Bootstrapping GBFV with CKKS

Jaehyung Kim

Stanford University
jaehk@stanford.edu

Abstract. The Generalized BFV [Geelen and Vercauteren; Eurocrypt'25] is an efficient fully homomorphic encryption scheme that supports integer computations over large cyclotomic moduli. However, the only known bootstrapping approach cannot support large precision as it uses BFV linear transformation as a subroutine. In this work, we introduce a GBFV bootstrapping that relies on CKKS bootstrapping as in the BFV bootstrapping from CKKS [Kim et al.; CCS'24]. The new bootstrapping can handle arbitrary precision, notably bootstrapping the CLPX scheme [Chen et al.; CT-RSA'18] for the first time, bootstrapping up to $500,000$ bits of plaintext modulus in less than 20 seconds. In addition, we introduce conversions between GBFV and CKKS and discuss its impact.

Keywords: GBFV · Bootstrapping · CLPX · CKKS.

1 Introduction

Fully homomorphic encryption (FHE) is a cryptography that allows computation in an encrypted state. Since Gentry's first scheme in 2009 [15], many schemes have been suggested. The BGV [8] and BFV [7, 13] schemes use the plaintext space $\mathbb{Z}_q[X]/\Phi_M(X)$ for a modulus $q \in \mathbb{Z}$ and a cyclotomic polynomial $\Phi_M(X)$. To achieve high precision, the CLPX scheme [9] takes quotient by a linear polynomial $X - b \in \mathbb{Z}[X]/\Phi_M(X)$, enabling a large plaintext space $\mathbb{Z}[X]/(\Phi_M(X), X-b) \cong \mathbb{Z}_{\Phi_M(b)}$ without significant noise growth. Recently, Geelen and Vercauteren suggested a generalization of both BFV and CLPX called generalized BFV (GBFV) [14] that takes the quotient by a more general polynomial, thereby introducing a flexible trade-off between the plaintext modulus size and the number of slots.

Although the original GBFV paper [14] instantiates a bootstrapping, it fails to support large precision like the case of CLPX [9]. This stems from the fact that the GBFV bootstrapping uses BFV linear transformation as a subroutine, and the noise growth of the linear transformation is proportional to the plaintext modulus. In this paper, we take a completely different approach. We revisit an alternative BFV bootstrapping from CKKS bootstrapping [18]. The philosophy of bootstrapping is that CKKS-bootstrapping the BFV noise results in BFV bootstrapping. We adopt the same idea: CKKS-bootstrapping the GBFV noise leads to GBFV bootstrapping, after small modifications.

1.1 Technical Overview

Let $N > 1$ be a power-of-two integer. Let $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. Given a power-of-two integer k that divides N and an integer $b \neq 0$, we consider a polynomial $t(X) = X^k - b$. The GBFV plaintext space is

$$\mathcal{R}_t = \mathcal{R}/t\mathcal{R} = \mathbb{Z}[X]/(X^N + 1, X^k - b) \cong \mathbb{Z}_{b^{N/k}+1}/(X^k - b).$$

Given a plaintext $m \in \mathcal{R}_t$, a GBFV ciphertext encrypting m can be written as

$$\mathsf{ct} = (-as + \lfloor (q/t) \cdot m \rceil + e, a) \in \mathcal{R}_q^2$$

where $s \in \mathcal{R}$ is a secret key. In other words, when we inner product ct with $(1, s)$, we get

$$\mathsf{ct} \cdot (1, s) = \lfloor (q/t) \cdot m \rceil + e,$$

where we encode m in the most significant bits via the scaling factor q/t and e is regarded as an error. The goal of the GBFV bootstrapping is to reduce the size of the error e.

Our key observation is that we can CKKS-bootstrap the error as in [18]. Our algorithm consists of the following four steps:

1. **Error Extraction**: It multiplies by t to extract the error,

$$t \cdot \mathsf{ct} = \mathsf{Enc}_s(te + e_{\mathsf{Round}}) \in \mathcal{R}_q^2,$$

   adding a small error e_{Round}.
2. **Error Bootstrapping**: It CKKS-bootstraps the error, i.e., homomorphically raise the ciphertext modulus while approximately preserving the underlying message. The result

$$\mathsf{CKKS.Bootstrap}(t \cdot \mathsf{ct}) = \mathsf{Enc}_s(te + e_{\mathsf{Round}} + e_{\mathsf{BTS}}) \in \mathcal{R}_{qq'}^2$$

   where $q' \simeq q$ and e_{BTS} is the CKKS bootstrapping error.
3. **Error Recovery**: It multiplies $\lfloor q'/t \rceil$ and rescale by q', getting

$$\mathsf{ct}' = \mathsf{Enc}_s(e + \lfloor (e_{\mathsf{Round}} + e_{\mathsf{BTS}})/t \rceil + e_{\mathsf{RS}}) \in \mathcal{R}_q^2$$

   where e_{RS} is (roughly) the rescaling error.
4. **Subtraction**: It subtracts ct' from the original ciphertext ct, having

$$\mathsf{Enc}_s(\lfloor (q/t) \cdot m \rceil + e')$$

   where $e' = -\lfloor (e_{\mathsf{Round}} + e_{\mathsf{BTS}})/r \rceil - e_{\mathsf{RS}}$ is small.

This can be written as the following informal theorem.[1]

Theorem 1 (Informal). *Leveraging a p-bit precision CKKS bootstrapping, we construct a GBFV bootstrapping that reduces the error by $p - O(1)$ bits.*

[1] The theorem is further specified in Theorem 2 in Section 3.

Next, we introduce a conversion from GBFV to CKKS and vice versa, exploring the usage of CKKS operations in the context of GBFV. The conversion from GBFV to CKKS is approximate in the sense that it decomposes the original message $m \in \mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b)$ into its possibly redundant digit-b representation. We observe that the conversion gives a CKKS ciphertext encrypting a polynomial of bounded norm and degree, allowing interesting operations despite the approximate (i.e. unexact) nature. In particular, we discuss the following two applications:

- **Arbitrary Function Evaluation**: The straightforward approach to evaluate arbitrary functions inside GBFV is to utilize univariate interpolation, but the polynomial degree is too high, requiring multiple bootstrappings. As our GBFV-to-CKKS conversion provides an approximate digit decomposition, we may use multivariate interpolation instead, significantly reducing the modulus consumption.
- **Approximate Modular Reduction**: As in [17], one may consider simulating $\mathbb{Z}_u$ arithmetic inside $\mathbb{Z}_{b^{N/k}+1}$ to allow a more general modulus. Here we observe that small polynomials in $\mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b)$ convert to polynomials of bounded degree, thereby allowing radix-based approximate modular reduction.

1.2 Contributions

The main idea of this paper is to leverage CKKS to make GBFV more efficient. The impact can be summarized as follows.

Large Precision Bootstrapping. Although the GBFV paper [14] instantiated a bootstrapping, their construction works only for small and specific moduli in practice, mainly because they rely on the subprocedure of the existing BFV bootstrapping methods. As the plaintext modulus becomes larger, the modulus consumption becomes extremely large, failing to be supported within practical RLWE dimensions like $\log N \leq 17$. On the other hand, our GBFV bootstrapping works almost independently of the underlying plaintext modulus, supporting an arbitrarily large plaintext modulus. Notably, we bootstrap CLPX [9], which is the special case of GBFV where $t(X) = X - b$ for the first time, bootstrapping plaintext modulus of size as large as $\simeq 500,000$ bits in less than 20 seconds. Note that this is not only new in the context of GBFV but also in the world of FHE in general. For instance, the largest plaintext modulus reported for bootstrapping was $\simeq 8,000$ bits in [4], which we improve by roughly two orders of magnitude, under the same RLWE dimension $N = 2^{16}$.[2]

Advanced Functionalities. In addition to the efficient GBFV bootstrapping from CKKS, we explore how the CKKS literature can help support more efficient

[2] According to their parameter table, they could have chosen at most $30,000$ bits, but it is still more than an order of magnitude away.

operations for GBFV. The main ingredient is a conversion from GBFV to CKKS (and vice versa), which can be regarded as an approximate b-digit decomposition (resp. recombination). As in the plaintext world, the digit decomposition can boost many operations. In this work, we mainly discuss arbitrary function evaluation and approximate modular reduction. Arbitrary function evaluation can be instantiated with multivariate interpolation over CKKS rather than univariate interpolation over GBFV, greatly improving efficiency such as modulus consumption. Approximate modular reduction can be used similarly as in [4], simulating arbitrary modulus computation inside a possibly smooth modulus $b^{N/k} + 1$. In addition, modular reduction enables rescaling and therefore fixed point arithmetic (as in the philosophy of CKKS), allowing efficient large precision fixed point real number computations.

