)
14. Chen, H., Iliashenko, I., Laine, K.: When HEAAN meets FV: A new somewhat homomorphic encryption with reduced memory overhead. In: IMACC. Lecture Notes in Computer Science, vol. 13129, pp. 265–285. Springer (2021)
15. Chen, H., Laine, K., Player, R., Xia, Y.: High-precision arithmetic in homomorphic encryption. In: CT-RSA. Lecture Notes in Computer Science, vol. 10808, pp. 116–136. Springer (2018)
16. Cheon, J.H., Cho, W., Kim, J., Stehlé, D.: Homomorphic multiple precision multiplication for CKKS and reduced modulus consumption. In: CCS. pp. 696–710. ACM (2023)
17. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 10624, pp. 409–437. Springer (2017)

18. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**(1), 34–91 (2020)
19. Cong, K., Moreno, R.C., da Gama, M.B., Dai, W., Iliashenko, I., Laine, K., Rosenberg, M.: Labeled PSI from homomorphic encryption with reduced computation and communication. In: CCS. pp. 1135–1150. ACM (2021)
20. Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K.E., Naehrig, M., Wernsing, J.: Manual for using homomorphic encryption for bioinformatics. Proc. IEEE **105**(3), 552–567 (2017)
21. Drucker, N., Moshkowich, G., Pelleg, T., Shaul, H.: BLEACH: cleaning errors in discrete computations over CKKS. J. Cryptol. **37**(1), 3 (2024)
22. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 9056, pp. 617–640. Springer (2015)
23. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144 (2012)
24. Gama, M., Beni, E.H., Kang, J., Spiessens, J., Vercauteren, F.: Blind zkSNARKs for private proof delegation and verifiable computation over encrypted data. Cryptology ePrint Archive, Paper 2024/1684 (2024), https://eprint.iacr.org/2024/1684
25. Geelen, R.: Revisiting the slot-to-coefficient transformation for BGV and BFV. Cryptology ePrint Archive, Paper 2024/153 (2024)
26. Geelen, R., Iliashenko, I., Kang, J., Vercauteren, F.: On polynomial functions modulo p^e and faster bootstrapping for homomorphic encryption. In: EUROCRYPT (3). Lecture Notes in Computer Science, vol. 14006, pp. 257–286. Springer (2023)
27. Geelen, R., Vercauteren, F.: Bootstrapping for BGV and BFV revisited. J. Cryptol. **36**(2), 12 (2023)
28. Genise, N., Micciancio, D., Polyakov, Y.: Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 11477, pp. 655–684. Springer (2019)
29. Gentry, C., Halevi, S., Peikert, C., Smart, N.P.: Field switching in bgv-style homomorphic encryption. J. Comput. Secur. **21**(5), 663–684 (2013)
30. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: Public Key Cryptography. Lecture Notes in Computer Science, vol. 7293, pp. 1–16. Springer (2012)
31. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 7237, pp. 465–482. Springer (2012)
32. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 850–867. Springer (2012)
33. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K.E., Naehrig, M., Wernsing, J.: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: ICML. JMLR Workshop and Conference Proceedings, vol. 48, pp. 201–210. JMLR.org (2016)
34. Halevi, S., Shoup, V.: Design and implementation of HElib: a homomorphic encryption library. Cryptology ePrint Archive, Paper 2020/1481 (2020)
35. Halevi, S., Shoup, V.: Bootstrapping for helib. J. Cryptol. **34**(1), 7 (2021)
36. Hamburg, M.: Ed448-goldilocks, a new elliptic curve. Cryptology ePrint Archive, Paper 2015/625 (2015)
37. Harmon, L., Delavignette, G., Roy, A., da Silva, D.W.H.A.: PIE: p-adic encoding for high-precision arithmetic in homomorphic encryption. In: ACNS (1). Lecture Notes in Computer Science, vol. 13905, pp. 425–450. Springer (2023)

