On Phase Decompositions with Applications to Digit Decomposition

Leonard Schild, Aysajan Abidin and Bart Preneel
COSIC, KU Leuven
Leuven, Belgium
Email: firstname.lastname [at] kuleuven.be

Abstract

Fully homomorphic encryption schemes enable arbitrary computation on encrypted data. Although the first candidate were considered too computationally inefficient, modern cryptosystems are now featured in practical deployments. Yet, despite a plethora of advances, many non-polynomial operations remain computationally challenging.

In this work, we investigate radix decomposition, i.e., the homomorphic conversion of an LWE ciphertext with a large plaintext message into encryptions of its digits. Such decompositions are a core building block for large-precision computation in FHEW/TFHE-like schemes, for instance for sign evaluation, comparisons, and large lookup tables. To this end, we introduce phase decomposition: decomposing the coefficients of an LWE ciphertext itself in a radix α yields small pseudo-LWE samples whose phases recombine linearly to the phase of the original ciphertext. For keys of bounded Hamming weight, these phases are small and can be made non-negative, and the base-α digits of the original phase follow from them through a simple carry recurrence. We evaluate this recurrence entirely in the exponent of an RLWE accumulator, using a new exact homomorphic truncation in the exponent based on automorphisms. The resulting digit decomposition requires a single blind rotation per digit, asymptotically half as many as the state of the art, introduces no modulus-switching error between iterations, and supports input moduli larger than the ring modulus. The same tools yield functional bootstrapping over the full ring $\mathbb{Z}_{2N}$, the evaluation of arbitrary, not necessarily negacyclic, functions on small plaintext spaces, and a scheme-independent digit decomposition for SIMD schemes based on parallel-prefix addition.

Index Terms

Fully Homomorphic Encryption, Digit Decomposition,Carry Addition, FHE, Bootstrapping

I. Introduction

Fully Homomorphic Encryption (FHE) enables computation directly on encrypted data, ensuring that sensitive information remains confidential throughout the processing pipeline. Since Gentry's seminal work in 2009 introducing the first FHE scheme [1], significant progress has been made to improve efficiency, noise management, and bootstrapping performance [2]–[4]. FHE have evolved into practical cryptographic tools for privacy-preserving analytics, federated learning, and secure cloud computation [5].

Accumulator-based schemes, such as FHEW [6] and TFHE [2], [3], have emerged as particularly efficient for Boolean and small-integer computations. Their efficiency stems from fast bootstrapping procedures that refresh ciphertext noise, enabling sub-millisecond gate evaluation. Although these schemes operate over small plaintext space–typically messages of length 4-5 bits–the mechanism of programmable bootstrapping allows them to evaluate arbitrary functions on encrypted inputs, thereby enabling computation of complex functions despite their small plaintext space [7]. An often overlooked feature of programmable bootstrapping is an implicit rescaling functionality, which allows such encrypted data to be lifted into larger plaintext space. Such rescaling is the primary mechanism enabling large-precision homomorphic digit decomposition procedures, which expresses encrypted values in a structured radix representation.

In this work, we revisit digit decomposition from a different perspective. We introduce phase splitting or phase decomposition, a lightweight technique that decomposes the encrypted phase into structured components that need not correspond to canonical radix digits. Although the space of resulting "pseudo-digits" is not isomorphic or even strictly related to the original ciphertext space, we show that obtaining the canonical radix digits becomes an efficient step from this perspective. Our approach asymptotically halves the number of blind rotations compared to the state of the art [8]. Beyond these performance gains, phase splitting naturally enables an expanded plaintext domain: we demonstrate that the plaintext modulus can be increased by a large factor while incurring only a doubling of computational complexity and allowing re-use of parameters for small spaces. This trade-off significantly enhances the expressive capacity of accumulator-based FHEW-like schemes without proportionally increasing the cost. While recent works on functional and generalized bootstrapping provide alternative means of enlarging plaintext space [9], our approach distinguishes itself through its conceptual simplicity, modularity, and compatibility with existing accumulator structures. Phase splitting requires no new cryptographic assumptions and integrates cleanly into current system designs, providing a practical and implementation-friendly path toward supporting larger plaintext domains in FHEW-style schemes.

Our contributions.

- *Digit-decomposition in the exponent*. We introduce a decomposition methodology which is carried out via exact arithmetic in the exponent of an encrypted polynomial. By enforcing the use of exact arithmetic, we remove traditional constraints regarding rounding errors related to modulus switching or other operations. Our approach is twice as fast asymptotically

and may be used under weaker requirements than prior work, including allowing the use of a ciphertext modulus exceeding the FHEW ring modulus.
- *Large plaintext space homomorphic function evaluation*. Based on the digit-decomposition methodology, we demonstrate how to extend the plaintext space for functional bootstrapping in which a negacylic function $f(x) = -f(x+N) \pmod{2N}$ is evaluated homomorphically. Conventionally, an encoding $\bar{m} \in Z_{2N}$ follows $\bar{m} = \frac{2N}{t}m + e, m \in \mathbb{Z}_t$ where m is the true encrypted message and e the LWE noise. Hence, it was not possible to treat $\mathbb{Z}_{2N}$ to be the message space as the error terms had to be taken into account. In this work, we show how to increase the exploitable space to the full $\mathbb{Z}_{2N}$
- *Half-ring, full-domain bootstrapping*. Prior works investigated how to eliminate the restriction that functions need be negacyclic during bootstrapping. As a final contribution, we show that applying our large plaintext space functional bootstrapping technique to a plaintext space $\mathbb{Z}_t$ with small size $t < 2N$ immediately enables us to evaluate not just negacyclic, but arbitrary functions.

A. Related Work

Digit decomposition in FHEW/TFHE. Liu, Micciancio, and Polyakov [8] decompose a large-precision LWE ciphertext from the least significant digit upwards. Each iteration clears one digit with a homomorphic floor function based on two bootstrappings and then switches the modulus down. To the best of our knowledge, this is the state of the art, and we compare our method to it in detail in Section IV-B. Head Start [10] proceeds in the opposite direction, from the most significant digit downwards. Each iteration extracts the leading bits with a single bootstrapping. The extracted value may be off by one, and this error is compensated in the next iteration. The outputs therefore form a redundant representation of the message, from which canonical digits are obtained by additional carry propagation. We discuss it further in Section IV-B.

High-precision functional bootstrapping. Programmable bootstrapping in FHEW/TFHE is limited to small plaintext spaces and to negacyclic functions. Full-domain functional bootstrapping removes the latter restriction [7], [11], [12]. To enlarge the plaintext space, other works increase the ring dimension [13] or start from digit-decomposed inputs [14]. The recent Meta-PBS [9] pursues goals similar to our full-ring bootstrapping: it evaluates negacyclic functions with a plaintext modulus of up to $2N$ on a single LWE ciphertext, without enlarging the ring dimension. It splits the input into a quotient and remainders that encode the modulus-switching error and processes them from the most significant part downwards, refining the exponent over several blind rotations. Section III-B relates it to our approach.

Digit decomposition in SIMD schemes. Cheon et al. [15] recently proposed a parallel digit decomposition for CKKS that describes each digit position by a window of two neighboring digits and computes all carries with a parallel-prefix scan. Our SIMD digit decomposition (Section III-A2) has the same structure but obtains the digit pairs exactly from a phase decomposition and only assumes an interface for function evaluation [16]. Related conversions between arithmetic and digit or bit representations are also central to recent work on homomorphic arithmetic logic units [17]–[20].

Ciphertext decompositions. The idea closest to phase decomposition is the homomorphic Euclidean division of Cheon, Cho, Kim, and Stehlé [21]. It splits a CKKS ciphertext into a quotient and a remainder ciphertext. Since decryption is linear, the two decryptions recombine to the original one, and since the remainder has small coefficients and the key is small, the remainder decrypts to a small value. Cheon et al. use this decomposition to reduce the modulus consumption of homomorphic multiplication; Meta-PBS [9] builds on it as well. Phase decomposition starts from the same observation but serves a different purpose. We iterate the split into a full radix decomposition of an LWE ciphertext, make the phases of all pieces non-negative and exactly bounded, and recover the digits of the original phase from them.

B. Notation

Given $a, b \in \mathbb{Z}$, let $[a, b]$ denote the set $\{a, a+1, \ldots, b\}$, and let $[a]$ denote $[1, a]$. For an integer q, let $\mathbb{Z}_q$ denote the ring of integers modulo q. $x \pmod{q}$ or $[x]_q$ will denote the modular reduction by q into $[0, q) \cap \mathbb{Z}$. For a power-of-two N, let $\mathcal{K} = \mathbb{Q}[X]/(X^N + 1)$, $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ and $\mathcal{R}_q = \mathcal{R}/(q\mathcal{R})$. We use lowercase bold such as $\mathbf{u}$ for (row) vectors and uppercase bold such as $\mathbf{U}$ for matrices. The inner product between two vectors $\mathbf{u}$ and $\mathbf{v}$ is denoted as $\langle \mathbf{u}, \mathbf{v} \rangle$. We use notations such as $\mathbf{b}$ or $b(X)$ to denote elements of a polynomial ring. The operator $\lfloor \cdot \rceil (\lfloor \cdot \rfloor)$ denotes rounding (resp. flooring), which extends coefficient-wise to a polynomial. Given a probability distribution χ, let $a \leftarrow \chi$ denote that a is sampled from χ. Given a set S, let $a \xleftarrow{\$} S$ denote that a is sampled uniformly random from S.

C. Learning with errors and ring learning with errors

The security of our methods and used FHE schemes rely on the hardness of the learning with errors (LWE) problem [22] and the ring learning with errors (RLWE) problem [23], [24].

1) LWE: The LWE problem is parametrized by a dimension n, integer modulus q, a key distribution χ_{key} over $\mathbb{Z}$, and an error distribution χ over $\mathbb{Z}$.

Definition 1 (LWE distribution $\mathcal{D}_{\mathbf{s},\chi}$)**.** *Given $\mathbf{s} \in \mathbb{Z}_q^n$, the* LWE *distribution $\mathcal{D}_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing a uniformly random $\mathbf{a} \in \mathbb{Z}_q^n$ and error $e \leftarrow \chi$, and outputting $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*

We can represent $m \geq 1$ LWE instances with the same secret $\mathbf{s} \in \mathbb{Z}_q^n$ in the matrix form $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q)$, where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{b}, \mathbf{e} \in \mathbb{Z}_q^m$. Below we describe two versions of the LWE problem, both conjectured to be hard for appropriately chosen parameters. In practice, concrete security estimates of LWE instances are typically estimated using Albrecht et al.'s lattice estimator [25].

Definition 2 (Search LWE)**.** *Given m LWE instances $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q)$ sampled from $\mathcal{D}_{\mathbf{s},\chi}$ for some $\mathbf{s} \leftarrow \chi_{key}$, the search* LWE *problem is to recover* $\mathbf{s}$*.*

Definition 3 (Decision LWE)**.** *The decision* LWE *problem is to distinguish m LWE instances $(\mathbf{A}, \mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q)$ sampled from $\mathcal{D}_{\mathbf{s},\chi}$ with respect to $\mathbf{s} \leftarrow \chi_{key}$, from $(\mathbf{A}, \mathbf{b}) \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$.*

2) RLWE: The RLWE problem extends the LWE problem to structured algebraic rings. This work only focuses on the ring $\mathcal{R}$, a cyclotomic ring with a power-of-two cyclotomic order.

Definition 4 (RLWE distribution $\mathcal{D}_{\mathbf{s},\chi}$)**.** *Given $\mathbf{s} \in \mathcal{R}_q$ and an error distribution χ over $\mathcal{R}_q$, the* RLWE *distribution $\mathcal{D}_{\mathbf{s},\chi}$ over $\mathcal{R}_q^2$ is sampled by choosing a uniformly random $\mathbf{a} \in \mathcal{R}_q$ and an error $\mathbf{e} \leftarrow \chi$, and outputting $(\mathbf{a}, \mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \bmod q) \in \mathcal{R}_q^2$.*

Then we describe the two versions of the RLWE problem. To date, no known attack on RLWE problems significantly exploits its additional ring structure compared to LWE. Therefore, the security of an RLWE instance is estimated by transforming it to a corresponding LWE instance with the same dimension, modulus, and an appropriate error distribution. Let χ_{key} denote a key distribution over the ring $\mathcal{R}_q$.