1.3 Additional Related Works

High Precision FHE. An alternative way to achieve high precision integer arithmetic is to use the discrete variant of CKKS [12, 3, 17]. In [16], they take a radix-based approach, efficiently supporting arithmetic over moderate-sized integers like 32 or 64 bit arithmetic. In [4], they construct a nested CRT system inside RLWE to simulate CRT-based large integer arithmetic. One can also consider high precision CKKS bootstrapping, which bootstraps real numbers rather than integers. The state-of-the-art high precision CKKS bootstrapping is META-BTS [2], which is used as a black box in our construction.

Scheme Conversion. The concept of converting FHE schemes from one to the other for better efficiency is well studied [6, 19]. Not much is known about conversions to/from GBFV, and the only known conversion is a conversion between GBFV and BFV in [14]. Their conversion is approximate as in our case, and they mainly use it to construct their GBFV bootstrapping.

2 Preliminaries

Let $N > 1$ be a power-of-two integer. Let $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. Given $t \in \mathcal{R}$, let $\mathcal{R}_t = \mathcal{R}/t\mathcal{R}$. The infinity norm $\|\cdot\|_\infty$ on $\mathcal{R}$ denotes the maximum absolute value among its coefficients. We use the notation $\equiv_q$ (resp. $\equiv_t$) to denote the modulo q (resp. t) equivalence relation. Unless stated otherwise, $[\cdot]_q : \mathbb{Z} \to \mathbb{Z}_q = [-q/2, q/2) \cap \mathbb{Z}$ denotes the (signed) modular reduction by q. Given a polynomial $\alpha(X) \in \mathcal{R}$, there exists a representation $\alpha(X) = \sum_{i=0}^{N-1} \alpha_i X^i$. This gives a natural embedding $\mathcal{R} \to \mathbb{Z}[X]$, and we denote $\deg(\alpha)$ to denote the polynomial degree of the embedded polynomial as an element of $\mathbb{Z}[X]$.

2.1 GBFV basics

Let $1 \leq k \leq N$ be a power-of-two integer and $b \in \mathbb{Z} \setminus \{0, \pm 1\}$ be an integer. Let $t(X) = X^k - b \in \mathcal{R}$ be a polynomial and $\mathcal{R}_t$ be the plaintext space for GBFV.[3] The GBFV scheme description follows [14].

GBFV Encryption. Given a message $m \in \mathcal{R}_t$ and secret key $s \in \mathcal{R}$, the GBFV encryption of m with secret key s is defined as

$$\mathsf{GBFV.Encrypt}(m, s) = (-a \cdot s + \lfloor \Delta \cdot m \rceil + e, a) \in \mathcal{R}_q^2$$

where $\Delta = q/t$ is a scaling factor, $a \leftarrow \mathcal{R}_q$, and $e \leftarrow \chi_{err}$[4]. Conversely, given a ciphertext $\mathsf{ct} \in \mathcal{R}_q^2$, the GBFV decryption of ct by s is defined as

$$\mathsf{GBFV.Decrypt}(\mathsf{ct} = (c_0, c_1), s) = \lfloor (c_0 + c_1 \cdot s)/\Delta \rceil \in \mathcal{R}_t.$$

GBFV Bootstrapping. Let $\mathsf{ct} = (c_0, c_1)$ be a GBFV ciphertext encrypting a message $m \in \mathcal{R}_t$. This means that

$$(c_0, c_1) \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e$$

for some small error $e \in \mathcal{R}$. The **GBFV Bootstrapping** significantly reduces the size of the error while keeping the underlying message m. That is,

$$\mathsf{GBFV.Bootstrap}(\mathsf{ct}) \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e'$$

where $e' \ll e$. To quantify this, we can assume that there exists $B_1, B_2 > 0$ such that if the input error e satisfies $\|e\|_\infty \leq B_1$ then the output error e' satisfies $\|e'\|_\infty \leq B_2 < B_1$.

2.2 CKKS basics

CKKS Encryption. Given a message $m \in \mathbb{R}[X]/(X^N + 1)$ and secret key $s \in \mathcal{R}$, the (coefficients-encoded) CKKS encryption is defined as

$$\mathsf{CKKS.Encrypt}(m, s) = (-a \cdot s + \lfloor \Delta' \cdot m \rceil + e, a) \in \mathcal{R}_q^2$$

where $\Delta' \in \mathbb{R}_{>0}$ is a scaling factor, $a \leftarrow \mathcal{R}_q$, and $e \leftarrow \chi'_{err}$[5]. Conversely, given a ciphertext $\mathsf{ct} \in \mathcal{R}_q^2$, the CKKS decryption of ct by s is defined as

$$\mathsf{CKKS.Decrypt}(\mathsf{ct} = (c_0, c_1), s) = [c_0 + c_1 \cdot s]_q/\Delta' \in \mathbb{R}[X]/(X^N + 1).$$

The decryption of $\mathsf{ct} \in \mathcal{R}_q^2$ without scaling down

$$[\mathsf{ct} \cdot (1, s)]_q = \lfloor \Delta' \cdot m \rceil + e \in \mathcal{R}$$

is called the underlying plaintext of ct.

[3] Our idea naturally generalizes to general $t \in \mathcal{R}$. We keep $t(X) = X^k - b$ for simplicity.
[4] χ_{err} is an error distribution for GBFV.
[5] χ'_{err} is an error distribution for CKKS.

CKKS Rescaling. Given a CKKS ciphertext $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_Q^2$, the rescaling of ct by $q \mid Q$ is defined as

$$\mathsf{CKKS.RS}_q(\mathsf{ct}) = \left(\frac{c_0 - [c_0]_q}{q}, \frac{c_1 - [c_1]_q}{q} \right) \in \mathcal{R}_{Q/q}^2.$$

CKKS Bootstrapping. Let $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ be a CKKS ciphertext encrypting a (erroneous) message $m \in \mathbb{R}[X]/(X^N + 1)$. This means that

$$[(c_0, c_1) \cdot (1, s)]_q = \Delta' \cdot m.$$

The **CKKS Bootstrapping** [11] significantly increases the modulus while approximately preserving the message. That is, the result of the bootstrapping lives in $\mathcal{R}_Q^2$ for $Q \gg q$, and satisfies

$$[\mathsf{CKKS.Bootstrap}(\mathsf{ct}) \cdot (1, s)]_Q = \Delta' \cdot m + e$$

for some small $e \in \mathcal{R}$. We often assume that $\|m\|_\infty \leq 1$ (so that $\|\Delta' \cdot m\|_\infty \leq \Delta' \ll q$) and $\|e/\Delta'\|_\infty \leq 2^{-p}$, and denote the bootstrapping precision as p-bits.

3 Our GBFV Bootstrapping

In [18], they leverage CKKS bootstrapping to bootstrap a BFV ciphertext. This comes from the idea that CKKS-bootstrapping the BFV error and subtracting it from the original ciphertext leads to BFV bootstrapping (i.e. reducing the error). We use the same idea in the case of GBFV. The basic idea remains the same: CKKS-bootstrapping the GBFV error gives GBFV bootstrapping. However, the algorithm in [18] does not immediately translate to GBFV due to the difference in scaling factor shape. To solve this issue, we match the scaling factor via simple plaintext-ciphertext multiplications before and after bootstrapping.

