Faster Bootstrapping for Fully Homomorphic Encryption over Mixed-Radix Cyclotomic Ring

Jung Hee Cheon[1,2], Seungwan Hong[3], Minsik Kang[4*], Taeseong Kim[1], Jonghyun Kim[5], Changmin Lee[5] and Junho Lee[1]

[1] Seoul National University, Seoul, Republic of Korea, jhcheon@snu.ac.kr, kts1023@snu.ac.kr, abab9579@snu.ac.kr
[2] CryptoLab Inc., Seoul, Republic of Korea
[3] Yale University, New Haven, CT, USA, seungwan.hong@yale.edu
[4] Sejong University, Seoul, Republic of Korea, mskang@sejong.ac.kr
[5] Korea University, Seoul, Republic of Korea, yoswuk@korea.ac.kr, changminlee@korea.ac.kr

Abstract. Homomorphic matrix arithmetic is central to many data-intensive fully homomorphic encryption (FHE) applications, including privacy-preserving machine learning. Standard CKKS over power-of-two cyclotomic rings provides only a one-dimensional slot structure, forcing encrypted matrix operations to rely heavily on cyclic slot rotations and the associated key-switching overhead. Although the multivariate cyclotomic ring introduced by Jang et al. (AsiaCCS 2022) and later leveraged by Gentry–Lee (Crypto 2026) naturally supports matrix-structured computations, existing implementations still rely on Number-Theoretic Transforms (NTTs) based on Rader's algorithm for the non-power-of-two dimensions, making homomorphic arithmetic inefficient. Moreover, they lack an efficient bootstrapping algorithm.

We instantiate CKKS over a mixed-radix cyclotomic ring whose order has 2 and 3 as prime factors, yielding a bivariate slot structure with independent row and column rotations. By realizing efficient NTT-based polynomial arithmetic over this ring and developing homomorphic radix-3 FFT algorithms for the slot-to-coefficient (StC) and coefficient-to-slot (CtS) transformations in bootstrapping, we obtain the first practical end-to-end implementation of CKKS over this mixed-radix ring, thereby broadening the parameter space of CKKS beyond conventional power-of-two ring degrees.

Based on this foundation, we further present an efficient bootstrapping algorithm for Gentry–Lee. Our mixed-radix decomposition factors the Discrete Fourier Transform and its inverse underlying StC and CtS into their radix-2 and radix-3 local counterparts. Exploiting the inherent batch structure of Gentry–Lee, we evaluate these components as batch plaintext–ciphertext matrix multiplications. This formulation replaces rotation-heavy linear transformations with matrix multiplications over cleartexts that can be accelerated by high-performance libraries such as BLAS and FLINT. Our proof-of-concept implementation achieves up to 5.31× and 2.96× speedups in amortized runtime for CtS and for full bootstrapping, respectively, over our optimized single bootstrapping, and is up to 1.87× and 1.39× faster than the construction of Crockett et al. (Crypto 2026).

Keywords: Homomorphic Encryption · Encrypted Matrix Multiplication · Mixed-Radix Cyclotomic Ring · Bootstrapping

1 Introduction

Fully homomorphic encryption (FHE) is a cryptographic primitive that allows computation on encrypted data without decryption. Following Gentry's breakthrough construc-

*Corresponding author.

tion [Gen09], many modern FHE schemes have been built on the learning-with-errors (LWE) problem [Reg09] or its ring variant, ring learning with errors (RLWE) [LPR10]. Among them, the Cheon–Kim–Kim–Song (CKKS) scheme [CKKS17] is an RLWE-based scheme that supports approximate arithmetic over real or complex numbers and SIMD (Single Instruction Multiple Data) operations, enabling parallel computation on multiple data elements within a single ciphertext. CKKS implementations in modern FHE libraries, including SEAL [SEA26], OpenFHE [BBB+22], Lattigo [Lat24], and HEaaN [Cry22], generally use the power-of-two cyclotomic ring $R = \mathbb{Z}[X]/(X^N + 1)$, where the ring degree N is a power of two. This ring enables efficient polynomial arithmetic through a Number-Theoretic Transform (NTT) based on the radix-2 Cooley–Tukey algorithm.