Definition 5 (Search RLWE)**.** *Given m RLWE instances $\left\{(\mathbf{a}_i, \mathbf{b}_i = \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \bmod q)\right\}_{i \in [m]}$ sampled from $\mathcal{D}_{\mathbf{s},\chi}$ for some $\mathbf{s} \leftarrow \chi_{key}$, the search* RLWE *problem is to recover* $\mathbf{s}$*.*

Definition 6 (Decision RLWE)**.** *The decision* RLWE *problem is to distinguish m RLWE instances $\{(\mathbf{a}_i, \mathbf{b}_i = \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \bmod q)\}_{i \in [m]}$ sampled from $\mathcal{D}_{\mathbf{s},\chi}$ with respect to $\mathbf{s} \leftarrow \chi_{key}$, from m uniformly sampled pairs $\left\{(\mathbf{a}_i, \mathbf{b}_i) \xleftarrow{\$} \mathcal{R}_q^2\right\}_{i \in [m]}$.*

D. FHE primitives

Our contributions adopt a cross-scheme approach, and leverages concepts from a range of FHE schemes such as GSW [26], FHEW [6] and TFHE [27], along with state-of the-art optimizations [7], [8], [28]–[33]. This subsection outlines the relevant FHE ciphertext types, basic operations and subroutines.

1) FHE ciphertexts: Let t denote a plaintext modulus, $Q, q \gg t$ denote ciphertext moduli, and N the dimension of $\mathcal{R}_q$

- $\mathsf{LWE}_{\vec{s}}^{n,q}(m; \Delta) = (\vec{a}, b) \in \mathbb{Z}_q^{n+1}$ denotes an LWE ciphertext c such that the **phase** $\varphi(c)$ satisfies

$$\varphi(c) = b - \langle \vec{a}, \vec{s} \rangle \equiv \Delta \cdot m + e \pmod q$$

  where $\vec{s} \in \mathbb{Z}_q^n$ is the secret key. Furthermore, $m \in \mathbb{Z}_q$ corresponds to the message and e an error, usually sampled from a (sub)gaussian distribution. In many cases, methods will be independent of the encoding, in which case we write $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m)$ when $\varphi(c) = m$, i.e. we do not distinguish between message and error contribution.

- $C = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\mathfrak{m}; \Delta) = (\mathfrak{a}, \mathfrak{b}) \in \mathcal{R}_Q^2$ denotes an RLWE ciphertext C such that its phase $\varphi(C)$ satisfies

$$\varphi(C) = \mathfrak{b} - \mathfrak{a} \cdot \mathfrak{s} \equiv \Delta \mathfrak{m} + \mathfrak{e} \bmod Q,$$

  where $\mathfrak{s} \in \mathcal{R}$ is the secret key, $\mathfrak{m} \in \mathcal{R}_Q$ is a message and $\mathfrak{e}$ and error polynomial as in the LWE case. Likewise, if no assumption regarding the encoding is made, we write $C = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\mathfrak{m})$ for $\varphi(C) = \mathfrak{m}$.

For the sake of completeness, we note that both LWE and RLWE samples are instances of *generalized* LWE samples and both enable homomorphic addition and scalar multiplication by default via component-wise operations. Finally, we note that the subprocedures we introduce next implicitly rely on extended ciphertext types (e.g. RGSW [26] samples which can be viewed as matrices of RLWE samples) but choose not to describe them in detail in order to simplify the exposition.

2) FHE basic operations: Fully homomorphic encryption allows computation over ciphertexts. The basic homomorphic operations used in this work are as follows.

- Add denotes the homomorphic addition between (i) two ciphertexts of the same type (e.g. LWE or RLWE) encrypted under the same secret key, or (ii) one ciphertext and one plaintext. We represent addition of ciphertexts with the usual $+$.

- $\mathsf{ScalarMul}$ denotes homomorphic scalar multiplication. In the case of LWE samples, this corresponds to multiplication by an element of $\mathbb{Z}_q$ and for RLWE with an element of $\mathcal{R}_Q$. We represent scalar multiplication with the $\cdot$ operator
- $\mathsf{Aut}_{\sigma_i}(\mathsf{acc}, \mathsf{autK})$ denotes the homomorphic automorphism that maps $\mathsf{acc} = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\mathfrak{m})$ to $\mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\sigma_i(\mathfrak{m}))$ using an automorphism key autK, where $\sigma_i : \mathfrak{m}(X) \to \mathfrak{m}(X^i)$ denotes an automorphism in the Galois group $\mathrm{Gal}(\mathcal{K}/\mathbb{Q})$ for $\gcd(i, N) = 1$.
- $\mathsf{BlindRotate}(\mathsf{acc}, \mathsf{ct}, \mathsf{bsk})$ denotes the blind rotation operation, which takes $\mathsf{ct} = [\vec{a}, b] = \mathsf{LWE}_{\vec{s}}^{n,q}(\tilde{m})$, a *bootstrapping key* bsk, and the accumulator $\mathsf{acc} = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\mathfrak{p})$ as inputs. The typical LWE modulus q is N or $2N$, and let the. Then the output is $\mathsf{RLWE}_{\mathfrak{s}}^{N,Q}\left(\mathfrak{p} \cdot X^{\varphi(\mathsf{ct}) \bmod 2N}\right)$ with a constant noise level independent of ct.
- $\mathsf{SampleExtract}(\mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\mathfrak{m}), k) \to \mathsf{LWE}_{\mathfrak{s}}^{N,Q}(m_k)$ denotes the sample extraction taking $\mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\mathfrak{m})$ and an index $k \in [0, N-1]$ as inputs, and outputting $\mathsf{LWE}_{\mathfrak{s}}^{N,Q}(m_k)$ where m_k is the k-th coefficient of $\mathfrak{m}$. With slight abuse of notation, the LWE key of the output is the coefficient vector of the RLWE secret $\mathfrak{s}$.

Furthermore, we recall FHE procedure that we mention in our work, but do not use:

- $\mathsf{ModSwitch}_{q \to q'}(\mathsf{ct})$ denotes the modulus switching operation (i) from $\mathsf{LWE}_{\mathbf{s}}^{n,q}(m)$ to $\mathsf{LWE}_{\mathbf{s}}^{n,q'}(m)$ for $m \in \mathbb{Z}_q$, or (ii) from $\mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\mathfrak{m})$ to $\mathsf{RLWE}_{\mathfrak{s}}^{N,Q'}(\mathfrak{m})$ for $\mathfrak{m} \in \mathcal{R}_Q$.
- $\mathsf{KeySwitch}_{\vec{s} \to \vec{s}'}(\mathsf{ct}, \mathsf{ksK})$ denotes the key switching operation mapping a $\mathsf{ct} = \mathsf{LWE}_{\vec{s}}^{n,q}(\tilde{m})$ to $\mathsf{ct}' = \mathsf{LWE}_{\vec{s}'}^{n',q}(\tilde{m})$, a ciphertext encoding the same message under a different LWE key $\vec{s}'$ using an key-switching key ksK.

Finally, throughout this work we will omit the mention of automorphism, bootstrapping and key-switching keys to keep the exposition concise as their internal structure does not affect the correctness of our approaches. For concrete details, we refer to Appendix A-1 [5]

II. Phase Decomposition

In this section, we show that an LWE sample with a large modulus can be split into a set of small *pseudo*-LWE samples whose phases recombine linearly to the phase of the original sample. If the secret key is sparse, the phases of these pseudo-LWE samples are small and can be bounded over the integers. Section III builds on this observation. Throughout this section, let $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m) = [\vec{a}, b]$ be an LWE sample under a binary key $\vec{s} \in \{0,1\}^n$ with Hamming weight $\|\vec{s}\|_0 \le \beta$. We assume that q is a power of two and that $q = \alpha^l$ for a decomposition basis α, i.e., α is a power of two. This simplifies the exposition; in practice, $\alpha \mid q$ suffices. We identify every element of $\mathbb{Z}_q$ with its representative in $[0, q-1]$ and index digits by $i \in [0, l-1]$.

We start by decomposing c in base α, yielding tuples $c_\alpha^{(i)} = [\vec{a}^{(i)}, b^{(i)}] \in [0, \alpha-1]^n \times [0, \alpha-1]$, $i \in [0, l-1]$, such that

$$c = \sum_{i=0}^{l-1} \alpha^i \cdot c_\alpha^{(i)}. \tag{1}$$

Each $c_\alpha^{(i)}$ has the structure of an LWE sample, and we define its phase similarly to LWE samples, as $\varphi(c_\alpha^{(i)}) = b^{(i)} - \langle \vec{a}^{(i)}, \vec{s} \rangle \in \mathbb{Z}$. We call such tuples *pseudo-LWE samples*: they are not encryptions of a meaningful message. Since the phase is linear, Equation (1) implies

$$\varphi(c) = \sum_{i=0}^{l-1} \alpha^i \, \varphi(c_\alpha^{(i)}) \pmod q. \tag{2}$$

In other words, coefficient-wise reduction and division by α decompose the ciphertext and its phase *simultaneously*: the phase of c is a fixed linear combination of the phases of the $c_\alpha^{(i)}$. We stress that the $c_\alpha^{(i)}$ are **not** encryptions of the base-α digits of $\varphi(c)$. The phase of a single $c_\alpha^{(i)}$ is in general unrelated to the i-th digit of $\varphi(c)$ and can exceed α in absolute value. If $\vec{s}$ is sparse, however, it is bounded, as Proposition 1 states.

Proposition 1. *Let $c_\alpha^{(i)}$, $i \in [0, l-1]$, be the pseudo-LWE samples obtained by decomposing the coefficients of $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m)$ in base α, with $\vec{s} \in \{0,1\}^n$ and $\|\vec{s}\|_0 \le \beta$. Then, for every i,*

$$-(\alpha-1)\beta \le \varphi(c_\alpha^{(i)}) \le \alpha - 1.$$

Proof. All coefficients of $c_\alpha^{(i)}$ lie in $[0, \alpha-1]$. Since $\vec{s}$ is binary,

$$\varphi(c_\alpha^{(i)}) = b^{(i)} - \sum_{j=0}^{n-1} a_j^{(i)} s_j = b^{(i)} - \sum_{j : s_j = 1} a_j^{(i)}.$$

The sum has at most β terms in $[0, \alpha-1]$, so it lies in $[0, (\alpha-1)\beta]$. Together with $b^{(i)} \in [0, \alpha-1]$, the claim follows. □

Our next goal is to allow processing of the pseudo-LWE samples with blind rotation. Recall that a blind rotation over $\mathcal{R} = \mathbb{Z}_Q[X]/(X^N+1)$ on an LWE sample c modulo $2N$ multiplies the message of an accumulator $C = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(\mathfrak{p}; \Delta)$

by $X^{\varphi(c) \bmod 2N}$. Since $X^N \equiv -1$ in $\mathcal{R}$, any phase in $[N, 2N)$ introduces a sign flip. Proposition 1 shows that the phases of the pseudo-LWE samples lie in an interval of length $(\alpha-1)(\beta+1)$. If this length is smaller than N, shifting each phase by $(\alpha-1)\beta$ moves it into $[0, N)$, and no sign flip can occur. Shifting each pseudo-LWE sample independently, however, breaks Equation (2):

$$\sum_{i=0}^{l-1} \alpha^i \left(\varphi(c_\alpha^{(i)}) + (\alpha-1)\beta\right) \not\equiv \varphi(c) \pmod q.$$

Algorithm 1 resolves this by subtracting the accumulated shift from c *before* decomposing it. The resulting pseudo-LWE samples have non-negative phases and still satisfy Equation (2), as stated in Theorem 1.