3.1 Main Algorithm

We start with a GBFV ciphertext $\mathsf{ct} \in \mathcal{R}_q^2$ encrypting a message $m \in \mathcal{R}_t$. In other words, we have

$$\mathsf{ct} \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e$$

where $\Delta = q/t$ and $e \in \mathcal{R}$ is the underlying error. Since $\Delta \cdot t = 0$ modulo q, multiplying ct by t extracts the error. That is, $\mathsf{ct}' = t \cdot \mathsf{ct} \in \mathcal{R}_q^2$ satisfies

$$[\mathsf{ct}' \cdot (1, s)]_q = te + e_{\mathsf{Round}}$$

for some small error e_{Round}. Next, we CKKS-bootstrap the error ciphertext ct' to a modulus qq' where $q' \simeq q$. The resulting ciphertext $\mathsf{ct}'' \in \mathcal{R}_{qq'}^2$ satisfies

$$[\mathsf{ct}'' \cdot (1, s)]_{qq'} = te + e_{\mathsf{Round}} + e_{\mathsf{BTS}}$$

for some small bootstrapping error e_{BTS}. Finally, we multiply this ciphertext by $\lfloor q'/t \rceil$ and rescale by q', approximately dividing the underlying message by t. Since $q' \simeq q$, q'/t has enough precision and leads to correct division by t. Thus, the resulting ciphertext $\mathsf{ct}''' \in \mathcal{R}_q^2$ satisfies

$$[\mathsf{ct}''' \cdot (1, s)]_q = e - e'$$

for some small error $e' \ll e$. By subtracting ct''' from the original ciphertext ct, we get $\mathsf{ct}_{\mathsf{out}}$ whose underlying error e' is significantly smaller than the initial error e. We illustrate the detailed algorithm in Algorithm 1.

Algorithm 1: GBFV Bootstrapping from CKKS

Setting: $\Delta = q/t$, $q' \simeq q$. $\mathsf{CKKS.Bootstrap} : \mathcal{R}_q^2 \to \mathcal{R}_{qq'}^2$ introduces a small error e_{BTS}.
Input : $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ such that $\mathsf{ct} \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e$.
Output: $\mathsf{ct}_{\mathsf{out}} = (c_0', c_1') \in \mathcal{R}_q^2$ such that $\mathsf{ct}_{\mathsf{out}} \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e'$, s.t. $e' \ll e$.

$\mathsf{ct}' \leftarrow t \cdot \mathsf{ct} \in \mathcal{R}_q^2$;
$\mathsf{ct}'' \leftarrow \mathsf{CKKS.Bootstrap}(\mathsf{ct}') \in \mathcal{R}_{qq'}^2$;
$\mathsf{ct}''' \leftarrow \mathsf{CKKS.RS}_{q'}(\lfloor q'/t \rceil \cdot \mathsf{ct}') \in \mathcal{R}_q^2$;
$\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{ct} - \mathsf{ct}''' \in \mathcal{R}_q^2$;
return $\mathsf{ct}_{\mathsf{out}}$

Theorem 2 (Bootstrapping Correctness). *Let* $\mathsf{CKKS.Bootstrap} : \mathcal{R}_q^2 \to \mathcal{R}_{qq'}^2$ *be a CKKS bootstrapping that inputs a CKKS ciphertext encrypting a plaintext in* $[-B_1, B_1]^N \subseteq \mathcal{R}$ *and introducing an error in* $[-B_2, B_2]^N \subseteq \mathcal{R}$.[6] *Then the Algorithm 1 gives a GBFV bootstrapping whose input error is in* $[-B_1/(|b|+1) + 1/2, B_1/(|b|+1) - 1/2]^N$ *and output error* e' *satisfies*

$$\|e'\|_\infty \leq B_2 + \frac{|b| + h + 3}{2} + \frac{N}{2q'} \cdot (B_1 + B_2)$$

reducing the error by $\log_2(B_1/B_2) - O_q(1)$ *bits assuming that* $q' \gg N$.[7] *In other words, a p-bit precision CKKS bootstrapping gives a GBFV bootstrapping whose error ratio is* $p - O(1)$ *bits.*

Proof. Let $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ be a GBFV ciphertext such that

$$\mathsf{ct} \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e$$

[6] Here we consider a natural embedding $[-B_i, B_i] \subseteq \mathbb{R}^N \xrightarrow{\simeq} \mathcal{R}$.
[7] Here h is the secret key Hamming weight.

for a message $m \in \mathcal{R}_t$ and an error $e \in \mathcal{R}$ satisfying $\|e\|_\infty \le B_1/(|b|+1) - 1/2$. Since $t(X) = X^k - b$, we have that

$$\begin{aligned}\|te\|_\infty &= \|(X^k - b) \cdot e(X)\|_\infty \\ &\le \|X^k \cdot e(X)\|_\infty + \|b \cdot e(X)\|_\infty \\ &= (|b|+1) \cdot \|e\|_\infty \le B_1 - \frac{|b|+1}{2}.\end{aligned}$$

The rounding error e_{Round} is defined as

$$e_{\mathsf{Round}} = t \cdot (\Delta \cdot m - \lfloor \Delta \cdot m \rceil)$$

and satisfies

$$\|e_{\mathsf{Round}}\|_\infty \le (|b|+1) \cdot \|\Delta \cdot m - \lfloor \Delta \cdot m \rceil\|_\infty \le \frac{|b|+1}{2}.$$

Thus by triangular inequality, we have

$$\|te + e_{\mathsf{Round}}\|_\infty \le B_1,$$

becoming a valid input of $\mathsf{CKKS.Bootstrap}$. It ensures that the bootstrapping error e_{BTS} satisfies $\|e_{\mathsf{BTS}}\|_\infty \le B_2$. In addition, we have

$$\begin{aligned}[\mathsf{ct}''' \cdot (1, s)]_q &= \frac{(te + e_{\mathsf{Round}} + e_{\mathsf{BTS}}) \cdot \lfloor q'/t \rceil}{q'} + e_{\mathsf{RS}} \\ &= \frac{(te + e_{\mathsf{Round}} + e_{\mathsf{BTS}}) \cdot (q'/t - (q'/t - \lfloor q'/t \rceil))}{q'} + e_{\mathsf{RS}} \\ &= e + (e_{\mathsf{Round}} + e_{\mathsf{BTS}})/t - \frac{(te + e_{\mathsf{Round}} + e_{\mathsf{BTS}}) \cdot (q'/t - \lfloor q'/t \rceil)}{q'} + e_{\mathsf{RS}} \\ &= e - e'.\end{aligned}$$

Here e_{RS} is the rescaling error that satisfies

$$\|e_{\mathsf{RS}}\|_\infty \le \frac{h+2}{2}$$

where h is the secret key Hamming weight.[8] The final error e' satisfies

$$\begin{aligned}\|e'\|_\infty &\le \|(e_{\mathsf{Round}} + e_{\mathsf{BTS}})/t\|_\infty + \frac{N}{2q'} \cdot \|te + e_{\mathsf{Round}} + e_{\mathsf{BTS}}\|_\infty + \|e_{\mathsf{RS}}\|_\infty \\ &\le \|e_{\mathsf{Round}}\|_\infty + \|e_{\mathsf{BTS}}\|_\infty + \frac{N}{2q'} \cdot (\|te + e_{\mathsf{Round}}\|_\infty + \|e_{\mathsf{BTS}}\|_\infty) + \|e_{\mathsf{RS}}\|_\infty \\ &\le \frac{|b|+h+3}{2} + B_2 + \frac{N}{2q'} \cdot (B_1 + B_2).\end{aligned}$$

[8] See [10, Theorem 3.2] for proof.

In the second inequality, we use the fact that $|b| \geq 2$ so that

$$\begin{aligned}\|t \cdot \alpha(X)\|_\infty &= \|(X^k - b) \cdot \alpha(X)\|_\infty \\ &= \max_{0 \leq i < N} |b\alpha_i - \alpha_{i-k}| \\ &\geq \max_{0 \leq i < N} |\alpha_i| = \|\alpha(X)\|_\infty\end{aligned}$$

for all $\alpha(X) \in \mathbb{Q}[X]/(X^N + 1)$. This finishes the proof. □

3.2 Efficiency Discussions

We briefly discuss how the performance of our bootstrapping is affected by different parameters.