38. Hoffstein, J., Silverman, J.H.: Optimizations for ntru. In: Proc. the Conf. on Public Key Cryptography and Computational Number Theory, Warsaw. pp. 77–88 (2000)
39. Hwang, I., Seo, J., Song, Y.: Concretely efficient lattice-based polynomial commitment from standard assumptions. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024. pp. 414–448. Springer Nature Switzerland, Cham (2024)
40. Kim, A., Polyakov, Y., Zucca, V.: Revisiting homomorphic encryption schemes for finite fields. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 13092, pp. 608–639. Springer (2021)
41. Kim, J., Seo, J., Song, Y.: Simpler and faster BFV bootstrapping for arbitrary plaintext modulus from CKKS. Cryptology ePrint Archive, Paper 2024/109 (2024)
42. Knapp, A.W.: Advanced algebra. Springer Science & Business Media (2007)
43. Lee, D., Min, S., Song, Y.: Functional bootstrapping for packed ciphertexts via homomorphic LUT evaluation. Cryptology ePrint Archive, Paper 2024/181 (2024)
44. Liu, Z., Wang, Y.: Relaxed functional bootstrapping: A new perspective on BGV/BFV bootstrapping. Cryptology ePrint Archive, Paper 2024/172 (2024)
45. Loeffler, D.: The resultant and the ideal generated by two polynomials in $\mathbb{Z}[x]$. MathOverflow, https://mathoverflow.net/q/17514
46. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM **60**(6), 43:1–43:35 (2013)
47. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-lwe cryptography. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 7881, pp. 35–54. Springer (2013)
48. Ma, S., Huang, T., Wang, A., Wang, X.: Accelerating BGV bootstrapping for large p using null polynomials over $\mathbb{Z}_{p^e}$. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 14652, pp. 403–432. Springer (2024)
49. Ma, S., Huang, T., Wang, A., Wang, X.: Faster BGV bootstrapping for power-of-two cyclotomics through homomorphic NTT. Cryptology ePrint Archive, Paper 2024/164 (2024)
50. Marcus, D.A., Sacco, E.: Number fields, vol. 1995. Springer (1977)
51. Era-boojum. GitHub (2023), https://github.com/matter-labs/era-boojum
52. Myerson, G.: Norms in polynomial rings. Bulletin of the Australian Mathematical Society **41**(3), 381–386 (1990)
53. Okada, H., Player, R., Pohmann, S.: Homomorphic polynomial evaluation using galois structure and applications to bfv bootstrapping. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023. pp. 69–100. Springer Nature Singapore, Singapore (2023)
54. Miden-vm. GitHub (2021), https://github.com/0xPolygonMiden/miden-vm
55. Plonky2. GitHub (2021), https://github.com/0xPolygonZero/plonky2
56. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (2009)
57. Risc zero. GitHub (2022), https://github.com/risc0/risc0
58. Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL (Jan 2023), microsoft Research, Redmond, WA.
59. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptogr. **71**(1), 57–81 (2014)
60. i Ventosa, X.T., Wiese, G.: Computing congruences of modular forms and galois representations modulo prime powers. Arithmetic, geometry, cryptography and coding theory 2009 **521**, 145–166 (2009)

A   Packing Capacity versus Noise Growth

The sections below illustrate the trade-off between packing capacity and multiplication noise growth for our 16-bit, 32-bit, 64-bit and 128-bit parameter families. The Hamming weight of the secret key was set to 128 for all tables. It is evident from all experiments that the noise can be reduced significantly by choosing $t(x)$ with small norm. Eventually, the noise gets bottlenecked by other factors, which are caused by the ring dimension and the secret key's Hamming weight.