Algorithm 1: PhaseDecomp

Input: LWE sample $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m)$, q a power of two
Input: Basis α with $q = \alpha^l$
Input: Bound β with $\|\vec{s}\|_0 \le \beta$
Output: Pseudo-LWE samples $(c_\alpha^{(i)})_{0 \le i < l}$

1 $P \leftarrow (\alpha-1)\beta$
2 $\kappa \leftarrow \sum_{i=0}^{l-1} \alpha^i P$
3 $\bar{c} \leftarrow c - [\vec{0}, \kappa] \pmod q$
4 **for** $i \leftarrow 0$ to $l-1$ **do**
5 $\quad c_\alpha^{(i)} \leftarrow [\bar{c}]_\alpha + [\vec{0}, P]$
6 $\quad \bar{c} \leftarrow \left\lfloor \frac{\bar{c}}{\alpha} \right\rfloor$
7 **end**

Result: $(c_\alpha^{(i)})_{0 \le i < l}$

Theorem 1. *Let $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m)$ with $q = \alpha^l$ a power of two, $\vec{s} \in \{0,1\}^n$, and $\|\vec{s}\|_0 \le \beta$, and let*

$$\eta = (\alpha-1)(\beta+1).$$

On input c, α, β, Algorithm 1 returns pseudo-LWE samples $c_\alpha^{(i)}$, $i \in [0, l-1]$, such that over $\mathbb{Z}$

$$0 \le \varphi(c_\alpha^{(i)}) \le \eta$$

and

$$\varphi(c) = \sum_{i=0}^{l-1} \alpha^i \, \varphi(c_\alpha^{(i)}) \pmod q.$$

Proof. Let $\bar{c}_\alpha^{(i)} = [\bar{c}]_\alpha$ denote the digits computed in Line 5 before adding $[\vec{0}, P]$, where $\bar{c}$ is the sample computed in Line 3. These are the base-α digits of $\bar{c}$, so Equation (2) holds for them. Adding $[\vec{0}, P]$ increases the phase by exactly P. Hence,

$$\begin{aligned}\sum_{i=0}^{l-1} \alpha^i \, \varphi(c_\alpha^{(i)}) &= \sum_{i=0}^{l-1} \alpha^i \left(\varphi(\bar{c}_\alpha^{(i)}) + P\right) \\ &\equiv \varphi(\bar{c}) + \kappa \\ &\equiv \varphi(c) - \kappa + \kappa = \varphi(c) \pmod q.\end{aligned}$$

For the bound, note that $\bar{c}$ is an LWE sample modulo q under the same key. Proposition 1 therefore applies to the digits $\bar{c}_\alpha^{(i)}$, giving $-(\alpha-1)\beta \le \varphi(\bar{c}_\alpha^{(i)}) \le \alpha - 1$. Adding $P = (\alpha-1)\beta$ yields $0 \le \varphi(c_\alpha^{(i)}) \le (\alpha-1)(\beta+1) = \eta$. □

We make three remarks on Algorithm 1. First, the correction term simplifies to $\kappa = P\frac{\alpha^l - 1}{\alpha - 1} = \beta(q-1) \equiv -\beta \pmod q$, so Line 3 amounts to adding β to the b-component of c. Second, the pseudo-LWE samples have integer coefficients bounded by $\alpha - 1 + P$, and their phases are exact. If $\eta < N$, a blind rotation on $c_\alpha^{(i)}$, with its coefficients reduced modulo $2N$, therefore multiplies the accumulator by exactly $X^{\varphi(c_\alpha^{(i)})}$. No modulus switching is needed, and no rounding error is introduced. Third, the perturbation-correction technique of Algorithm 1 is known in implementations: the TFHE library [34] uses it to compute a balanced gadget decomposition by adding a constant offset once and extracting unsigned digits. Compared to computing balanced digits with a carry from one digit to the next, this removes the dependency between consecutive digits.

The radix decomposition is only one way to split a ciphertext together with its phase. Other examples include a CRT decomposition of the modulus, or splitting along the key dimension, i.e., treating $c \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ as a tuple in $\mathbb{Z}_q^{n/2} \times \mathbb{Z}_q^{n/2} \times \mathbb{Z}_q$. Mixed radices and centered remainders are possible as well. For instance, the homomorphic Euclidean division used in the recent Meta-PBS [9] splits a ciphertext into a quotient modulo $2N$ and a centered remainder whose phases recombine as

in Equation (2), and it can be viewed as such a decomposition (cf. Section III-B). In this work, we focus on the radix decomposition.

Building on our previous discussion, we now show how the phases of the pseudo-LWE samples determine the base-α digits of $\varphi(c)$. Let $\varphi(c) = \sum_{i=0}^{l-1} m_i \alpha^i$ with $m_i \in [0, \alpha-1]$, i.e., m_i is the i-th digit of $\varphi(c)$ in base α. Theorem 2 gives a recurrence that computes the m_i from the $\varphi(c_\alpha^{(i)})$. We defer its homomorphic evaluation to Section III.

Theorem 2. *Let $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m)$ and let $c_\alpha^{(i)}$, $i \in [0, l-1]$, be the output of Algorithm 1. Define*

$$g_i = \varphi(c_\alpha^{(i)}) + h_i, \tag{3}$$

$$m_i = [g_i]_\alpha, \tag{4}$$

$$h_{i+1} = \left\lfloor \frac{g_i}{\alpha} \right\rceil, \quad h_0 = 0. \tag{5}$$

Then $m_0, \ldots, m_{l-1}$ are the base-α digits of $\varphi(c)$, i.e., $\varphi(c) = \sum_{i=0}^{l-1} m_i \alpha^i$ with $m_i \in [0, \alpha-1]$.

Proof. We show by induction on $k \in [0, l]$ that

$$\sum_{i=0}^{k-1} \alpha^i \, \varphi(c_\alpha^{(i)}) = \sum_{i=0}^{k-1} \alpha^i m_i + \alpha^k h_k.$$

For $k = 0$, both sides are 0. For the induction step, add $\alpha^k \, \varphi(c_\alpha^{(k)})$ to both sides. The right-hand side becomes $\sum_{i<k} \alpha^i m_i + \alpha^k g_k$, and $g_k = m_k + \alpha h_{k+1}$ by Equations (4) and (5). For $k = l$, the term $\alpha^l h_l = q h_l$ vanishes modulo q, and Theorem 1 gives $\varphi(c) \equiv \sum_{i=0}^{l-1} \alpha^i m_i \pmod q$. Since $m_i \in [0, \alpha-1]$, the right-hand side lies in $[0, q-1]$, so it equals $\varphi(c)$, and the m_i are its base-α digits. □

The recurrence has a simple interpretation. By Theorem 1, the values $\varphi(c_\alpha^{(i)})$ form a base-α representation of $\varphi(c)$ with the redundant digit set $[0, \eta]$. Theorem 2 converts this representation into the standard digit set $[0, \alpha-1]$, and h_i is the carry into position i.

To close this section, we note that the recurrence in Theorem 2 processes the pseudo-LWE samples from least to most significant and keeps only a small state between steps. It is therefore an instance of a finite-state transducer, and we can generalize Equations (3)–(5) to

$$\mathsf{out}_i = F(g_i), \tag{6}$$

$$g_i = G(\varphi(c_\alpha^{(i)})) + H(g_{i-1}, \mathsf{out}_{i-1}), \tag{7}$$

for functions F, G, and H. Theorem 2 is the special case $F(x) = [x]_\alpha$, $G(x) = x$, and $H(g, \mathsf{out}) = \lfloor g/\alpha \rfloor$. Equations (6) and (7) are a blueprint for evaluating transducer-type computations on the phase of c. Guimarães et al. [35] studied a similar class of functions under the name "chaining method". Their approach, however, requires the message to be given as encrypted digits, i.e., it requires a prior digit decomposition or. In Section III we show the homomorphic evaluation of the recurrence relation in Theorem 2 which extends immediately to Equations 6 and Equation 7, making a prior digit decomposition unnecessary.

III. Applications of Phase Decomposition

In this section, we build concrete algorithms on top of Section II. We mainly target accumulator-based schemes such as FHEW and TFHE, but the methodology also applies to SIMD schemes, which we illustrate in Section III-A2.

A. Digit Decomposition

We first derive a digit decomposition algorithm from Theorem 2. Section III-A1 considers accumulator-based schemes with a sparse binary key. Section III-A2 then sketches a variant for SIMD schemes that only assumes the scheme can evaluate univariate functions on small plaintext spaces, which shows that the approach is largely independent of the underlying scheme.

1) Accumulator-Based Schemes: By Equations (3)–(5), a digit decomposition needs only two operations on the intermediate values g_i: reduction modulo α, which yields the digit m_i, and truncation by α, which yields the carry h_{i+1}. Both operations are well studied and are often the bottleneck of homomorphic computations, e.g., in CKKS bootstrapping.

A first natural approach keeps the g_i in the phase of LWE samples. A blind rotation on $c_\alpha^{(0)}$ yields $C = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{g_0}; \Delta_Q)$. Multiplying by $\mathfrak{p} = \sum_{j=0}^{N-1} \left\lfloor \frac{j}{\alpha} \right\rceil X^{-j}$ gives a ciphertext whose constant coefficient encrypts $\Delta_Q \left\lfloor \frac{g_0}{\alpha} \right\rceil + e = \Delta_Q h_1 + e$. We extract it as an LWE sample, switch it to the parameters of $c_\alpha^{(1)}$, and add $c_\alpha^{(1)}$ to obtain an LWE sample c^+ of g_1. To obtain m_1 or h_2, we must then blind-rotate c^+, and this requires $\varphi(c^+) < N$. This is where the approach breaks down: the coefficients of c^+ are no longer bounded by α, and c^+ carries the error e of the previous step. Both effects force a much larger ring dimension N, which severely degrades performance.

Hence we look for a different path and avoid this problem entirely by never leaving the exponent. Let

$$C^{(i)} = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{g_i}; \Delta_Q),$$

where Δ_Q denotes the scaling factor of the accumulator. Given $C^{(i)}$, we obtain $C^{(i+1)}$ in two steps:

1) Compute $\mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{h_{i+1}}; \Delta_Q)$ with $h_{i+1} = \lfloor g_i/\alpha \rfloor$ through a homomorphic truncation in the exponent;
2) Blind-rotate this ciphertext with $c_\alpha^{(i+1)}$, which adds $\varphi(c_\alpha^{(i+1)})$ to the exponent and yields $X^{g_{i+1}}$ by Equation (3).

The digit $m_i = [g_i]_\alpha$ is read out from $C^{(i)}$ by a polynomial multiplication. For all of this, the exponents g_i must stay in $[0, N)$, since otherwise the negacyclic wrap-around flips signs. Theorem 1 is not sufficient for this: it bounds $\varphi(c_\alpha^{(i)})$, but g_i also contains the carry h_i. Lemma 1 bounds both.

Lemma 1. *Let g_i and h_i be defined as in Theorem 2. Then, for all $i \in [0, l-1]$,*

$$0 \le h_i \le \beta \quad \textit{and} \quad 0 \le g_i \le \alpha(\beta+1) - 1.$$

In particular, $g_i < N$ for all i if $\alpha(\beta+1) \le N$.

Proof. Non-negativity follows from Theorem 1. We show $h_i \le \beta$ by induction. The claim holds for $h_0 = 0$. If $h_i \le \beta$, then Theorem 1 gives

$$g_i = \varphi(c_\alpha^{(i)}) + h_i \le (\alpha - 1)(\beta + 1) + \beta = \alpha(\beta+1) - 1,$$

and hence $h_{i+1} = \lfloor g_i/\alpha \rfloor \le \beta$. The bound on g_i follows from the same inequality. □

The condition $\alpha(\beta+1) \le N$ replaces the condition $\eta < N$ of Theorem 1 and is only slightly stronger, since $\alpha(\beta+1) - 1 = \eta + \beta$. We assume it for the rest of this section. Since α and N are powers of two and $\alpha < N$, it also implies $\alpha \mid N$.

We now describe the homomorphic truncation in the exponent procedure, i.e. how to compute $\mathsf{RLWE}(\Delta_Q X^{\lfloor g/\alpha \rfloor})$ from $\mathsf{RLWE}(\Delta_Q X^g)$ for $0 \le g < N$. Algorithm 2 gives the full procedure, and Figure 1 illustrates it. Since α is a power of two, α has no inverse modulo $2N$, and the map $X \mapsto X^{\alpha^{-1}}$ is not available as an automorphism. We therefore proceed in three steps. We write $g = \alpha A + B$ with $A \in [0, N/\alpha - 1]$ and $B \in [0, \alpha - 1]$ and proceed as follows.

First, multiplying by $\sum_{j=0}^{\alpha-1} X^{-j}$ turns X^g into the sum of α consecutive monomials $X^g, X^{g-1}, \ldots, X^{g-\alpha+1}$. Exactly one of them, $X^{\alpha A}$, has an exponent divisible by α. If the window wraps around below 0, the wrapped monomials have negative sign and exponents in $[N - \alpha + 1, N - 1]$, none of which is divisible by α. Second, a partial or relative homomorphic trace with $\log_2(\alpha)$ automorphisms removes all coefficients at indices not divisible by α, which leaves $\mathsf{RLWE}(X^{\alpha A})$ up to a known scalar. Finally, the remaining message lies in the subring $\mathbb{Z}[X^\alpha]$ and has N/α potentially nonzero coefficients. A ciphertext expansion as in [30] splits it into N/α ciphertexts C_j, where C_j encrypts the constant Δ_Q if $j = A$ and 0 otherwise. The sum $\sum_j X^j C_j$ then encrypts $\Delta_Q X^A$.