META-BTS [2] The most important metric for GBFV bootstrapping performance is the denoising factor (defined in [18] for BFV), which can be defined as the ratio between input and output GBFV noises. As noise grows after each multiplication, the denoising factor corresponds to the multiplicative depth between two consecutive bootstrappings. Theorem 2 asserts that the denoising factor is proportional to the CKKS bootstrapping precision. Hence, the higher the CKKS bootstrapping precision, the higher the GBFV multiplicative depth. The state-of-the-art method for high-precision CKKS bootstrapping is META-BTS [2], which iteratively bootstraps the errors to achieve higher precision. In particular, the bootstrapping precision is proportional to the number of iterations.

Theorem 3 (Theorem 3.2 of [2], Simplified). *Given a p-bit precision CKKS bootstrapping $\mathsf{BTS} : \mathcal{R}_q^2 \to \mathcal{R}_Q^2$, we may iterate BTS k times and achieve kp-bit precision CKKS bootstrapping $\mathsf{BTS}^{(k)} : \mathcal{R}_{2^{(k-1)p} \cdot q} \to \mathcal{R}_Q^2$.*

When combined with the correctness theorem, we get the following informal statement:

Corollary 1 (Bootstrapping Efficiency, Informal). *Given a CKKS bootstrapping with p-bits of bootstrapping precision, we may iterate the bootstrapping $k \geq 1$ times to construct a GBFV bootstrapping whose denoising factor is $O(k) = kp - O(1)$ bits and computational complexity is $O(k)$.*

See [18, Section 5.2] for more details, as most of the discussions in the section are also applicable in our GBFV bootstrapping.

The Choice of b. In practice, we can freely choose $t(X) = X^k - b$ so that we have a plaintext modulus of $b^{N/k} + 1$ and k slots. The parameter k introduces a trade-off between plaintext modulus size and the number of slots, while b does not affect the number of slots. When aiming for high precision, one may increase $|b|$ instead of decreasing k. Theorem 2 says that the bootstrapping input and output error bounds are functions of b.

Theorem 4 (Theorem 2, Simplified). *Given a fixed CKKS bootstrapping algorithm, the input error bound is $\Omega(1/|b|)$ and the output error bound is $O(|b|)$, thereby having $O(1) - O(\log(|b|)$ denoising factor.*

Since the plaintext modulus $b^{N/k} + 1$ is of size $O(\log(b))$ bits, the amount of increase in plaintext modulus bit size is linear in denoising factor decrease in bits. Concretely, since N/k can easily achieve thousands of bits already, choosing a very large b does not help much as it would increase the bootstrapping parameter which is worse. The rough behavior or error growth can be observed in Table 2.

4 Conversion between GBFV and CKKS

In this section, we describe how to convert a GBFV ciphertext into a CKKS ciphertext and vice versa, and what it provides in terms of functionality. This somewhat generalizes the algorithm in the previous section.

4.1 Conversions

Unlike GBFV to BFV conversion in [14], we identify a GBFV ciphertext with a corresponding digit-wise representation in CKKS. We first elaborate on this.

Definition 1 (Flatten [14]). *We identify an element in $\mathcal{R}_t$ as an element in $\mathcal{R}$ via the embedding $\mathsf{Flatten} : \mathcal{R}_t \to \mathcal{R}$ defined as*

$$\mathsf{Flatten}(m) = t \cdot \left[\frac{m}{t}\right]_1$$

where $[\cdot]_1$ denotes the (signed) modulo 1 function. It satisfies $\mathsf{Flatten}(m) \equiv_t m$.

Theorem 5. *Let $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ be a GBFV ciphertext encrypting a message $m \in \mathcal{R}_t$, satisfying*

$$\mathsf{ct} \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e$$

and $\|e\|_\infty < q/2$. Then we have

$$[\mathsf{ct} \cdot (1, s)]_q = \lfloor \Delta \cdot \mathsf{Flatten}(m) \rceil + q \cdot I + e$$

where $I \in \mathcal{R}$ is a small polynomial satisfying $\|I\|_\infty \leq 1$.

Proof. By definition,

$$\begin{aligned}
[\mathsf{ct} \cdot (1, s)]_q &= [\lfloor \Delta \cdot m \rceil + e]_q \\
&= [\lfloor (q/t) \cdot \mathsf{Flatten}(m) \rceil + e]_q \\
&= \left[\left\lfloor q \cdot \left[\frac{m}{t}\right]_1 \right\rceil + e\right]_q = \left\lfloor q \cdot \left[\frac{m}{t}\right]_1 \right\rceil + q \cdot I + e
\end{aligned}$$

for some $I \in \mathcal{R}$. Note that

$$\begin{aligned} I &= \frac{1}{q} \cdot \left(\left[\left\lfloor q \cdot \left[\frac{m}{t} \right]_1 \right\rceil + e \right]_q - \left\lfloor q \cdot \left[\frac{m}{t} \right]_1 \right\rceil - e \right) \\ &= - \left\lfloor \frac{\left\lfloor q \cdot \left[\frac{m}{t} \right]_1 \right\rceil + e}{q} \right\rceil. \end{aligned}$$

Since $\|[m/t]_1\|_\infty \leq 1/2$, we have that

$$\left\| \left\lfloor q \cdot \left[\frac{m}{t} \right]_1 \right\rceil + e \right\|_\infty \leq \frac{q}{2} + \|e\|_\infty < q.$$

Therefore, $\|I\|_\infty \leq 1$ as desired. □

If the size of the message is small, we get a stronger result.

Lemma 1. *Let $m \in \mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b) \simeq \mathcal{R}_t$ be a polynomial. If $\|m\|_\infty \leq |b|^\ell$ for some $\ell < N/k$, then $\deg(\mathsf{Flatten}(m)) \leq k(\ell+1) - 1$.*

Proof. Let $m(X) = \sum_{i=0}^{k-1} m_i X^i$, where $|m_i| \leq |b|^\ell$ for each $0 \leq i \leq k-1$. By the definition of $\mathsf{Flatten}$, we have

$$\begin{aligned} \mathsf{Flatten}(m) &= t \cdot \left[\frac{m}{t} \right]_1 = t \cdot \left(\frac{m}{t} - \left\lfloor \frac{m}{t} \right\rceil \right) = m - t \cdot \left\lfloor \frac{m}{t} \right\rceil \\ &= m(X) - (b - X^k) \cdot \left\lfloor \frac{b^{N/k-1} + b^{N/k-2} \cdot X^k + \cdots + X^{N-k}}{b^{N/k}+1} \cdot \left(\sum_{i=0}^{k-1} m_i X^i \right) \right\rceil \\ &= m(X) - (b - X^k) \cdot \left\lfloor \sum_{j=0}^{N/k-1} \sum_{i=0}^{k-1} \frac{b^{N/k-j-1} \cdot m_i}{b^{N/k}+1} \cdot X^{kj+i} \right\rceil \\ &= m(X) - (b - X^k) \cdot \alpha(X) \end{aligned}$$

where $\alpha(X) \in \mathbb{Q}[X]/(X^N + 1)$. Since $|m_i| \leq |b|^\ell$ and $|b| \geq 2$,

$$\left| \frac{b^{N/k-j-1} \cdot m_i}{b^{N/k}+1} \right| < \left| \frac{b^{N/k-j-1} \cdot b^\ell}{b^{N/k}} \right| = |b|^{\ell-j-1} \leq 1/2$$

for all $j \geq \ell$. This means that $\deg(\alpha(X)) \leq k\ell - 1$, leading to $\deg(\mathsf{Flatten}(m)) \leq k(\ell+1) - 1$ as desired. □

Theorem 6. *Let $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ be a GBFV ciphertext encrypting a message $m \in \mathcal{R}_t$, satisfying*

$$\mathsf{ct} \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e,$$

$\|m\|_\infty \leq |b|^\ell$ for some $\ell < N/k$ as an element of $\mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b) \simeq \mathcal{R}_t$ and $\|e\|_\infty < q/4$. Then we have

$$[\mathsf{ct} \cdot (1, s)]_q = \lfloor \Delta \cdot \mathsf{Flatten}(m) \rceil + q \cdot I + e$$

where $I \in \mathcal{R}$ is a small polynomial satisfying $\|I\|_\infty \leq 1$ and $\deg(I) \leq k(\ell+1) - 1$.