Encrypted matrix multiplication (MM) is one of the most fundamental operations in privacy-preserving machine learning (PPML). As concerns over privacy leakage from large language models (LLMs) continue to grow, FHE has emerged as a promising approach for enabling evaluation of LLMs on encrypted user data [MYJK25, RKP+25, PPP+26]. Most FHE-based LLM systems are built on the CKKS scheme because its approximate arithmetic efficiently supports the real-valued computations required by neural networks. In these systems, matrix multiplication accounts for a substantial portion of the computational cost: plaintext-ciphertext matrix multiplication (PCMM) is used for linear transformations with model parameters, while ciphertext-ciphertext matrix multiplication (CCMM) arises from pairwise interactions among encrypted representations, such as those in attention mechanisms. Therefore, improving the performance of encrypted matrix multiplication is crucial for accelerating practical FHE-based LLM evaluation.

However, conventional CKKS instantiated over power-of-two cyclotomic rings exposes only a one-dimensional cyclic slot structure, providing no native representation of matrices. Consequently, encrypted MM must be realized by emulating matrix operations through one-dimensional cyclic rotations, Hadamard products, and carefully designed packing strategies [HS14, HS18, JKLS18, ZsHCJ24, MYJK25]. Since each cyclic rotation requires an expensive key-switching operation, these approaches incur substantial overhead.

To overcome this limitation, recent works have explored FHE schemes based on multivariate rings, which naturally expose multi-dimensional slot structures. Unlike the one-dimensional cyclic layout of conventional CKKS, these schemes support independent rotations along each dimension, enabling matrices to be represented and manipulated directly within ciphertext slots. Jang et al. [JLK+22] first introduced this idea in MatHEAAN, a multivariate RLWE-based extension of CKKS that supports matrix-packed ciphertexts and efficient row and column rotations. Building on this capability, Gentry and Lee [GL26] proposed a matrix-arithmetic FHE scheme over bivariate cyclotomic rings that leverages reduction-based matrix multiplication. By utilizing the non-power-of-two factor of the cyclotomic order as the batch dimension, their construction reduces batched encrypted MM to a small number of plaintext matrix multiplications over ciphertext components, generalizing previous reduction-based MM algorithms [BCH+24, Par25] while decoupling the matrix dimension from the ring degree.

However, these matrix-oriented ring structures come at the cost of substantially more expensive polynomial arithmetic. Unlike conventional CKKS over power-of-two cyclotomic rings, the non-power-of-two factors in the cyclotomic order prevent the use of highly optimized Cooley–Tukey-based NTT algorithms. Instead, previous implementations [JLK+22, GL26] evaluate the corresponding prime-length transforms using Rader's algorithm [Rad68], which converts each transform into a cyclic convolution. Even with precomputed convolution kernels, this approach still requires one forward and one inverse power-of-two NTT for every prime-length transform. Since this overhead is incurred at every modulus in the RNS representation, arithmetic over matrix-oriented rings remains substantially slower than that of conventional power-of-two CKKS.

Beyond efficient polynomial arithmetic, practical bootstrapping remains another major

challenge for matrix-oriented FHE schemes. Bootstrapping relies on the slot-to-coefficient (StC) and coefficient-to-slot (CtS), which homomorphically evaluate the forward and inverse discrete Fourier transforms, respectively. MatHEAAN does not provide an efficient FFT-based realization of these transformations, and its reported implementation requires 22.30 minutes to perform a single bootstrapping operation on a single-thread CPU [JLK+22]. Gentry–Lee likewise adopts the MatHEAAN bootstrapping procedure across multiple batches without exploiting its reduction-based matrix multiplication framework to accelerate StC and CtS. Consequently, despite enabling efficient homomorphic matrix arithmetic, existing matrix-oriented FHE schemes still lack a practical bootstrapping.