Steps 2 and 3 use the same sequence of automorphisms σ_{2^i+1}, $i = \log_2(N), \ldots, 1$. Furthermore, each step halves the number of nonzero coefficients in any RLWE and doubles the values of remaining ones; Step 2 discards all coefficient with indices not a multiple of α, whereas Step 3 iteratively splits the coefficients into different ciphertexts. Hence, the pre-multiplication by $N^{-1} \bmod Q$ cancels the total factor $2^{\log_2 N} = N$ and requires Q to be odd, though we note that alternatives exist in the even case [36]. The computation of $[g]_\alpha$ in the exponent would follow a similar blueprint, but we will not use it. Theorem 3 summarizes the properties of Algorithm 2

Theorem 3 (Correctness of Algorithm 2)**.** *Let Q be odd, let $\alpha \mid N$ be a power of two, and let $\mathsf{acc} = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^g; \Delta_Q)$ with $0 \le g < N$ and noise variance σ^2. On input acc and α, Algorithm 2 outputs $\mathsf{acc}_{\mathsf{out}} = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{\lfloor g/\alpha \rfloor}; \Delta_Q)$ with noise variance*

$$\sigma^2_{\mathsf{Trunc}} \le \alpha \cdot \sigma^2 + \frac{N}{\alpha} \cdot \sigma^2_{\mathsf{Trace}},$$

where $\sigma^2_{\mathsf{Trace}}$ is the key-switching noise variance accumulated along one path of $\log_2(N)$ automorphism evaluations, i.e., the noise of a homomorphic trace [36]. Furthermore, the algorithm uses $\log_2(\alpha) + N/\alpha - 1$ automorphism evaluations.

Proof. Write $g = \alpha A + B$ with $B \in [0, \alpha - 1]$. Then, following the previous discussion, acc' encrypts $N^{-1}\Delta_Q \mathfrak{q}$ after Line 2, where $\mathfrak{q}$ has coefficients in $\{0, \pm 1\}$, and corresponds to the sum of consecutive monomials, with a single one having an index divisible by α.

We use the following property of the automorphisms $\tau_k = \sigma_{2^k+1} : X \mapsto X^{2^k+1}$ of $\mathcal{R}$. Let $\mathfrak{u} = \sum_{j=0}^{2^k-1} u_j X^{jN/2^k}$ be a polynomial whose coefficients have stride $N/2^k$. Since $X^{jN/2^k(2^k+1)} = (-1)^j X^{jN/2^k}$,

$$\mathfrak{u} + \tau_k(\mathfrak{u}) = 2 \sum_{j \text{ even}} u_j X^{jN/2^k},$$
$$\mathfrak{u} - \tau_k(\mathfrak{u}) = 2 \sum_{j \text{ odd}} u_j X^{jN/2^k}.$$

Algorithm 2: HomTruncTrace

Input: RLWE sample $\mathsf{acc} = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^g; \Delta_Q)$, $0 \le g < N$
Input: Truncation factor α, a power of two with $\alpha \mid N$
Output: RLWE sample $\mathsf{acc}_{\mathsf{out}} = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{\lfloor g/\alpha \rceil}; \Delta_Q)$

1 $\mathfrak{p} \leftarrow N^{-1} \sum_{j=0}^{\alpha-1} X^{-j} \pmod{Q}$
2 $\mathsf{acc}' \leftarrow \mathfrak{p} \cdot \mathsf{acc}$
3 **for** $i = \log_2(N)$ down to $\log_2(N/\alpha) + 1$ **do**
4 $\quad \mathsf{acc}' \leftarrow \mathsf{acc}' + \mathsf{Aut}_{\sigma_{2^i+1}}(\mathsf{acc}')$
5 **end**
6 $S \leftarrow \{\langle \mathsf{acc}' : 0 \rangle\}$
7 **for** $i = \log_2(N/\alpha)$ down to 1 **do**
8 $\quad d \leftarrow \log_2(N/\alpha) - i$
9 $\quad S' \leftarrow \emptyset$
10 $\quad$ **for** $\langle \mathsf{acc}^* : v \rangle$ in S **do**
11 $\quad\quad \mathsf{acc}^\pi \leftarrow \mathsf{Aut}_{\sigma_{2^i+1}}(\mathsf{acc}^*)$
12 $\quad\quad \mathsf{acc}^+ \leftarrow \mathsf{acc}^* + \mathsf{acc}^\pi$
13 $\quad\quad \mathsf{acc}^- \leftarrow X^{-N/2^i} \cdot (\mathsf{acc}^* - \mathsf{acc}^\pi)$
14 $\quad\quad S' \leftarrow S' \cup \{\langle \mathsf{acc}^+ : v \rangle, \langle \mathsf{acc}^- : v + 2^d \rangle\}$
15 $\quad$ **end**
16 $\quad S \leftarrow S'$
17 **end**
18 $\mathsf{acc}_{\mathsf{out}} \leftarrow 0$
19 **for** $\langle \mathsf{acc}^* : v \rangle$ in S **do**
20 $\quad \mathsf{acc}_{\mathsf{out}} \leftarrow \mathsf{acc}_{\mathsf{out}} + X^v \cdot \mathsf{acc}^*$
21 **end**

Result: $\mathsf{acc}_{\mathsf{out}}$

Both results have stride $N/2^{k-1}$, the second one after multiplication by $X^{-N/2^k}$. Any message polynomial in Algorithm 2 fulfills the property of $\mathfrak{u}$ for $k = i$ in every iteration of both loops, starting from a stride of $1 = N/2^{\log_2 N}$. After the first loop, acc' therefore encrypts $\alpha N^{-1} \Delta_Q \sum_{j=0}^{N/\alpha-1} q_{\alpha j} X^{\alpha j} = \alpha N^{-1} \Delta_Q X^{\alpha A}$. The second loop is the ciphertext expansion of [30] ran in reverse and applied to the subring $\mathbb{Z}[X^\alpha]$. After the iteration with level d, the ciphertext with label v holds the coefficients $q_{\alpha j}$ with $j \equiv v \pmod{2^{d+1}}$, shifted so that $q_{\alpha v}$ is the constant coefficient. After $\log_2(N/\alpha)$ levels, the ciphertext with label v encrypts the constant $N \cdot N^{-1} \Delta_Q q_{\alpha v} = \Delta_Q q_{\alpha v}$, which is Δ_Q for $v = A$ and 0 otherwise. Line 20 thus yields an encryption of $\Delta_Q X^A = \Delta_Q X^{\lfloor g/\alpha \rceil}$. The first loop performs $\log_2(\alpha)$ automorphisms, and level d of the second loop performs 2^d, for a total of $\log_2(\alpha) + N/\alpha - 1$.

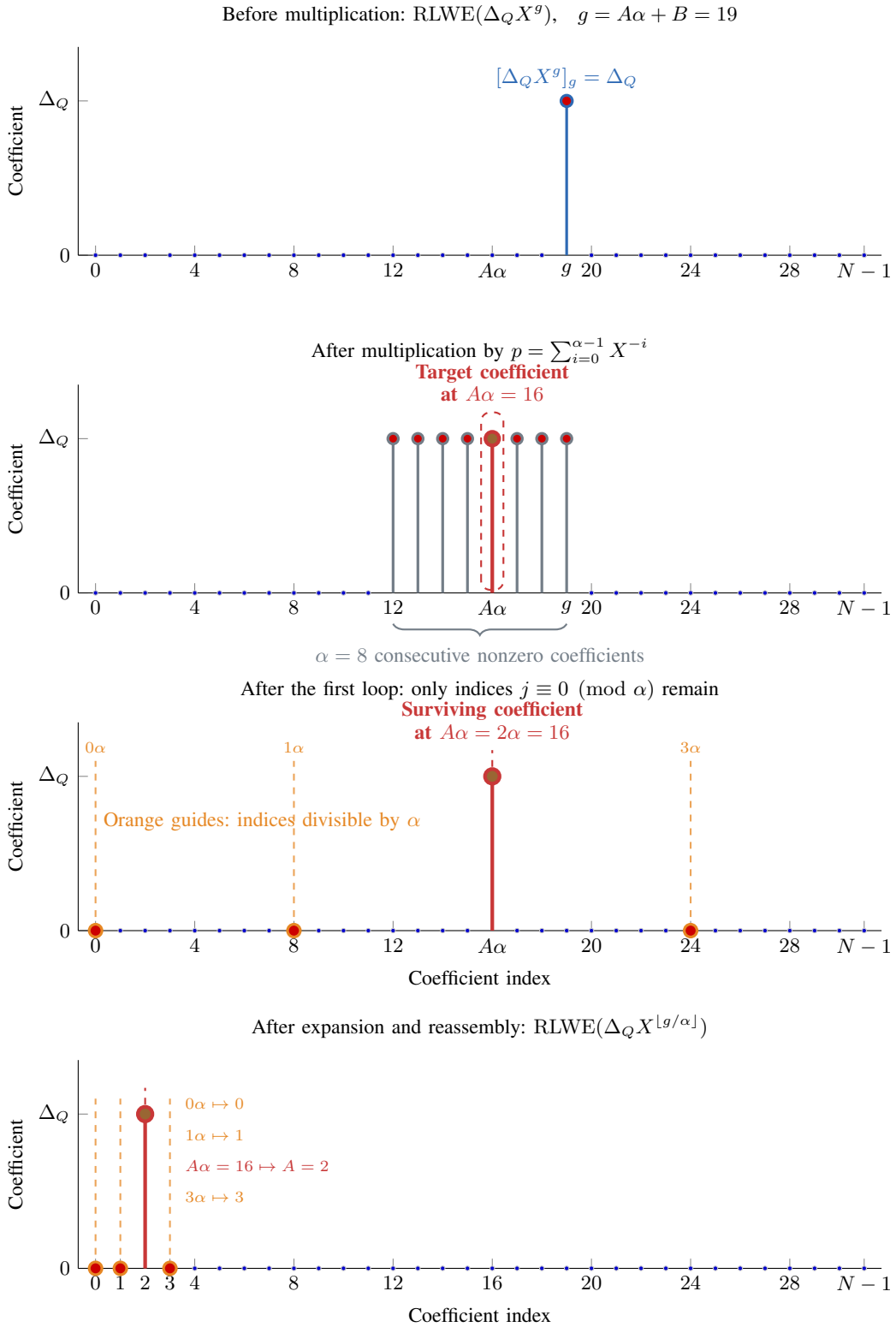
For the noise, note that $\mathfrak{p}$ has α coefficients $\pm N^{-1}$. After Line 2, each coefficient of the input noise, scaled by N, has variance at most $\alpha\sigma^2$. The automorphisms act on the noise in the same way as on the message, so this component is decimated and rescaled exactly like the message, and the factors N and N^{-1} cancel. Each output ciphertext of the expansion therefore contains one coefficient of this noise in its constant term, and these coefficients end up in distinct positions in Line 20. They are not amplified, which gives the term $\alpha\sigma^2$. In addition, each ciphertext in S carries the key-switching noise of $\log_2(N)$ automorphism evaluations on its path, which corresponds to the variance of a homomorphic trace $\sigma^2_{\mathsf{Trace}}$ [36] . Monomial multiplications do not change the noise variance, so summing the N/α ciphertexts in Line 20 contributes at most $\frac{N}{\alpha}\sigma^2_{\mathsf{Trace}}$ per coefficient. □

Finally, we give the digit extraction approach in order to complete the digit decomposition procedure. The digits $m_i = [g_i]_\alpha$ are never reused, and the carries h_i do not depend on them. We therefore do not need a reduction modulo α in the exponent. Instead, we read out m_i by multiplying $C^{(i)}$ with the centered test polynomial

$$\mathfrak{p}_{\bmod} = \sum_{j=0}^{N-1} \left([j]_\alpha - \frac{\alpha}{2} \right) X^{-j}.$$

Since $0 \le g_i < N$ by Lemma 1, the constant coefficient of $X^{g_i} \cdot \mathfrak{p}_{\bmod}$ is $[g_i]_\alpha - \frac{\alpha}{2}$. The phase of $C^{(i)} \cdot \mathfrak{p}_{\bmod}$ therefore has constant coefficient $\Delta_Q([g_i]_\alpha - \frac{\alpha}{2}) + e$ for a small error e. A sample extraction followed by adding the constant $\Delta_Q \frac{\alpha}{2}$ yields an LWE sample of m_i. Note that centering the coefficients of the test polynomial around zero reduces its norm, and hence the output noise, by a factor of about four compared to the uncentered polynomial $\sum_j [j]_\alpha X^{-j}$. Finally, we combine all steps in Algorithm 3 and give its properties in Theorem 4. We note that after each iteration, the output noise variance grows by a factor of roughly α per digit. In Section IV, we discuss how to restart the computation to keep it bounded instead.