Proof. As in the proof of Theorem 5, we have

$$[\mathsf{ct}\cdot(1,s)]_q = \left\lfloor q\cdot\left[\frac{m}{t}\right]\right\rceil + q\cdot I + e$$

for some $I\in\mathcal{R}$. Note that

$$I = -\left\lfloor \frac{\lfloor q\cdot\left[\frac{m}{t}\right]_1\rceil + e}{q}\right\rceil.$$

As in the proof of Lemma 1, the coefficient of the term X^{kj+i} ($0\le j<N/k$, $0\le i<k$) is

$$-\left\lfloor \frac{\lfloor q\cdot b^{N/k-j-1}\cdot m_i/(b^{N/k}+1)\rceil + e_i}{q}\right\rceil = 0$$

if $j\ge \ell+1$ where $e=\sum_{i=0}^{N-1} e_i X^i$. Here we use the fact that

$$\left|q\cdot b^{N/k-j-1}\cdot m_i/(b^{N/k}+1)\right| < |b|^{\ell-j-1}\le q/4$$

and $|e_i|<q/4$. Hence, $\deg(I)\le k(\ell+1)-1$. The fact that $\|I\|_\infty\le 1$ directly follows from Theorem 5. □

Next, we consider a variant of CKKS bootstrapping that allows messages to be put in the most significant bits. Such bootstrapping can be instantiated as in [17] via the iterative discrete bootstrapping. Theorem 5 and 6, when compiled with this CKKS bootstrapping, give a conversion from GBFV to CKKS:

Algorithm 2: GBFV to CKKS Conversion

Setting: $\Delta = q/t$, $\Delta' = q$. $\mathsf{CKKS.Bootstrap}:\mathcal{R}_q^2\to\mathcal{R}_Q^2$ introduces a small error e_{BTS}.
Input : $\mathsf{ct}=(c_0,c_1)\in\mathcal{R}_q^2$ such that $\mathsf{ct}\cdot(1,s)\equiv_q \lfloor\Delta\cdot m\rceil + e$.
Output: $\mathsf{ct}_{\mathsf{out}}=(c_0',c_1')\in\mathcal{R}_q^2$ such that $[\mathsf{ct}_{\mathsf{out}}\cdot(1,s)]_q = \lfloor q\cdot\mathsf{Flatten}(m)\rceil + q\cdot t\cdot I + e'$, $\|I\|_\infty\le 1$.

$\mathsf{ct}_{\mathsf{out}}\leftarrow t\cdot\mathsf{CKKS.Bootstrap}(\mathsf{ct})$;
return $\mathsf{ct}_{\mathsf{out}}$

Theorem 7 (GBFV to CKKS). *Let* $\mathsf{ct}=(c_0,c_1)\in\mathcal{R}_q^2$ *be a GBFV ciphertext encrypting a message* $m\in\mathcal{R}_t$. *That is,*

$$\mathsf{ct}\cdot(1,s)\equiv_q \lfloor\Delta\cdot m\rceil + e$$

for a scaling factor $\Delta=q/t$ *and some small error* $e\in\mathcal{R}$ *satisfying* $\|e\|_\infty\ll q/2$. *Let* $\mathsf{CKKS.Bootstrap}:\mathcal{R}_q^2\to\mathcal{R}_Q^2$ *be a CKKS bootstrapping accepting any input plaintext in* $[-q/2,q/2)^N$. *Then* $t\cdot\mathsf{CKKS.Bootstrap}(\mathsf{ct})\in\mathcal{R}_Q^2$ *is a CKKS*

ciphertext encrypting $\mathsf{Flatten}(m)+t\cdot I$ *with scaling factor* q *where* $I \in \mathcal{R}$ *is a small polynomial satisfying* $\|I\|_\infty \le 1$. *Furthermore, if* $\|m\|_\infty \le |b|^\ell$ *as an element of* $\mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b)$ *for some* $\ell < N/k$, *then* $\deg(\mathsf{Flatten}(m)) \le k(\ell+1)-1$ *and* $\deg(I) \le k(\ell+1)-1$.

Proof. By Theorem 5, we observe that

$$[\mathsf{ct}\cdot(1,s)]_q = \lfloor \Delta \cdot \mathsf{Flatten}(m) \rceil + q \cdot I + e$$

for some $I \in \mathcal{R}$ satisfying $\|I\|_\infty \le 1$. Hence, ct can be regarded as a CKKS ciphertext encrypting a plaintext $\lfloor \Delta \cdot \mathsf{Flatten}(m) \rceil + q \cdot I + e$. In the case where $\|m\|_\infty \le |b|^\ell$, we have $\deg(I) \le k(\ell+1)-1$ in addition, by Theorem 6. Next, the CKKS bootstrapping $\mathsf{CKKS.Bootstrap}$ raises the modulus while approximately preserving the underlying plaintext, thereby giving ct' satisfying

$$[\mathsf{ct}'\cdot(1,s)]_Q = \lfloor \Delta \cdot \mathsf{Flatten}(m) \rceil + q \cdot I + e + e_{\mathsf{BTS}}$$

where e_{BTS} is the bootstrapping error. When multiplied by t, we have

$$[t\cdot\mathsf{ct}'\cdot(1,s)]_Q = q\cdot(\mathsf{Flatten}(m)+t\cdot I) + t\cdot(e+e_{\mathsf{BTS}}) + e_{\mathsf{Round}}$$

for some small rounding error as in Section 3. Since $t\cdot(e+e_{\mathsf{BTS}})+e_{\mathsf{Round}}$ is small, the ciphertext $t\cdot\mathsf{ct}'$ can be regarded as a CKKS ciphertext with scaling factor q and message $\mathsf{Flatten}(m)+t\cdot I$. □

The backward conversion, CKKS to GBFV, can be defined more easily through a simple modulus switching.

Theorem 8 (CKKS to GBFV). *Let* $\mathsf{ct} = (c_1, c_1) \in \mathcal{R}^2_{qq'}$ *be a CKKS ciphertext encrypting a message* $\mathsf{Flatten}(m) + t\cdot I$ *where* $m \in \mathcal{R}_t$, $I \in \mathcal{R}$ *satisfying* $\|I\|_\infty \le B$, $q' \simeq q$, *and* $B = O_q(1)$. *That is,*

$$[\mathsf{ct}\cdot(1,s)]_{qq'} = q\cdot(\mathsf{Flatten}(m)+t\cdot I)+e$$

for some small error $e \in \mathcal{R}$. *Then* $\mathsf{CKKS.RS}_{q'}(\lfloor q'/t \rceil \cdot \mathsf{ct}) \in \mathcal{R}^2_q$ *can be regarded as a GBFV ciphertext encrypting* $m \in \mathcal{R}_t$.

Proof. We first observe that

$$\begin{aligned}
\lfloor q'/t \rceil \cdot \mathsf{ct}\cdot(1,s) &\equiv_{qq'} \lfloor q'/t \rceil \cdot q \cdot (\mathsf{Flatten}(m)+t\cdot I) + \lfloor q'/t \rceil \cdot e \\
&\equiv_{qq'} \lfloor q'/t \rceil \cdot q \cdot \mathsf{Flatten}(m) + (q'/t)\cdot q\cdot t\cdot I + \lfloor q'/t \rceil \cdot e + e' \\
&\equiv_{qq'} \lfloor q'/t \rceil \cdot q \cdot \mathsf{Flatten}(m) + e''
\end{aligned}$$

where $e' = [q'/t]_1 \cdot q \cdot t \cdot I$ and $e'' = \lfloor q'/t \rceil \cdot e + e'$. After rescaling by q', the resulting ciphertext $\mathsf{ct}' = \mathsf{CKKS.RS}_{q'}(\lfloor q'/t \rceil \cdot \mathsf{ct})$ satisfies

$$\mathsf{ct}'\cdot(1,s) \equiv_q \lfloor (q/t)\cdot\mathsf{Flatten}(m) \rceil + e''' + \lfloor e''/q' \rceil + e_{\mathsf{RS}}$$