A.1   16-bit prime $p = \Phi_2(2^{16}) = 2^{16} + 1$

We fix $m = 2^{15}$ and $t(x) = x^k - b$ with $k = 2^{i+10}$ and $b = 2^{2^i}$ for $0 \leq i \leq 3$.

i	0	1	2	3	BFV
Number of slots	1024	2048	4096	8192	16384
Noise PT $\times$ CT (bits)	6.4	7.3	9.1	13.2	21.1
Noise CT $\times$ CT (bits)	10.5	11.2	13.0	17.3	25.1

A.2   32-bit prime $p = \Phi_2(288^4) = 288^4 + 1$

We fix $m = 2^{15}$ and $t(x) = x^k - b$ with $k = 2^{i+12}$ and $b = 288^{2^i}$ for $0 \leq i \leq 1$.

i	0	1	BFV
Number of slots	4096	8192	16384
Noise PT $\times$ CT (bits)	13.2	21.7	38.0
Noise CT $\times$ CT (bits)	17.2	25.8	42.1

A.3   64-bit prime $p = \Phi_6(2^{32}) = 2^{64} - 2^{32} + 1$

Base field encoding. We fix $m = 3 \cdot 2^{14}$ and $t(x) = x^k - b$ with $k = 2^{i+8}$ and $b = 2^{2^i}$ for $0 \leq i \leq 5$. The slots are defined over $\mathbb{F}_p$.

i	0	1	2	3	4	5	BFV
Number of slots	256	512	1024	2048	4096	8192	16384
Noise PT $\times$ CT (bits)	6.5	7.4	9.2	13.1	21.3	37.3	68.9
Noise CT $\times$ CT (bits)	10.3	11.3	13.1	17.2	25.2	41.3	73.0

Alternatively, we may also use power-of-two cyclotomics by fixing $m' = 2^{14}$ and $t' = N_{\mathcal{K}/\mathcal{K}'}(t)$, where $t = t(x)$ is the same as above and $0 \leq i \leq 4$. The slots are still defined over $\mathbb{F}_p$.

i	0	1	2	3	4	BFV
Number of slots	256	512	1024	2048	4096	8192
Noise PT $\times$ CT (bits)	6.9	8.9	12.9	20.7	37.0	68.7
Noise CT $\times$ CT (bits)	11.4	12.9	16.7	24.9	40.7	72.8

Quadratic extensions. We fix $m = 7 \cdot 3 \cdot 2^{11}$ and $t(x) = x^k - b$ with $k = 7 \cdot 2^{i+5}$ and $b = 2^{2^i}$ for $0 \le i \le 5$. The slots are defined over $\mathbb{F}_{p^2}$.

i	0	1	2	3	4	5	BFV
Number of slots	96	192	384	768	1536	3072	6144
Noise PT $\times$ CT (bits)	6.3	7.3	9.1	12.9	20.9	37.0	69.2
Noise CT $\times$ CT (bits)	10.2	11.1	13.0	16.9	24.7	41.0	73.0

Cubic extensions. We fix $m = 9 \cdot 2^{12}$ and $t(x) = x^k - b$ with $k = 3 \cdot 2^{i+6}$ and $b = 2^{2^i}$ for $0 \le i \le 5$. The slots are defined over $\mathbb{F}_{p^3}$.

i	0	1	2	3	4	5	BFV
Number of slots	64	128	256	512	1024	2048	4096
Noise PT $\times$ CT (bits)	6.2	6.9	9.0	12.9	21.1	36.6	68.8
Noise CT $\times$ CT (bits)	10.1	10.8	12.9	16.9	24.8	40.9	73.0

A.4   128-bit prime $p = \Phi_6(236^8) = 236^{16} - 236^8 + 1$

We fix $m = 3 \cdot 2^{14}$ and $t(x) = x^k - b$ with $k = 2^{i+10}$ and $b = 236^{2^i}$ for $0 \le i \le 3$.

i	0	1	2	3	BFV
Number of slots	1024	2048	4096	8192	16384
Noise PT $\times$ CT (bits)	13.3	21.7	37.8	69.3	131.2
Noise CT $\times$ CT (bits)	17.2	25.4	41.7	73.2	135.4