Fig. 1: Homomorphic truncation in the exponent (Algorithm 2) for $N = 32$, $\alpha = 8$, and $g = A\alpha + B = 19$ with $A = 2$, $B = 3$. From top to bottom: the input $\mathsf{RLWE}(\Delta_Q X^g)$; after multiplication by $\sum_{j=0}^{\alpha-1} X^{-j}$, which creates α consecutive nonzero coefficients, exactly one of them at an index divisible by α; after the first loop, which removes all coefficients at indices not divisible by α; and after expansion and reassembly, which moves the coefficient at index $A\alpha$ to index $A = \lfloor g/\alpha \rfloor$. The factor N^{-1} and the scaling by the automorphism steps cancel and are omitted.



Algorithm 3: Digit decomposition DigitDec

Input: LWE sample $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m)$
Input: Basis α with $q = \alpha^l$
Input: Bound β with $\|\vec{s}\|_0 \le \beta$
Input: Output scaling factor Δ_Q
Output: LWE samples $(c_{\mathsf{out},i})_{0\le i<l}$ with $c_{\mathsf{out},0} = \mathsf{LWE}_{\vec{s}}^{n,N}(m_0; \frac{N}{\alpha})$ and $c_{\mathsf{out},i} = \mathsf{LWE}_{\mathfrak{s}}^{N,Q}(m_i; \Delta_Q)$ for $i \ge 1$

1 $(c_\alpha^{(i)})_{0\le i<l} \leftarrow \mathsf{PhaseDecomp}(c, \alpha, \beta)$
2 $c_{\mathsf{out},0} \leftarrow \frac{N}{\alpha} \cdot c_\alpha^{(0)} \pmod N$
3 $\mathsf{acc} \leftarrow [\mathfrak{o}, \Delta_Q]$
4 $\mathfrak{p}_{\bmod} \leftarrow \sum_{j=0}^{N-1}([j]_\alpha - \frac{\alpha}{2})X^{-j}$
5 $\mathsf{acc} \leftarrow \mathsf{BlindRotate}(\mathsf{acc}, c_\alpha^{(0)})$
6 **for** $i = 1$ to $l-1$ **do**
7 $\quad \mathsf{acc} \leftarrow \mathsf{HomTruncTrace}(\mathsf{acc}, \alpha)$
8 $\quad \mathsf{acc} \leftarrow \mathsf{BlindRotate}(\mathsf{acc}, c_\alpha^{(i)})$
9 $\quad c_{\mathsf{out},i} \leftarrow \mathsf{SampleExtract}(\mathsf{acc} \cdot \mathfrak{p}_{\bmod}, 0) + [\vec{0}, \Delta_Q \frac{\alpha}{2}]$
10 **end**
Result: $(c_{\mathsf{out},i})_{0\le i<l}$

Theorem 4 (Correctness of Algorithm 3)**.** *Let $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m)$ with $\vec{s} \in \{0,1\}^n$, $\|\vec{s}\|_0 \le \beta$, and $q = \alpha^l$, let Q be odd, and let $\alpha(\beta+1) \le N$. Let $\varphi(c) = \sum_{i=0}^{l-1} \alpha^i m_i$ with $m_i \in [0, \alpha-1]$. Algorithm 3 returns $c_{\mathsf{out},0} = \mathsf{LWE}_{\vec{s}}^{n,N}(m_0; \frac{N}{\alpha})$ without error and $c_{\mathsf{out},i} = \mathsf{LWE}_{\mathfrak{s}}^{N,Q}(m_i; \Delta_Q)$ for $i \in [1, l-1]$. Let σ_i^2 denote the noise variance of* acc *after the blind rotation in iteration i, with $\sigma_0^2 = \sigma_{\mathsf{br}}^2$ after Line 5. Then*

$$\begin{aligned}\sigma_i^2 &\le \alpha\sigma_{i-1}^2 + \frac{N}{\alpha}\sigma_{\mathsf{Trace}}^2 + \sigma_{\mathsf{br}}^2 \\ &= \frac{\alpha^{i+1}-1}{\alpha-1}\sigma_{\mathsf{br}}^2 + \frac{\alpha^i - 1}{\alpha - 1} \cdot \frac{N}{\alpha}\sigma_{\mathsf{Trace}}^2,\end{aligned}$$

and the output samples have variance

$$\sigma_{\mathsf{out},i}^2 \le \|\mathfrak{p}_{\bmod}\|_2^2 \, \sigma_i^2 = \frac{N(\alpha^2+2)}{12}\sigma_i^2.$$

Proof. Correctness. For $i = 0$, all terms of Equation (2) except the first are multiples of α, so $\varphi(c_\alpha^{(0)}) \equiv \varphi(c) \equiv m_0 \pmod \alpha$. Multiplying by N/α and reducing modulo N gives a noiseless LWE sample of m_0 with scale N/α. For $i \ge 1$, we show by induction that acc encrypts $\Delta_Q X^{g_i}$ after the blind rotation in iteration i. For $i = 0$, Line 5 gives $\Delta_Q X^{\varphi(c_\alpha^{(0)})} = \Delta_Q X^{g_0}$, since $h_0 = 0$. Assume acc encrypts $\Delta_Q X^{g_{i-1}}$. Since $g_{i-1} < N$ by Lemma 1, Theorem 3 shows that HomTruncTrace returns an encryption of $\Delta_Q X^{h_i}$. The blind rotation with $c_\alpha^{(i)}$ multiplies it by $X^{\varphi(c_\alpha^{(i)})}$, which is exact because $\varphi(c_\alpha^{(i)}) \in [0, N)$ (Theorem 1). The result encrypts $\Delta_Q X^{g_i}$, again with $g_i < N$. Multiplying by $\mathfrak{p}_{\bmod}$, extracting the constant coefficient, and adding $\Delta_Q \frac{\alpha}{2}$ yields an LWE sample of $[g_i]_\alpha = m_i$, which is the i-th digit of $\varphi(c)$ by Theorem 2.

Noise. Each iteration applies Theorem 3 to a ciphertext with variance σ_{i-1}^2, and the blind rotation adds σ_{br}^2, since it preserves the noise of the input accumulator. Solving the recurrence with $\sigma_0^2 = \sigma_{\mathsf{br}}^2$ gives the closed form. Multiplying by $\mathfrak{p}_{\bmod}$ scales the variance by $\|\mathfrak{p}_{\bmod}\|_2^2 = \frac{N}{\alpha}\sum_{u=-\alpha/2}^{\alpha/2-1} u^2 = \frac{N(\alpha^2+2)}{12}$. Sample extraction and adding a constant do not change the noise. □

2) SIMD Schemes: We now give a blueprint for digit decomposition in a SIMD scheme. We assume that the scheme can evaluate arbitrary univariate functions on slot values in $[0, \alpha^2 - 1]$; this has been studied, e.g., for CKKS in [16]. We also assume

$$\alpha > \beta + 1,$$

for reasons explained below. A direct approach implements Equations (4) and (5) with the function evaluation, where the phases $\varphi(c_\alpha^{(i)})$ are obtained through a linear transformation. Let $\mathsf{d}_{\mathsf{func}}$ denote the multiplicative depth of one function evaluation. Since the truncation in each step depends on the previous one, this approach has depth $l \cdot \mathsf{d}_{\mathsf{func}}$.

However, we can reduce the depth by viewing the recurrence as an addition. Write $\varphi(c_\alpha^{(i)}) = x_i + \alpha y_i$ with $x_i = [\varphi(c_\alpha^{(i)})]_\alpha$

and $y_i = \left\lfloor \varphi(c_\alpha^{(i)})/\alpha \right\rfloor$. Inserting this into Equation (2) and splitting the sum gives

$$\begin{aligned}\varphi(c) &\equiv \sum_{i=0}^{l-1} \alpha^i (x_i + \alpha y_i) \\ &\equiv \sum_{i=0}^{l-1} \alpha^i x_i + \sum_{i=1}^{l-1} \alpha^i y_{i-1} \pmod q, \end{aligned} \tag{8}$$

where the term $\alpha^l y_{l-1}$ vanishes modulo $q = \alpha^l$. Equation (8) is the sum of the base-α numbers $X = x_{l-1}x_{l-2}\dots x_0$ and $Y = y_{l-2}\dots y_0 0$, reduced modulo α^l. For X and Y to be valid base-α numbers, all their digits must be smaller than α. This holds for x_i by definition. For y_i, Theorem 1 gives $y_i \le \left\lfloor \frac{(\alpha-1)(\beta+1)}{\alpha} \right\rfloor \le \beta$, and $\beta < \alpha$ holds under our assumption $\alpha > \beta+1$. Each digit position then sums to at most $2\alpha - 2$, so all carries are in $\{0, 1\}$.

Implementing Equations (4) and (5) directly corresponds to a ripple-carry adder. Parallel-prefix adders such as the Kogge–Stone [37] or Sklansky [38] adders compute all carries in depth proportional to $\log_2(l)$. They require only one change compared to the binary case, namely the initial propagate and generate signals. For position j with digit sum $x_j + y_{j-1}$, the propagate signal is 1 if and only if the sum equals $\alpha - 1$, in which case an incoming carry is passed on. The generate signal is 1 if and only if the sum is at least α, in which case the position produces a carry by itself. The prefix computation is then identical to the binary case.

Algorithm 4 shows a Kogge–Stone variant. For a scheme with N_s slots, it processes N_s/l LWE samples in parallel, with the l pseudo-LWE samples of the i-th input in slots $li, \dots, li+l-1$. We write $\mathsf{Rotate}(C, d)$ for the rotation that moves the content of slot j to slot $j + d$, and $\boxdot$ for slot-wise multiplication with a plaintext vector. The mask $\vec{\mu}^{(d)}$ with $\mu_j^{(d)} = [\,(j \bmod l) \ge d\,]$ removes values that a rotation moves across block boundaries, and $\vec{o}^{(d)} = \vec{1} - \vec{\mu}^{(d)}$. In a conventional Kogge–Stone adder, the group generate signal is updated as $G \leftarrow G \vee (P \wedge G')$, where (P, G) are the signals of the current group and G' is the generate signal of the adjacent lower group. Evaluated arithmetically, $x \vee y = x + y - xy$ costs one multiplication in addition to the one for $P \wedge G'$. In our setting, the two operands of the $\vee$ are never both 1. A group propagates, i.e., $P = 1$, only if every position in it has digit sum exactly $\alpha - 1$, and then no position generates a carry, i.e., $G = 0$. Hence $P \wedge G' = 1$ implies $G = 0$, and the $\vee$ reduces to an addition (Line 14). Counting each plaintext multiplication as one level, each iteration of the loop then has depth 2, and Algorithm 4 has a total depth of $3\mathsf{d}_{\mathsf{func}} + 2\log_2(l) + 2$.

Algorithm 4: SIMD digit decomposition SimdDigitDec

Input: Decomposition basis α with $q = \alpha^l$, l a power of two
Input: Bound $\beta < \alpha - 1$ with $\|\vec{s}\|_0 \le \beta$
Input: LWE samples $c^{(i)} = \mathsf{LWE}_{\vec{s}}^{n,q}(m^{(i)})$, $0 \le i < \frac{N_s}{l}$
Output: SIMD ciphertext C_{out} whose slot $li + j$ encrypts $m_j^{(i)}$, the j-th digit of $\varphi(c^{(i)})$

1 **for** $i = 0$ to $\frac{N_s}{l} - 1$ **do**
2 $\quad (c^{(i,j)})_{0 \le j < l} \leftarrow \mathsf{PhaseDecomp}(c^{(i)}, \alpha, \beta)$
3 **end**
4 $C \leftarrow \mathsf{Pack}\big((c^{(i,j)})_{i,j}\big)$ // slot $li + j$: $\varphi(c^{(i,j)})$
5 $M(x) = [x]_\alpha,\ T(x) = \left\lfloor \frac{x}{\alpha} \right\rfloor$
6 $P_{\mathsf{init}}(x) = [x = \alpha - 1],\ G_{\mathsf{init}}(x) = [x \ge \alpha]$
7 $C_X \leftarrow \mathsf{EvalFunc}(C, M)$
8 $C_Y \leftarrow \mathsf{Rotate}(\mathsf{EvalFunc}(C, T), 1) \boxdot \vec{\mu}^{(1)}$
9 $C_S \leftarrow C_X + C_Y$
10 $C_P \leftarrow \mathsf{EvalFunc}(C_S, P_{\mathsf{init}}),\ C_G \leftarrow \mathsf{EvalFunc}(C_S, G_{\mathsf{init}})$
11 **for** $k = 0$ to $\log_2(l) - 1$ **do**
12 $\quad C'_P \leftarrow \mathsf{Rotate}(C_P, 2^k) \boxdot \vec{\mu}^{(2^k)} + \vec{o}^{(2^k)}$
13 $\quad C'_G \leftarrow \mathsf{Rotate}(C_G, 2^k) \boxdot \vec{\mu}^{(2^k)}$
14 $\quad C_G \leftarrow C_G + C_P \cdot C'_G$
15 $\quad C_P \leftarrow C_P \cdot C'_P$
16 **end**
17 $C_{\mathsf{Carry}} \leftarrow \mathsf{Rotate}(C_G, 1) \boxdot \vec{\mu}^{(1)}$
18 $C_{\mathsf{out}} \leftarrow \mathsf{EvalFunc}(C_S + C_{\mathsf{Carry}}, M)$
Result: C_{out}

Here, Pack converts the pseudo-LWE samples into a single SIMD ciphertext whose slots contain their phases, e.g., through a homomorphic linear transformation with an encryption of $\vec{s}$ [39]. All values passed to EvalFunc lie in $[0, \alpha^2 - 1]$: the phases

are at most $\eta < \alpha^2$, and the digit sums are at most $2\alpha - 2$. After the prefix computation, slot j of C_G holds the carry out of positions $\lfloor j/l \rfloor l, \ldots, j$ within its block, so C_{Carry} holds the carry into each position.