where

$$\begin{aligned} e''' &= \lfloor (q/t) \cdot \mathsf{Flatten}(m) \rceil - \lfloor \lfloor q'/t \rceil \cdot q \cdot \mathsf{Flatten}(m)/q' \rceil \\ &\simeq (q/t) \cdot \mathsf{Flatten}(m) \rceil - \lfloor q'/t \rceil \cdot q \cdot \mathsf{Flatten}(m)/q' \\ &= [q'/t]_1 \cdot (q/q') \cdot \mathsf{Flatten}(m) = O_q(1) \end{aligned}$$

and e_{RS} is the rescaling error. Since

$$\lfloor (q/t) \cdot \mathsf{Flatten}(m) \rceil \equiv_q \lfloor (q/t) \cdot m \rceil$$

by definition, it remains to prove that the error $\lfloor e''/q' \rceil$ is small. For this, we check that

$$\begin{aligned} \lfloor e''/q' \rceil &\simeq e''/q' = (\lfloor q'/t \rceil \cdot e + e')/q' \\ &\simeq e/t + e'/q' = e/t + \lfloor q'/t \rceil_1 \cdot q \cdot t \cdot I/q' \\ &= e/t + (q/q') \cdot (\lfloor q'/t \rceil_1 \cdot t \cdot I) = O_q(1) \end{aligned}$$

which is small. This finishes the proof. □

Corollary 2 (Conversion Correctness). *Let $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ be a GBFV ciphertext encrypting a message $m \in \mathcal{R}_t$. If we apply the CKKS-to-GBFV conversion $: \mathcal{R}_q^2 \to \mathcal{R}_{qq'}^2$ and the GBFV-to-CKKS converison $: \mathcal{R}_{qq'}^2 \to \mathcal{R}_q^2$, the resulting GBFV ciphertext encrypts the same message m as before.*

Proof. The proof directly follows from Theorem 5 and 6. □

4.2 Applications

In this subsection, we discuss two applications of our conversion framework, arbitrary function evaluation and (approximate) modular reduction.

Arbitrary Function Evaluation. The first application of our conversions is the arbitrary function evaluation. At a high level, we use the GBFV-to-CKKS conversion as an approximate digit decomposition, thereby utilizing multivariate interpolation rather than univariate interpolation (which is the case of GBFV).

Lemma 2. *Let $m \in \mathcal{R}_t$. Then $\|\mathsf{Flatten}(m)\|_\infty \leq (|b|+1)/2$.*

Proof. By definition,

$$\begin{aligned} \|\mathsf{Flatten}(m)\|_\infty &= \left\| t \cdot \left[\frac{m}{t} \right]_1 \right\|_\infty \\ &= \left\| (X^k - b) \cdot \left[\frac{m}{t} \right] \right\|_\infty \\ &\leq (|b|+1) \cdot \left\| \left[\frac{m}{t} \right] \right\|_\infty \leq (|b|+1)/2. \end{aligned}$$

The first inequality is from the triangular inequality. □

Corollary 3. *Let* $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ *be a GBFV ciphertext encrypting a message* $m \in \mathcal{R}_t$*, and* ct' *be the output of the GBFV-to-CKKS conversion on* ct*. Let* $m' = \mathsf{Flatten}(m) + t \cdot I \in \mathcal{R}$ *be the underlying message as in Theorem 5, then we have* $\|m'\|_\infty \leq 3(|b|+1)/2$*.*

Proof. By Theorem 5 and Lemma 2, we have

$$\begin{aligned}\|m'\|_\infty &= \|\mathsf{Flatten}(m) + t \cdot I\|_\infty \\ &= \|\mathsf{Flatten}(m) + (X^k - b) \cdot I\|_\infty \\ &\leq \|\mathsf{Flatten}(m)\|_\infty + (|b|+1) \cdot \|I\|_\infty \leq 3(|b|+1)/2.\end{aligned}$$

This finishes the proof. □

When the plaintext modulus $p = b^{N/k} + 1$ is a prime, the naive GBFV approach needs to evaluate a univariate interpolation of degree p. Instead, we may first decompose the GBFV ciphertext into digits by using our GBFV-to-CKKS conversion, and perform a multivariate interpolation of individual degree $\leq 2\|m'\|_\infty + 1 \leq 3|b| + 4$. The multiplicative depth reduces from $\log_2(b^{N/k}+1) \simeq (N/k) \cdot \log_2(|b|)$ to $\log_2(N/k) + \log_2(3|b|+4)$, achieving logarithmic asymptotic in N/k. In addition, one may use the discrete CKKS framework [12, 3], which supports more efficient bootstrapping than GBFV in large parameters.

Modular Reduction. One of the important applications of GBFV is large integer arithmetic, as we can choose a sufficiently large plaintext modulus $b^{N/k}+1$ and simulate integer arithmetic inside the ring $\mathbb{Z}_{b^{N/k}+1}$. In addition, one may want to simulate $\mathbb{Z}_u$ arithmetic for $u < b^{N/k} + 1$, as in the recent work on large integer computations with discrete CKKS [4]. Our construction leverages GBFV and CKKS to handle $\mathbb{Z}_u$ arithmetic inside $\mathbb{Z}_{b^{N/k}+1}$.

Lemma 3. *Let* $u < |b|^\ell$ *be a positive integer for some* $\ell < N/k$*. Let* $\Omega = \sum_{i=0}^{\ell-1} \omega_i \cdot b^i$ *be a (redundant)* b*-digit representation of* $\Omega \in \mathbb{Z}$*, where* $|w_i| \leq B$ *for each* i*. Then*

$$\Omega' = \sum_{i=0}^{\ell-1} w_i \cdot [b^i]_u$$

satisfies $\Omega' \equiv_u \Omega$ *and*

$$|\Omega'| \leq Bu\ell/2.$$

Proof. By definition of Ω',

$$\Omega' = \sum_{i=0}^{\ell-1} w_i \cdot [b^i]_u \equiv_u \sum_{i=0}^{\ell-1} w_i \cdot b^i.$$

For the upper bound of Ω',

$$|\Omega'| \leq \sum_{i=0}^{\ell-1} B \cdot (u/2) = Bu\ell/2.$$

This finishes to proof. □

We apply the algorithm of Lemma 3 to the output of Algorithm 2. Let $u < |b|^\ell$ for some $\ell < N/k$. Let $\Phi(X) = \sum_{i=0}^{N-1} \varphi_i \cdot X^i \in \mathcal{R}$ satisfies $|\varphi_i| \leq B$ for some $B > 0$ and $\deg(\Phi) \leq k(\ell+2)-1$. Φ corresponds to

$$\check{\Phi}(X) = \sum_{i=0}^{k-1} \left(\sum_{j=0}^{\ell+1} \varphi_{kj+i} \cdot b^j \right) \cdot X^i = \sum_{i=0}^{k-1} \Omega_i \cdot X^i \in \mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b).$$

We may precompute $[b^i]_u$ for $0 \leq i \leq \ell+1$ in advance and represent it as a b-digit form

$$[b^i]_u = \sum_{j=0}^{\ell-1} \beta_{ij} \cdot b^j.$$

Next, we replace b with X^k and get

$$\Phi'(X) = \sum_{i=0}^{k-1} \left[\sum_{j=0}^{\ell+1} \left(\varphi_{kj+i} \cdot \left(\sum_{v=0}^{\ell-1} \beta_{jv} \cdot X^{kv} \right) \right) \cdot X^i \right] \tag{1}$$

$$= \sum_{i=0}^{k-1} \left[\sum_{v=0}^{\ell-1} \left(\sum_{j=0}^{\ell+1} \varphi_{kj+i} \cdot \beta_{jv} \right) \cdot X^{kv+i} \in \mathcal{R} \right]. \tag{2}$$

which is an approximate modular reduction of Φ with respect to our encoding.

As we described our algorithm in terms of plaintext polynomials in $\mathcal{R}$, it remains to translate it into ciphertext arithmetic. In CKKS, such an operation on plaintexts can be represented as a homomorphic linear transformation, which is very efficient. Let $\mathsf{LinTrans}$ be the corresponding homomorphic linear transformation. The correctness of the algorithm, i.e., that applying Lemma 3 homomorphically gives the correct modular reduction, mainly relies on the following Lemma.