1.1 Our Contributions

In this work, we instantiate CKKS over the mixed-radix cyclotomic ring

$$R = \mathbb{Z}[X]/(X^N - X^{N/2} + 1) = \mathbb{Z}[X]/(\Phi_{3N}(X)),$$

whose cyclotomic order is $3N$ and whose ring degree N has 2 and 3 as prime factors. Prior lattice-based schemes, including NTTRU [LS19] and NTRU+ [KP23], employed this mixed-radix ring primarily to obtain finer-grained ring degrees while retaining efficient NTT-based polynomial arithmetic.

We observe that the ring inherently admits a bivariate representation, induced by the coprime power-of-two and power-of-three components of its cyclotomic order, with independent row and column rotations. Exploiting this structure for homomorphic matrix arithmetic, we provide, to the best of our knowledge, the first practical end-to-end CKKS instantiation beyond power-of-two cyclotomic rings. Built on top of the Lattigo FHE library [Lat24], our library supports fast homomorphic operations through an efficient NTT implementation, together with efficient key-switching operations for row and column rotations, matrix arithmetic, and bootstrapping for both MatHEAAN and Gentry–Lee.

Homomorphic Mixed-Radix FFT and Bootstrapping. We develop an explicit mixed-radix Cooley–Tukey-style FFT algorithm for the cyclotomic trinomial ring and realize it homomorphically as the StC and CtS procedures during bootstrapping. Our construction evaluates these linear transformations with $O(r \cdot N^{1/(2r)})$ key-switching operations at multiplicative depth r, the same asymptotic cost as radix-2 CKKS [CCS19, HHC19], thereby enabling efficient MatHEAAN and Gentry-Lee bootstrapping.

Our implementation achieves 0.96×—1.35× the throughput of conventional power-of-two CKKS bootstrapping, showing that the mixed-radix setting can provide comparable bootstrapping performance. This establishes a practical extension of CKKS beyond the conventional power-of-two cyclotomic setting, enabling bootstrapping at a ring degree as low as $N = 2^8 \cdot 3^4 = 20{,}736$, whereas conventional CKKS bootstrapping generally requires a ring degree of at least $N = 2^{15}$ (for real data). The resulting parameter flexibility allows the ring parameters to be selected based on both the required ciphertext modulus and the need for independent row and column rotations in matrix-oriented workloads.

Fast Bootstrapping for Gentry–Lee [GL26]. We propose a bootstrapping algorithm for Gentry–Lee that replaces the rotation-intensive linear transforms with reduction-based batch MM. When expanded along its extension variable, a Gentry–Lee ciphertext comprises a batch of ciphertexts over the mixed-radix cyclotomic base ring that share the same secret key. We exploit this structure to evaluate the DFT and iDFT layers of StC and CtS jointly across the batch, rather than bootstrapping its constituent ciphertexts independently.

The CRT decomposition separates each transform into a radix-2 factor and a radix-3 factor, but the two factors require different algebraic realizations. The radix-2 factor is handled in the original trivariate extension ring of Gentry–Lee, whose matrix axes support the batch left multiplication. The radix-3 factor acts along the batching axis of that ring and

therefore does not admit the Gentry–Lee reduction in its original representation. To handle this factor, we introduce a distinct auxiliary extension ring over the cyclotomic trinomial component, in which the relevant axis becomes a matrix dimension. The resulting batch MMs are evaluated using the reduction-based MM algorithms of [GL26, CKL26] and can be accelerated by high-performance MM libraries such as BLAS [bla25] or FLINT [The25]. Moreover, the homomorphic DFT for StC consumes only two multiplicative levels, while the inverse DFT for CtS can be evaluated using a single level, compared with at least three levels per transform in standard CKKS bootstrapping.

Our proof-of-concept implementation achieves throughput speedups of 2.64× to 5.31× for CtS, 3.51× to 5.66× for StC, and 1.90× to 2.96× for the overall bootstrapping throughput, all relative to a baseline that sequentially applies our FFT-accelerated MatHEAAN bootstrapping to each batch.