We note that if the scaling factor Δ of the input is a power of α, both digit decomposition algorithms also act as a bootstrapping procedure. The digits below Δ contain only the error and can be discarded, and the remaining digits can be recombined into a refreshed high-precision ciphertext. Conversions from arithmetic to digit or binary representations are also central to recent work on homomorphic ALUs [17]–[20], where the most efficient approaches rely on special message encodings. Our approach may be of independent interest in that setting: it requires no special encoding and does not depend on the scheme used for the decomposition.

Relation to [15]: Concurrently, Cheon et al. [15] proposed a parallel digit decomposition for CKKS with the same structure. Each digit position is described by a window of two neighboring digits, all carries lie in $\{0, 1\}$, and the carries are computed by a Kogge–Stone prefix scan. Their windows are obtained by scaling a CKKS ciphertext and discretizing it, which introduces a bounded displacement E and requires a radix larger than $4E$, the analogue of our condition $\alpha > \beta + 1$. In our construction, the digit pairs (x_i, y_i) are given exactly by the phases of the pseudo-LWE samples, and Algorithm 4 only assumes an interface for function evaluation.

B. Full-Ring Bootstrapping

We now show that phase decomposition and homomorphic truncation also enlarge the plaintext space of functional bootstrapping in accumulator-based schemes. Standard bootstrapping first switches the modulus to $2N$, and the rounding error of this step occupies the low-order part of the phase. Our method instead evaluates functions with message space equal to $\mathbb{Z}_{2N}$. We also sketch how the same tools yield a full-domain bootstrapping.

Let $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m; \Delta)$ with $q = 2N\Delta$, Δ a power of two, $\|\vec{s}\|_0 \le \beta$, and $\Delta(\beta + 1) \le N$. Then $m \equiv \lfloor (\varphi(c) + \Delta/2)/\Delta \rfloor \pmod{2N}$ if the error is smaller than $\Delta/2$ in absolute value. We perform a two-digit phase decomposition with radix Δ for the lower digit:

$$\begin{aligned} c' &= c + \left[\vec{0}, \frac{\Delta}{2}\right] \pmod{q}, \\ c_{\mathsf{lo}} &= [c']_{\Delta} + [\vec{0}, \Delta\beta], \\ c_{\mathsf{hi}} &= \left\lfloor \frac{c'}{\Delta} \right\rfloor - [\vec{0}, \beta] \pmod{2N}. \end{aligned}$$

Here, c_{lo} is a pseudo-LWE sample, and by Proposition 1, its phase lies in $[\beta, \Delta(\beta + 1) - 1] \subseteq [0, N)$. The sample c_{hi} is an LWE sample modulo $2N$. The shift by $\Delta\beta$ makes the phase of c_{lo} non-negative and increases $\lfloor \varphi(c_{\mathsf{lo}})/\Delta \rfloor$ by exactly β, which the subtraction of β in c_{hi} compensates. Writing $h = \lfloor \varphi(c_{\mathsf{lo}})/\Delta \rfloor$, one verifies as in Theorem 1 that

$$h + \varphi(c_{\mathsf{hi}}) \equiv \left\lfloor \frac{\varphi(c')}{\Delta} \right\rfloor \equiv m \pmod{2N}.$$

With $\Delta_Q = \frac{Q}{2N}$, we obtain

$$\begin{aligned} C' &= \mathsf{BlindRotate}([\mathfrak{o}, \Delta_Q], c_{\mathsf{lo}}) \\ &= \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{\varphi(c_{\mathsf{lo}})}; \Delta_Q), \\ C'' &= \mathsf{HomTruncTrace}(C', \Delta) = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{h}; \Delta_Q), \\ C''' &= \mathsf{BlindRotate}(C'', c_{\mathsf{hi}}) \\ &= \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{h+\varphi(c_{\mathsf{hi}})}; \Delta_Q) = \mathsf{RLWE}_{\mathfrak{s}}^{N,Q}(X^{m}; \Delta_Q). \end{aligned} \tag{9}$$

A negacyclic function $f : \mathbb{Z}_{2N} \to \mathbb{Z}$, i.e., one with $f(x + N) = -f(x)$, is evaluated by multiplying C'' with $\mathfrak{p} = \sum_{j=0}^{N-1} f(j) X^{-j}$ before the final blind rotation and extracting the constant coefficient. Algorithm 5 summarizes the procedure.

Relation to full-domain bootstrapping: Many works [7], [11], [12] circumvent the negacyclicity of $\mathcal{R}$ to evaluate arbitrary functions rather than only negacyclic ones. Algorithm 5 by itself does not achieve this, since the exponent $h + \varphi(c_{\mathsf{hi}})$ of the final blind rotation can take any value modulo $2N$. Algorithm 3 with two digits, however, does: let $q = \alpha^2$ and $\Delta = \alpha$, so that, after adding $\Delta/2$, the message is the upper digit $m_1 = [g_1]_\alpha$. By Lemma 1, the exponent g_1 of the final accumulator lies in $[0, N)$. Replacing $\mathfrak{p} \bmod$ in the last iteration by $\sum_{j=0}^{N-1} f([j]_\alpha) X^{-j}$, centered around the mean of f as above, therefore evaluates an arbitrary function $f : \mathbb{Z}_\alpha \to \mathbb{Z}$ on m_1, with no restriction to negacyclic functions, at the cost of two blind rotations and one truncation.

Relation to Meta-PBS: The recent Meta-PBS [9] pursues a similar goal: functional bootstrapping with a plaintext modulus of up to $2N$ without enlarging the ring dimension. It splits the input with a homomorphic Euclidean division into a quotient modulo $2N$ and remainders whose phases encode the modulus-switching error, which can be viewed as a mixed-radix phase

Algorithm 5: FullRingBoot

Input: LWE sample $\mathsf{ct} = \mathsf{LWE}_{\vec{s}}^{n,q}(\Delta m + e)$ with $q = 2N\Delta$ and $|e| < \Delta/2$
Input: Negacyclic function $f : \mathbb{Z}_{2N} \to \mathbb{Z}$
Output: LWE sample $\mathsf{ct}_{\mathsf{out}} = \mathsf{LWE}_{\mathsf{s}}^{N,Q}(\Delta_Q f(m) + e')$

$\mathsf{ct}' \leftarrow \mathsf{ct} + [\vec{0}, \Delta/2] \pmod q$
$\mathsf{ct}_{\mathsf{lo}} \leftarrow [\mathsf{ct}']_{\Delta} + [\vec{0}, \Delta\beta]$
$\mathsf{ct}_{\mathsf{hi}} \leftarrow \left\lfloor \frac{\mathsf{ct}'}{\Delta} \right\rfloor - [\vec{0}, \beta] \pmod{2N}$
$\mathsf{acc} \leftarrow [\mathsf{o}, \Delta_Q]$
$\mathsf{p} \leftarrow \sum_{j=0}^{N-1} f(j) X^{-j}$
$\mathsf{acc} \leftarrow \mathsf{BlindRotate}(\mathsf{acc}, \mathsf{ct}_{\mathsf{lo}})$
$\mathsf{acc} \leftarrow \mathsf{HomTruncTrace}(\mathsf{acc}, \Delta)$
$\mathsf{acc} \leftarrow \mathsf{p} \cdot \mathsf{acc}$
$\mathsf{acc} \leftarrow \mathsf{BlindRotate}(\mathsf{acc}, \mathsf{ct}_{\mathsf{hi}})$
$\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{SampleExtract}(\mathsf{acc}, 0)$

Result: $\mathsf{ct}_{\mathsf{out}}$

decomposition with centered remainders. In contrast to Algorithm 5, it proceeds from the most significant part. The first blind rotation uses the quotient, and each following blind rotation refines the perturbation of the exponent, while a packing step amplifies the redundancy of the test vector. Its bounds on the perturbation are probabilistic and rely on a Gaussian model of the modulus-switching error under a dense key. Algorithm 5 instead relies on the deterministic bounds of Section II for sparse keys and truncates exactly in the exponent.

IV. Extensions and Comparison

In this section, we first discuss optimizations and extensions of the digit decomposition of Algorithm 3 (Section IV-A). We then compare it theoretically to the digit decomposition of [8] (Section IV-B). We restrict the comparison to [8] because it addresses the same task, the decomposition of an LWE ciphertext into encrypted digits, in the same setting. Furthermore, we discuss [10], which makes different assumptions, at the end of Section IV-B. Works in other settings, such as Meta-PBS [9] and parallel digit decomposition in CKKS [15], are discussed in Section I-A and Section III.

A. Optimizations and Extensions of Digit Decomposition

a) Restarting: By Theorem 4, the noise variance of the accumulator grows by a factor of about α per digit. One option is to choose parameters that are large enough for all digits. This is inefficient, and it couples the LWE parameters, through the number of digits l, to the blind-rotation parameters, which we want to avoid. Instead, we restart the accumulator every ρ digits. Assume that after the blind rotation in iteration i, acc encrypts $\Delta_Q X^{g_i}$. Instead of truncating, we proceed as follows.

1) *Extract the carry.* Since $g_i < N$ (Lemma 1), multiplying acc by the centered test polynomial $\sum_{j=0}^{N-1}(\lfloor j/\alpha \rfloor - \lfloor \beta/2 \rfloor) X^{-j}$, extracting the constant coefficient, and adding $\Delta_Q \lfloor \beta/2 \rfloor$ yields an LWE sample of $h_{i+1} = \lfloor g_i/\alpha \rfloor$.
2) *Switch.* Key switching to $\vec{s}$ and modulus switching to $2N$ yields $c_h = \mathsf{LWE}_{\vec{s}}^{n,2N}(h_{i+1}; \Delta_h)$ with noise $|e| < \Delta_h/2$. Adding $[\vec{0}, \Delta_h/2]$ makes the noise non-negative, so that $\varphi(c_h) = \Delta_h h_{i+1} + e'$ with $e' \in [0, \Delta_h)$.
3) *Re-enter the exponent.* If $\Delta_h(\beta + 1) \le N$, then $\varphi(c_h) < N$ by Lemma 1. A blind rotation of a fresh accumulator $[\mathsf{o}, \Delta_Q]$ with c_h therefore yields $\Delta_Q X^{\varphi(c_h)}$ without sign flip, and $\mathsf{HomTruncTrace}$ with truncation factor Δ_h yields a fresh encryption of $\Delta_Q X^{h_{i+1}}$.