Lemma 4. *Let $m \in \mathcal{R}_t$ satisfies $\|m\|_\infty \leq |b|^\ell$ as an element of $\mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b)$ for some $\ell < N/k - 4$. Let $\tilde{m} \in \mathbb{Z}[X]/(X^k - b)$ be a natural embedding of m via the signed embedding $\mathbb{Z}_{b^{N/k}+1} = [-(b^{N/k}+1)/2, (b^{N/k}+1)/2) \hookrightarrow \mathbb{Z}$. Let $m' = \mathsf{Flatten}(m) + t \cdot I \in \mathcal{R}$ satisfies $I \in \mathcal{R}$, $\|I\|_\infty \leq 1$, and $\deg(I) \leq k(\ell+1)-1$. Let $\tilde{m'} \in \mathbb{Z}[X]$ be a natural embedding of m' via $\mathcal{R} = \mathbb{Z}[X]/(X^N+1) \hookrightarrow \mathbb{Z}[X]$. Then we have*

$$[\tilde{m'}]_t = \tilde{m}$$

as an element of $\mathbb{Z}[X]/(X^k - b)$.

Proof. We write $m' \in \mathcal{R}$ as

$$m'(X) = \sum_{i=0}^{N-1} \mu_i X^i = \sum_{i=0}^{k-1} \left(\sum_{j=0}^{N/k-1} \mu_{kj+i} X^{kj} \right) X^i$$

which gives $\tilde{m'}(X) = \sum_{i=0}^{N-1} X^i$. Then we have

$$[\tilde{m'}]_t = [\tilde{m'}]_{X^k - b} = \sum_{i=0}^{k-1} \left(\sum_{j=0}^{N/k-1} \mu_{kj+i} \cdot b^j \right) \cdot X^i \in \mathbb{Z}[X]/(X^k - b).$$

Since

$$\deg(m') \leq \max(\deg(\mathsf{Flatten}(m)), \deg(t) + \deg(I)) \leq k(\ell + 2) - 1$$

and

$$\|m'\|_\infty \leq \|\mathsf{Flatten}(m)\|_\infty + (|b| + 1) \cdot \|I\|_\infty \leq 3(|b| + 1)/2$$

(as in the proof of Corollary 3), we have that

$$\begin{aligned}
\|[\tilde{m'}]_t\|_\infty &\leq \max_i \left(\sum_{j=0}^{N/k-1} |\mu_{kj+i}| \cdot |b|^j \right) \\
&= \max_i \left(\sum_{j=0}^{\ell+1} |\mu_{kj+i}| \cdot |b|^i \right) \\
&\leq \sum_{j=0}^{\ell} 3(|b| + 1) \cdot |b|^i / 2 = \frac{3(|b| + 1) \cdot (|b|^{\ell+1} - 1)}{2(|b| - 1)} \\
&\leq |b|^{\ell+4} < (b^{N/k} + 1)/2 \quad (\because |b| \geq 2, \ell < N/k - 4).
\end{aligned}$$

In addition, we check that

$$\begin{aligned}
[[\tilde{m'}]_t]_{b^{N/k}+1} &= [[\tilde{m'}]_t]_{X^N+1} \\
&= [[\tilde{m'}]_{X^N+1}]_t \\
&= [\mathsf{Flatten}(m) + t \cdot I]_t \equiv_t m.
\end{aligned}$$

Therefore, $[\tilde{m'}]_t \in \mathbb{Z}[X]/(X^k - b)$ is a representation of m with normalized coefficients, meaning that it should be equal to $\tilde{m}$. □

Now we are ready to describe our approximate modular reduction leveraging CKKS, which we describe in Algorithm 3.

Theorem 9 (Modular Reduction). *Let* $\mathsf{ct} = (c_0, c_1)$ *be a GBFV ciphertext encrypting* $m \in \mathcal{R}_t$ *which satisfies* $\|m\|_\infty < |b|^\ell$ *as an element of* $\mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b)$ *for some* $\ell < N/k - 4$*. Then the output of Algorithm 3 is a GBFV ciphertext encrypting* $m' = m + u \cdot J \in \mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b)$ *for some small integral* J *satisfying* $\|J\|_\infty \leq 3(|b| + 1)u(\ell + 2)/2$.

Proof. After GBFV-to-CKKS, we have a CKKS ciphertext encrypting $m' = \mathsf{Flatten}(m) + t \cdot I$ with scaling factor q where $I \in \mathcal{R}$ is a small polynomial satisfying $\|I\|_\infty \leq 1$ and $\deg(m') \leq k(\ell + 2) - 1$ (from Theorem 6). In addition, Corollary 3 gives the upper bound $\|m'\|_\infty \leq 3(|b| + 1)/2$.

Algorithm 3: Approximate Modular Reduction through CKKS

Setting: $\Delta = q/t$. $q' \simeq q$ and $Q \gg q$. $\mathsf{GBFV\text{-}to\text{-}CKKS} : \mathcal{R}_q^2 \to \mathcal{R}_Q^2$ and $\mathsf{CKKS\text{-}to\text{-}GBFV} : \mathcal{R}_{qq'}^2 \to \mathcal{R}_q^2$ are conversions from GBFV to CKKS and vice versa, as in the previous subsection. The integer $u > 0$ which we want to take the modular reduction. Let $\mathsf{LinTrans}$ be the homomorphic linear transformation as defined previously.

Input : $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_q^2$ such that $\mathsf{ct} \cdot (1, s) \equiv_q \lfloor \Delta \cdot m \rceil + e$, where $\|m\|_\infty < |b|^\ell$ as an element of $\mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b)$ for some $\ell < N/k - 4$.

Output: $\mathsf{ct}_{\mathsf{out}} = (c_0', c_1') \in \mathcal{R}_q^2$ such that $\mathsf{ct}_{\mathsf{out}} \cdot (1, s) \equiv_q \lfloor \Delta \cdot m' \rceil + e'$ for some $m \in \mathcal{R}_t$ such that $m \equiv_u m'$ as an element of $\mathbb{Z}[X]/(X^k - b)$ and $\|m'\|_\infty \ll \|m\|_\infty$.

1 $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{CKKS\text{-}to\text{-}GBFV} \circ \mathsf{LinTrans} \circ \mathsf{GBFV\text{-}to\text{-}CKKS}(\mathsf{ct})$;

2 **return** $\mathsf{ct}_{\mathsf{out}}$

Next, applying $\mathsf{LinTrans}$ performs Lemma 3 in a vectorized manner, thereby homomorphically getting approximate modular reduction with upper bound $(3(|b|+1)/2) \cdot u \cdot (\ell + 2)$. Here, Lemma 4 ensures that there is no overflow (i.e. wrapping around modulo $b^{N/k} + 1$) so that the correctness is guaranteed.

Finally, we convert the mod-reduced CKKS ciphertext into a GBFV ciphertext. The input CKKS ciphertext can be regarded as a valid encryption of $\mathsf{Flatten}(m') + t \cdot I'$ for some m' and I', where $m' \equiv_u m$ as an element of $\mathbb{Z}_{b^{N/k}+1}[X]/(X^k - b)$ and $\|m'\|_\infty \leq 3(|b|+1)u(\ell+2)/2$. To be correctly be converted to a GBFV ciphertext, $\|I\|_\infty$ needs to be bounded (see Theorem 8). To check this, we revisit Line (2) where we computed Φ' for modular reduction. The upper bound is computed as

$$\begin{aligned}
\|\Phi'\|_\infty &= \max_{i,v} \left(\sum_{j=0}^{\ell+1} \varphi_{kj+i} \cdot \beta_{jv} \right) \\
&\leq (\ell+2) \cdot \max_i |\varphi_i| \cdot \max_j |\beta_j| \\
&\leq (\ell+2) \cdot (3(|b|+1)/2) \cdot (|b|/2) \\
&= |b|(3|b|+1)(\ell+2)/4 = O_q(1)
\end{aligned}$$

which is small. This finishes the proof. □

5 Experimental Results

We provide proof-of-concept implementation results for our GBFV bootstrapping in Section 3. Our code is built upon the golang lattigo library [1] and all our experiments are taken single-threaded over an Apple M4 Max processor with 128GB of RAM running macOS Sequoia 15.4.1. All of our parameters satisfy 128 bits of security according to [5]. All the experiments are measured after at least 20 iterations.