The decomposition then continues with the blind rotation on $c_\alpha^{(i+1)}$. Each restart costs one additional blind rotation and one key switch, and it replaces the truncation by α with a truncation by Δ_h. After a restart, the noise variance of the accumulator is that of a single blind rotation followed by a truncation. A restart reintroduces one source of failure: if the noise of c_h exceeds $\Delta_h/2$, the carry is wrong and the error propagates to all higher digits. This mirrors the behavior of [8], but only at the restart points.

b) Ternary keys: Proposition 1 extends to ternary keys. Let β_+ and β_- denote the number of entries $+1$ and -1 of $\vec{s}$, with $\beta_+ + \beta_- \le \beta$. Then

$$-(\alpha - 1)\beta_+ \le \varphi(c_\alpha^{(i)}) \le (\alpha - 1)(1 + \beta_-).$$

The phase still lies in an interval of length $(\alpha - 1)(\beta + 1)$. If β_+ is known, e.g., because the key is sampled with a fixed number of $+1$ and -1 entries, the shift $P = (\alpha - 1)\beta_+$ in Algorithm 1 keeps the bound η of Theorem 1. If only β is known, the shift $P = (\alpha - 1)\beta$ doubles the length of the interval to be accommodated.

c) Dense keys: Algorithm 3 requires a key of Hamming weight at most β with $\alpha(\beta+1) \le N$. For the sake of completeness, we sketch how to handle keys of larger weight but make on statement regarding its optimality. Partition the indices $[1, n]$ into Γ blocks $I_1, \dots, I_\Gamma$ such that $\|\vec{s}_{I_j}\|_0 \le \beta$ for all j. For a uniform binary key, blocks of size β guarantee this, giving $\Gamma = \lceil n/\beta \rceil$, while larger blocks satisfy the bound only with high probability. Let $c_j = [\vec{a}_{I_j}, b_j]$ with $b_1 = b$ and $b_j = 0$ for $j > 1$. Then $\varphi(c) \equiv \sum_{j=1}^{\Gamma} \varphi(c_j) \pmod q$, and each c_j satisfies the requirements of Algorithm 3. The Γ decompositions are independent and can be computed in parallel. Each yields the base-α digits of $\varphi(c_j)$, which must then be added homomorphically.

A conventional adder propagates carries through all l positions. Instead, we suggest to use carry-free signed-digit addition [40]. Digits in $[0, \alpha-1]$ are valid digits of the redundant signed-digit set $D = [-(\alpha-1), \alpha-1]$. To add two numbers with digits $x_k, y_k \in D$, compute for every position k the sum $w_k = x_k + y_k$, a transfer digit $\theta_{k+1} \in \{-1, 0, 1\}$ with $\theta_{k+1} = 1$ if $w_k \ge \alpha - 1$ and $\theta_{k+1} = -1$ if $w_k \le -(\alpha-1)$, and the interim sum $u_k = w_k - \alpha\theta_{k+1} \in [-(\alpha-2), \alpha-2]$. The result digit $z_k = u_k + \theta_k$ then lies in D. Each position requires one homomorphic evaluation of the (non-negacyclic) transfer function on w_k, and all positions are independent. Adding the Γ digit vectors in a binary tree thus has depth $\lceil \log_2 \Gamma \rceil$ function evaluations. A final conversion from D to the standard digit set requires one carry-propagating step, e.g., with a parallel-prefix adder.

Let c_{f} denote the cost of one evaluation of an arbitrary function on an encrypted digit, which is about two blind rotations with [8]. The total cost is then about $\Gamma \cdot l$ blind rotations and $\Gamma(l-1)$ truncations for the decompositions, plus $(\Gamma-1) \cdot l \cdot c_{\mathsf{f}}$ for the additions and $O(l \cdot c_{\mathsf{f}})$ for the conversion. In addition, the digits must be key- and modulus-switched before they enter the adder. The total work therefore grows linearly in Γ, and the approach is competitive only for small Γ, i.e., for keys of moderate weight. Its latency, however, grows only logarithmically in Γ.

d) Probabilistic phase bounds: Proposition 1 and Lemma 1 are worst-case bounds. The phase of a pseudo-LWE sample is, however, far more concentrated. The decomposition $\bar{c} \mapsto \lfloor \bar{c}/\alpha \rfloor$ in Algorithm 1 is a modulus switch from q to q/α by flooring, and the phase of $c_\alpha^{(i)}$ is exactly the overflow that such a modulus switch discards. For $i = 0$, let $x = [\varphi(\bar{c})]_\alpha$ and $S = \sum_{j:s_j=1} a_j^{(0)}$. Then

$$\varphi(c_\alpha^{(0)}) = x - \alpha \left\lfloor \frac{x+S}{\alpha} \right\rfloor + P.$$

If $\vec{a}$ is uniform modulo $q = \alpha^l$, its base-α digits are independent and uniform in $[0, \alpha-1]$. Hence S is a sum of $\|\vec{s}\|_0$ independent discrete uniform variables, whose distribution is given exactly by polynomial coefficients [41]:

$$\Pr[S = y] = \alpha^{-\|\vec{s}\|_0} \binom{\|\vec{s}\|_0}{y}_\alpha,$$

where $\binom{m}{y}_\alpha$ denotes the coefficient of X^y in $(1 + X + \cdots + X^{\alpha-1})^m$. For ternary keys, the same holds after a shift, since $\alpha - 1 - U$ is uniform whenever U is. For digits $i \ge 1$, the b-digits additionally depend on borrows from lower digits, and we heuristically assume the same distribution. Table I compares the worst-case width of the phase interval with the width of the smallest interval that contains the phase with probability at least $1 - 2^{-60}$.

TABLE I: Width of the phase interval of $c_\alpha^{(0)}$: worst case (Proposition 1) versus the smallest interval containing the phase with probability at least $1 - 2^{-60}$, computed exactly from the distribution of S.

α	β	Worst case	$1 - 2^{-60}$	Std. dev.
16	116	1756	865	49.6
32	60	1892	1185	71.5
64	60	3844	2369	143.1
64	30	1954	1601	101.2

Replacing the worst-case bound by the probabilistic one relaxes the condition $\alpha(\beta+1) \le N$. For $N = 2^{11}$ and a union bound over six digits, the Hamming weight can increase from 127 to about 390 for $\alpha = 16$, and from 63 to about 130 for $\alpha = 32$. This improves security at the same performance. The price is that the exponents are correct only with high probability, so a failure would again propagate to higher digits. The same analysis applies to modulus switching in general. It gives sharper estimates of the modulus switching error than the worst-case bound of [8] or approximations based on the Irwin–Hall distribution, and it would also benefit [8]. A comparison must therefore apply it to both methods consistently. We leave a full treatment, including rounding instead of flooring, moduli whose ratio is not a power of two, and non-ternary keys, to future work.

B. Theoretical Comparison to Related Work

We compare our digit decomposition to that of [8], which, to the best of our knowledge, is the state of the art in this setting, and briefly discuss [10] at the end of this section.

TABLE II: Comparison of our digit decomposition (Algorithm 3) to [8] for an input modulus q, ring dimension N, and Hamming weight β of the binary key. r_{LMP} is the output radix of [8], α our radix, $l = \lceil \log_\alpha q \rceil$, and σ_i^2 the accumulator variance from Theorem 4. Counts exclude restarts (Section IV-A).

Work	#Blind rotations	#Truncations	Condition	Output digits	Output noise
[8]	$2\left\lceil \frac{\log_2 q - \log_2 2N}{\log_2 r_{\mathsf{LMP}}} \right\rceil$	–	$N \geq 2r_{\mathsf{LMP}} \max(\beta_{\mathsf{LMP}}, \beta_{\mathsf{MS}})$	message, $\mathsf{LWE}_{\vec{s}}^{n,2N}$	bound β_{LMP}
Ours	l	$l-1$	$\alpha(\beta+1) \leq N$	phase, $\mathsf{LWE}_{\mathfrak{s}}^{N,Q}$	variance $\frac{N(\alpha^2+2)}{12}\sigma_i^2$

We clarify different notational aspects. In [8], α denotes a scaling factor and not the radix. To avoid confusion, we write α_{LMP} for this scaling factor. Each iteration of [8] bootstraps modulo $2N$ and clears $\log_2 r_{\mathsf{LMP}}$ bits, where $r_{\mathsf{LMP}} = 2N/\alpha_{\mathsf{LMP}}$ is the radix of the output digits. The bound β of [8] bounds the bootstrapping noise, and we write β_{LMP} for it. Throughout the following discussion, β denotes the Hamming weight of the binary key. For such a key, the modulus switching error is bounded by $\beta_{\mathsf{MS}} = (\beta+1)/2$ [8, Lemma 1]. We consider an input $c = \mathsf{LWE}_{\vec{s}}^{n,q}(m;\Delta)$ with plaintext modulus $t = q/\Delta$, i.e., a message precision of $\log_2 t$ bits.

1) Comparison to [8]:

a) Output and Cost: Table II summarizes both methods. The method of [8] performs $\lceil (\log_2 q - \log_2 2N)/\log_2 r_{\mathsf{LMP}} \rceil$ calls to its floor function with two blind rotations each. The last digit requires no bootstrapping. Our method performs l blind rotations and $l-1$ truncations, plus one blind rotation per restart.

The two methods do not compute the same output. The method of [8] returns the digits of the message, whereas ours returns all digits of the phase, i.e., the $\lceil \log_\alpha t \rceil$ message digits and, if Δ is a power of α, $\log_\alpha \Delta$ digits that contain the exact noise. Exact noise digits are useful in their own right, e.g., to bootstrap only the noise instead of the whole message, or for inputs in which message and noise cannot be separated, as in CKKS. If only the message digits are needed, the noise digits are an overhead for which both methods pay in different ways: [8] reserves about $\log_2(4\beta_{\mathsf{LMP}})$ bits of headroom in *every* digit, whereas we pay for the noise digits once. Our per-digit headroom of $\log_2(\beta+1)$ bits depends only on the key, not on the bootstrapping noise.

Each truncation costs $\log_2 \alpha + N/\alpha - 1$ automorphism evaluations and requires automorphism keys, which [8] does not need. However, by performing blind-rotation with automorphism based methods, we may re-use the same keys for truncation. Furthermore, [8] requires two or three blind rotations at each iteration ($\mathsf{HomFloor}$ vs. $\mathsf{HomFloorAlt}$). An automorphism evaluation costs at most about as much as half of an external product, the core step in blind-rotation, whereas a blind rotation performs n of them. In practice, for $N = 2^{11}$ and $n = 1800$, a truncation thus costs at most about 7% of a blind rotation for $\alpha = 16$ and 4% for $\alpha = 32$.

The outputs also differ in format. The method of [8] returns LWE samples modulo $2N$ under $\vec{s}$ with noise bounded by β_{LMP}, which can directly be bootstrapped again. Our outputs are LWE samples modulo Q under the ring key $\mathfrak{s}$, with a variance that grows with the digit index. Converting them to the same format requires one key switch and one modulus switch per digit, and restarts bound the variance. Depending on concrete parameters, the result may yield LWE samples already at post-bootstrap noise levels. Otherwise, each digit can be encoded with a scaling factor that leaves its most significant bit zero, key- and modulus-switched and bootstrapped. The zero most significant bit avoids the negacyclicity issue. The total number of blind rotations then equals that of [8], but all refreshing bootstrappings are independent and can be executed in parallel. In such a case, this reduces the latency but not the total amount of work.

b) Conditions: For a key of Hamming weight β, the scaling factor of [8] must satisfy $\alpha_{\mathsf{LMP}} \geq 4\max(\beta_{\mathsf{LMP}}, \beta_{\mathsf{MS}})$ [8, Theorem 2]. With $r_{\mathsf{LMP}} = 2N/\alpha_{\mathsf{LMP}}$, this becomes

$$N \geq \max\left(2r_{\mathsf{LMP}}\beta_{\mathsf{LMP}},\, r_{\mathsf{LMP}}(\beta+1)\right).$$

For the same radix $r_{\mathsf{LMP}} = \alpha$, the condition of [8] therefore implies our condition $\alpha(\beta+1) \leq N$. If the modulus switching error dominates, both conditions coincide. If the bootstrapping noise dominates, ours is strictly weaker. However, we note for the sake of completeness that our comparison uses the worst-case bound on the modulus switching error. With heuristic estimates of order $O(\sqrt{\beta})$, which [8] uses for concrete parameters, the implication does not hold in general, and the probabilistic analysis of Section IV-A should be applied to both methods.

The modulus switching error matters most for the most significant digits of [8]. After each iteration, the bootstrapping output is switched from the ring modulus to the current, decreasing ciphertext modulus. The bootstrapping noise shrinks with this modulus, whereas the modulus switching error does not. For dense keys, whose norm determines this error, the headroom of [8] is therefore determined by the key in every late iteration. Our method requires a sparse key for its main variant. Ternary keys are supported at the same bound if the number of $+1$ entries is known, and dense keys at a higher cost (Section IV-A).