5.1 Bootstrapping GBFV

We first describe the FHE parameter set used in the experiments. As in Section 3, we denote q for the GBFV modulus which is the input modulus for $\mathsf{CKKS.Bootstrap} : \mathcal{R}_q^2 \to \mathcal{R}_{qq'}^2$, and denote qq' for the output modulus of $\mathsf{CKKS.Bootstrap}$. We choose q to be a product of a 60-bit NTT prime and a 45-bit NTT prime, and q' to be a product of two 45 bit primes. Using META-BTS [2], this parameter supports $\simeq$ 80-bit precision CKKS bootstrapping within $\simeq$ 18 seconds. The detailed information is written in Table 1.

Table 1. The FHE parameter set we used for the experiments. Here N denotes the ring dimension, QP denotes the maximum switching key modulus, $(h, \tilde{h})$ denote the dense and spare Hamming weights for the sparse secret encapsulation [5], respectively, q denotes the GBFV modulus (= CKKS bootstrapping input modulus), and qq' denotes the CKKS bootstrapping output modulus.

$\log N$	$\log QP$	$(h, \tilde{h})$	$\log q$	$\log q'$
16	1288	$(192, 32)$	$60 + 45$	45×2

We test our main bootstrapping algorithm in Algorithm 1 by using the CKKS bootstrapping as a black box. In particular, we change the combination of b, k for $t(X) = X^k - b$, providing various parameter choices. The bootstrapping time and the denoising factor remained almost the same as we rely on the same CKKS bootstrapping. We first fix $b = 2$ and change k, which provides a different plaintext modulus $b^{N/k}+1$ and maximum number of slots k, introducing a trade-off between the modulus size and the number of slots. Second, we fix $k = 1$ (i.e. bootstrapping CLPX [9]) and change b, which gives different plaintext moduli $b^N + 1$. Such choices support extremely large moduli. For instance, for $b = 32$ and $N = 2^{16}$, the plaintext modulus is $\log_2(32^{2^{16}} + 1) \simeq 5 \cdot 2^{16}$ bits. The detailed results are illustrated in Table 2.

5.2 Comparison with Prior Works

First, we concretely compare our results with the only existing GBFV bootstrapping in [14]. As they only provide plaintext modulus $p = 2^{16} + 1$ case, we compare it with our result for plaintext modulus p. In [14], they discuss two optimization strategies for latency and throughput. We compared our results with both types. When compared to $N = 2^{14}$ (resp. $N = 2^{15}$) implementations of [14], our work achieves a similar denoising factor (resp. number of slots). However, in either case, our method loses significantly in terms of latency or denoising factor, respectively. Therefore, the bootstrapping of [14] remains competitive for small modulus like $p = 2^{16} + 1$.

When it comes to larger modulus, the GBFV bootstrapping in [14] performs significantly worse than our method. The key reason is that they use BFV linear transformation as a subroutine, which means that at some point the plaintext

Table 2. GBFV Bootstrapping Experimental Results. Here $t(X) = X^k - b$, T_{BTS} denotes the GBFV bootstrapping time, and the denoising factor denotes $\log(B_1/B_2)$ where B_1 and B_2 are the input and output error upper bound, respectively.

b	k	T_{BTS} (sec)	Denoising Factor (bits)
2	1	18.4	77.1
	4	18.5	77.3
	16	18.4	77.3
	64	18.3	77.3
	256	18.5	77.2
	1024	18.4	77.2
	4096	18.4	77.1
4	1	18.4	77.4
8		18.4	74.7
16		18.4	74.1
32		17.9	74.1
256		18.4	70.8

Table 3. Comparison with [14] on modulus $p = 2^{16} + 1$. The figures are borrowed from [14, Table 1 and 3], including both single and batch bootstrappings.

	N	# slots	T_{BTS} (sec)	Denoising Factor (bits)
[14]	2^{14}	1024	1.94	124
		2048	1.98	115
		4096	1.98	90
		8192	2.01	38
	2^{15}	32768	13.24	374
Ours	2^{16}	32768	18.4	70.1

space becomes $\mathbb{Z}_{b^{N/k}+1}$ as a BFV ciphertext. In particular, if N/k is very large (e.g. CLPX [9] where $k = 1$), the modulus consumption during BFV linear transformation is $O(b^{N/k} + 1)$ which is not affordable for practical RLWE dimensions. In this regard, our bootstrapping is the first GBFV bootstrapping that can support an arbitrary plaintext modulus.

When the target is achieving large precision bootstrapping within a single RLWE ciphertext, one may compare our method with the state-of-the-art high precision bootstrapping in [4]. Asymptotically, both works support the same maximum precision $O(N)$ bits. In practice, our method supports much larger moduli as we do not lose modulus from embedding smaller moduli into larger moduli. Concretely, the maximum reported plaintext modulus in [4] is 7 679 bits, and the theoretical maximum is $\simeq 30\,000$ bits. On the other hand, we support up to $2^{19} = 524\,288$ bits[9] which is more than an order of magnitude better, within the same RLWE dimension $N = 2^{16}$. Nevertheless, this does not mean that our

[9] In the case of $b = 256$ and $k = 1$, as in Table 2. $\log_2(b^{N/k} + 1) \simeq (N/k) \cdot \log_2(b) = 2^{16} \cdot 8 = 2^{19}$.

scheme outperforms [4] in all aspects: they do support arbitrary moduli while our modulus choice is restricted to cyclotomic moduli.

Acknowledgements

We thank Robin Geelen for answering a question on GBFV encoding. This work was funded by NSF, DARPA, and the Simons Foundation. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. The author of this work was partially supported by the Stanford Graduate Fellowship in Science and Engineering as a David Cheriton Fellow.

References

1. Lattigo v6. Online: https://github.com/tuneinsight/lattigo (Aug 2024), ePFL-LDS, Tune Insight SA
2. Bae, Y., Cheon, J.H., Cho, W., Kim, J., Kim, T.: META-BTS: Bootstrapping precision beyond the limit. In: CCS (2022)
3. Bae, Y., Kim, J., Stehlé, D., Suvanto, E.: Bootstrapping small integers with CKKS. In: ASIACRYPT (2024)
4. Boneh, D., Kim, J.: Homomorphic encryption for large integers from nested residue number systems. Cryptology ePrint Archive, Paper 2025/346 (2025)
5. Bossuat, J.P., Troncoso-Pastoriza, J., Hubaux, J.P.: Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation. In: ACNS (2022)
6. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: CHIMERA: Combining ring-LWE-based fully homomorphic encryption schemes. J. Math. Crypt. (2020)
7. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: CRYPTO (2012)
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS (2012)
9. Chen, H., Laine, K., Player, R., Xia, Y.: High-precision arithmetic in homomorphic encryption. In: CT-RSA (2018)
10. Cheon, J.H., Cho, W., Kim, J., Stehlé, D.: Homomorphic multiple precision multiplication for CKKS and reduced modulus consumption. In: CCS (2023)
11. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: EUROCRYPT (2018)
12. Drucker, N., Moshkowich, G., Pelleg, T., Shaul, H.: BLEACH: Cleaning errors in discrete computations over CKKS. J. Cryptol. (2024)
13. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144 (2012), https://eprint.iacr.org/2012/144
14. Geelen, R., Vercauteren, F.: Fully homomorphic encryption for cyclotomic prime moduli. In: EUROCRYPT (2025)
15. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC (2009)
16. Kim, J.: Efficient homomorphic integer computer from CKKS. Cryptology ePrint Archive, Paper 2025/066 (2025), https://eprint.iacr.org/2025/066
17. Kim, J., Noh, T.: Modular reduction in CKKS. Cryptology ePrint Archive, Paper 2024/1638 (2024)

18. Kim, J., Seo, J., Song, Y.: Simpler and faster BFV bootstrapping for arbitrary plaintext modulus from CKKS. In: CCS (2024)
19. Lu, W.j., Huang, Z., Hong, C., Ma, Y., Qu, H.: PEGASUS: Bridging polynomial and non-polynomial evaluations in homomorphic encryption. In: S&P (2021)