Finally, the floor function of [8] with two bootstrappings requires the input noise to be at most $\alpha_{\mathsf{LMP}}/4$, since each intermediate digit is used to modify the input before the next one is extracted. For inputs with arbitrary noise, e.g., ciphertexts obtained from CKKS by scheme switching, [8] must use its alternative floor function with three bootstrappings per digit. Our method

makes no assumption on the input noise and always returns the digits of the phase, so the advantage in the number of blind rotations approaches a factor of three in this setting.

c) Asymptotic Cost: For large q, the ratio of blind rotations approaches $2\log_2\alpha/\log_2 r_{\mathsf{LMP}}$. However, with growing q our method has additional benefits.

First, the cost of our blind rotations does not depend on q. They always act on pseudo-LWE samples with small integer coefficients, and the ring modulus is chosen only to accommodate the noise. In contrast, [8] adds bootstrapping outputs to the input ciphertext modulo q. Its ring modulus Q' must therefore satisfy $qB_g/Q' \ll 1$ for the gadget base B_g, which forces more gadget digits as q grows. Their own measurements show a cost increase from $442\,\mathrm{ms}$ to $785\,\mathrm{ms}$ per bootstrapping as the modulus grows from 2^{16} to 2^{29} [8]. A prior modulus switch reduces q only to about $\log_2 t$ plus the bits needed for the noise. Since our method never modifies the input after decomposition, the input modulus may even exceed the ring modulus.

Second, our digit size is independent of the bootstrapping noise. The radix of [8] is at most $N/(2\beta_{\mathsf{LMP}})$, because every digit must accommodate two bootstrapping errors. With $N = 2^{11}$ and $\beta_{\mathsf{LMP}} = 128$, this gives 8, or 16 with the average-case optimization of [8]. Our radix is at most $N/(\beta+1)$, which gives 32 for $\beta = 60$.

Third, errors do not propagate in our method. The exponents g_i are computed exactly, without modulus switching or rounding, and the noise only affects the decryption of each output digit individually. In [8], a single bootstrapping error beyond its bound yields a wrong carry and corrupts all higher digits. Restarts reintroduce this effect in our method, but only at the restart points. In addition, the least significant digit is obtained without any bootstrapping, and the last iteration can evaluate an arbitrary function of the most significant digit instead of extracting it (Section III-B). The method of [8] needs a separate function evaluation with two further bootstrappings for this.

2) Comparison to Head Start: The authors of [10] decompose LWE samples into digits starting from the most significant bits. This approach is fundamentally different from ours and from [8]. First, it does not yield an exact decomposition, and making it exact requires additional bootstrappings, after which it has no advantage over [8]. Second, it assumes that the most significant bit of the message is known or enforced to be constant. The authors consider the summation of high-precision values as an application. Maintaining a known most significant bit then requires bootstrapping even when the noise is low, since the sum of t values of k bits can grow to $k + \log_2 t$ bits and overflow into the most significant bit. Neither [8] nor our method has such a restriction. Finally, the comparison in [10, Table 1] states that the input variance of [8] propagates to the output digits. This holds only for the least significant digit, which, depending on the scaling factor, may not even be part of the message. For all other digits, the output noise of [8] is bounded by about twice the bootstrapping noise, and the least significant digit can be bootstrapped at a constant additional cost.

V. Conclusion

In this work, we introduced the notion of phase splitting and described methods to perform exact homomorphic truncation in the exponent. We showed that splitting a LWE ciphertext into smaller elements leads to performance gains for a range of problems, specifically homomorphic digit-decomposition and functional bootstrapping. Yet, several directions for future work remain. On the one hand, we believe that a more efficient truncation procedure must exist. On the other hand, both our digit-decomposition method and full-ring bootstrapping approach perform blind-rotation on pseudo-ciphertext with strongly bounded coefficients. While methods exist that use such a bound to evaluate multiple functions at a lower noise cost, to the best of our knowledge there is no approach that exploits this property to enable faster blind-rotation for a single ciphertext specifically for our case in which the secret is a sparse, binary vector.

References

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *41st ACM STOC*, M. Mitzenmacher, Ed. ACM Press, May / Jun. 2009, pp. 169–178.

[2] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *ASIACRYPT 2016, Part I*, ser. LNCS, J. H. Cheon and T. Takagi, Eds., vol. 10031. Springer, Berlin, Heidelberg, Dec. 2016, pp. 3–33.

[3] ——, "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE," in *ASIACRYPT 2017, Part I*, ser. LNCS, T. Takagi and T. Peyrin, Eds., vol. 10624. Springer, Cham, Dec. 2017, pp. 377–408.

[4] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT 2017, Part I*, ser. LNCS, T. Takagi and T. Peyrin, Eds., vol. 10624. Springer, Cham, Dec. 2017, pp. 409–437.

[5] J. Kang and L. Schild, "Pirouette: Query efficient single-server PIR," Cryptology ePrint Archive, Report 2025/680, 2025. [Online]. Available: https://eprint.iacr.org/2025/680

[6] L. Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *EUROCRYPT 2015, Part I*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9056. Springer, Berlin, Heidelberg, Apr. 2015, pp. 617–640.

[7] I. Chillotti, D. Ligier, J.-B. Orfila, and S. Tap, "Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE," in *ASIACRYPT 2021, Part III*, ser. LNCS, M. Tibouchi and H. Wang, Eds., vol. 13092. Springer, Cham, Dec. 2021, pp. 670–699.

[8] Z. Liu, D. Micciancio, and Y. Polyakov, "Large-precision homomorphic sign evaluation using FHEW/TFHE bootstrapping," in *ASIACRYPT 2022, Part II*, ser. LNCS, S. Agrawal and D. Lin, Eds., vol. 13792. Springer, Cham, Dec. 2022, pp. 130–160.

[9] S. Ma, T. Huang, A. Wang, C. Xu, T. Wei, and X. Wang, "Meta-PBS: Compact high-precision programmable bootstrapping," Cryptology ePrint Archive, Report 2025/2284, 2025. [Online]. Available: https://eprint.iacr.org/2025/2284

[10] J.-P. D'Anvers, X. Pottier, T. de Ruijter, and I. Verbauwhede, "Head start: Digit extraction in TFHE from MSB to LSB," Cryptology ePrint Archive, Report 2025/2012, 2025. [Online]. Available: https://eprint.iacr.org/2025/2012

[11] K. Kluczniak and L. Schild, "FDFB: Full domain functional bootstrapping towards practical fully homomorphic encryption," *IACR TCHES*, vol. 2023, no. 1, pp. 501–537, 2023.

[12] S. Ma, T. Huang, A. Wang, Q. Zhou, and X. Wang, "Fast and accurate: Efficient full-domain functional bootstrap and digit decomposition for homomorphic computation," *IACR TCHES*, vol. 2024, no. 1, pp. 592–616, 2024.

[13] K. Lee and J. W. Yoon, "Discretization error reduction for high precision torus fully homomorphic encryption," in *PKC 2023, Part II*, ser. LNCS, A. Boldyreva and V. Kolesnikov, Eds., vol. 13941. Springer, Cham, May 2023, pp. 33–62.

[14] T. Nishimura, K. Tozawa, and K. Sadakane, "Surpassing the word size limitation of TFHE with noise calibration," in *ACM CCS 2025*, C.-Y. Huang, J.-C. Chen, S.-P. Shieh, D. Lie, and V. Cortier, Eds. ACM Press, Oct. 2025, pp. 438–452.

[15] J. H. Cheon, K. Cho, J. Jung, and J. Lim, "Low-latency parallel digit decomposition and sign evaluation for CKKS," Cryptology ePrint Archive, Paper 2026/2368, 2026.

[16] A. Alexandru, A. Kim, and Y. Polyakov, "General functional bootstrapping using CKKS," in *CRYPTO 2025, Part III*, ser. LNCS, Y. T. Kalai and S. F. Kamara, Eds., vol. 16002. Springer, Cham, Aug. 2025, pp. 304–337.

[17] M. Gao and H. Zheng, "FHE for SIMD arithmetic logic units with amortized $O(1)$ bootstrapping per ciphertext," in *Advances in Cryptology – CRYPTO 2026, Part II*, ser. LNCS, vol. 16801. Springer, 2026, pp. 299–330.

[18] Y. Wei, K. Ma, M. Gao, and H. Zheng, "FHE for ALU over large prime moduli and application to one-round threshold ECDSA," Cryptology ePrint Archive, Paper 2026/1836, 2026.

[19] X. Shen, Z. Li, R. Wang, and X. Lu, "Efficient homomorphic arithmetic logical units with $O(1)$ bootstrapping," Cryptology ePrint Archive, Paper 2026/1840, 2026.

[20] J. Liu, S. Ma, A. Wang, and X. Wang, "High-precision homomorphic ALU over arbitrary moduli with $O(1)$ bootstrapping," Cryptology ePrint Archive, Paper 2026/1869, 2026.

[21] J. H. Cheon, W. Cho, J. Kim, and D. Stehlé, "Homomorphic multiple precision multiplication for CKKS and reduced modulus consumption," in *ACM CCS 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM Press, Nov. 2023, pp. 696–710.

[22] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *37th ACM STOC*, H. N. Gabow and R. Fagin, Eds. ACM Press, May 2005, pp. 84–93.

[23] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *EUROCRYPT 2010*, ser. LNCS, H. Gilbert, Ed., vol. 6110. Springer, Berlin, Heidelberg, May / Jun. 2010, pp. 1–23.

[24] ——, "A toolkit for ring-LWE cryptography," in *EUROCRYPT 2013*, ser. LNCS, T. Johansson and P. Q. Nguyen, Eds., vol. 7881. Springer, Berlin, Heidelberg, May 2013, pp. 35–54.

[25] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *J. Math. Cryptol.*, vol. 9, no. 3, pp. 169–203, 2015. [Online]. Available: http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml

[26] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *CRYPTO 2013, Part I*, ser. LNCS, R. Canetti and J. A. Garay, Eds., vol. 8042. Springer, Berlin, Heidelberg, Aug. 2013, pp. 75–92.

[27] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, Jan. 2020.

[28] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart, "Ring switching in BGV-style homomorphic encryption," in *SCN 12*, ser. LNCS, I. Visconti and R. De Prisco, Eds., vol. 7485. Springer, Berlin, Heidelberg, Sep. 2012, pp. 19–37.

[29] S. Halevi and V. Shoup, "Algorithms in HElib," in *CRYPTO 2014, Part I*, ser. LNCS, J. A. Garay and R. Gennaro, Eds., vol. 8616. Springer, Berlin, Heidelberg, Aug. 2014, pp. 554–571.

[30] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient homomorphic conversion between (ring) LWE ciphertexts," in *ACNS 2021, Part I*, ser. LNCS, K. Sako and N. O. Tippenhauer, Eds., vol. 12726. Springer, Cham, Jun. 2021, pp. 460–479.

[31] D. Micciancio and Y. Polyakov, "Bootstrapping in FHEW-like cryptosystems," in *WAHC '21: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Virtual Event, Korea, 15 November 2021*. WAHC@ACM, 2021, pp. 17–28.

[32] K. Cong, D. Das, J. Park, and H. V. L. Pereira, "SortingHat: Efficient private decision tree evaluation via homomorphic encryption and transciphering," in *ACM CCS 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM Press, Nov. 2022, pp. 563–577.

[33] R. Wang, Y. Wen, Z. Li, X. Lu, B. Wei, K. Liu, and K. Wang, "Circuit bootstrapping: Faster and smaller," in *EUROCRYPT 2024, Part II*, ser. LNCS, M. Joye and G. Leander, Eds., vol. 14652. Springer, Cham, May 2024, pp. 342–372.

[34] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption library," https://tfhe.github.io/tfhe/, 2016.

[35] A. Guimarães, E. Borin, and D. F. Aranha, "Revisiting the functional bootstrap in TFHE," *IACR TCHES*, vol. 2021, no. 2, pp. 229–253, 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8793

[36] K. H. Lee and J. W. Yoon, "Homomorphic field trace revisited: Breaking the cubic noise barrier," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2026, no. 1, pp. 105–134, 2026.

[37] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786–793, 1973.

[38] J. Sklansky, "Conditional-sum addition logic," *IRE Transactions on Electronic Computers*, vol. EC-9, no. 2, pp. 226–231, 1960.

[39] W. jie Lu, Z. Huang, C. Hong, Y. Ma, and H. Qu, "PEGASUS: Bridging polynomial and non-polynomial evaluations in homomorphic encryption," in *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021, pp. 1057–1073.

[40] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 389–400, 1961.

[41] C. Caiado and P. Rathie, "Polynomial coefficients and distribution of the sum of discrete uniform variables," Eighth Annual Conference of the Society of Special Functions and their Applications, 2007. [Online]. Available: https://durham-repository.worktribe.com/output/1158118