1.2 Technical Overview

Homomorphic Mixed-Radix FFT. Let $3N = m_1 m_2$, where m_1 and m_2 are powers of two and three, respectively. We set $n_1 = \phi(m_1)/2$, $n_2 = \phi(m_2)$, and $n = N/2 = n_1 n_2$. Note that the trinomial ring R is a cyclotomic ring of order $3N$:

$$R = \mathbb{Z}[X]/(X^N - X^{N/2} + 1) = \mathbb{Z}[X]/(\Phi_{3N}(X)),$$

whose automorphisms are induced by elements of $\mathbb{Z}_{3N}^\times \cong \mathbb{Z}_{m_1}^\times \times \mathbb{Z}_{m_2}^\times$. The slot structure is induced by the DFT map

$$\mathsf{DFT} : \mathbb{R}[X]/(\Phi_{3N}(X)) \longrightarrow \mathbb{C}^{N/2},$$

whose matrix representation with respect to the monomial basis is a Vandermonde matrix. We choose generators $g_1, g_2 \in \mathbb{Z}_{3N}^\times$ such that the induced slot ordering simultaneously supports independent row and column rotations over $\mathbb{C}^{N/2} = \mathbb{C}^{n_1 \times n_2}$ and yields

$$\mathsf{DFT} = \big[\, \boldsymbol{D} \mid \omega_6 \boldsymbol{D} \,\big], \qquad \boldsymbol{D} \in \mathcal{M}_n(\mathbb{C}).$$

Moreover, the resulting square matrix $\boldsymbol{D}$ admits a mixed-radix Cooley–Tukey-style decomposition, allowing an efficient FFT algorithm and its homomorphic evaluation over the NTT-efficient univariate ring.

Fast Bootstrapping for Gentry–Lee. Gentry–Lee [GL26] represents the base ring and defines its ciphertext ring as

$$R \cong \mathcal{R}^{(\mathrm{i})} = \mathbb{Z}[\mathrm{i}][U, W]/(U^{n_1} - \mathrm{i},\ W^{n_2} + W^{n_2/2} + 1),$$
$$\mathcal{S}^{(\mathrm{i})} = \mathcal{R}^{(\mathrm{i})}[V]/(V^{n_1} - \mathrm{i}) = \mathbb{Z}[\mathrm{i}][U, V, W]/(U^{n_1} - \mathrm{i},\ V^{n_1} - \mathrm{i},\ W^{n_2} + W^{n_2/2} + 1).$$

Under coefficient and slot orderings with regard to CRT, DFT over bivariate ring $\mathcal{R}^{(\mathrm{i})}$ can be realized as a two-sided matrix representation

$$\mathsf{DFT}\,(\mu) = \boldsymbol{D}_1\, \boldsymbol{M}\, \boldsymbol{D}_2^T, \tag{1}$$

where $\boldsymbol{M}$ is the bivariate coefficient matrix of $\mu \in R$, and $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are the local DFT matrices associated with the radix-2 and radix-3 components, respectively. Since the secret key s lies in $\mathcal{R}^{(\mathrm{i})}$, writing a ciphertext over $\mathcal{S}^{(\mathrm{i})}$ coefficient-wise in V yields n_1 ciphertexts over $\mathcal{R}^{(\mathrm{i})}$ under the same secret key:

$$b + a \cdot s \approx \mu = \sum_{k \in [n_1]} \mu_k\, V^k \in \mathcal{S}^{(\mathrm{i})} \quad \Longleftrightarrow \quad b_k + a_k \cdot s \approx \mu_k \in \mathcal{R}^{(\mathrm{i})}, \quad k \in \{0, 1, \ldots, n_1 - 1\}.$$

Let $\boldsymbol{M}_k \in \mathbb{C}^{n_1 \times n_2}$ denote the coefficient matrix of μ_k. StC then requires the evaluation of $\boldsymbol{D}_1 \boldsymbol{M}_k \boldsymbol{D}_2^T$ for all $k \in \{0, 1, \ldots, n_1 - 1\}$, while CtS replaces the two local DFT factors by their inverses. Therefore, it suffices to evaluate left PCMM by $\boldsymbol{D}_1 \in \mathbb{C}^{n_1 \times n_1}$ and right PCMM by $\boldsymbol{D}_2^T \in \mathbb{C}^{n_2 \times n_2}$ in a batched manner.

These two multiplications interact differently with the extension-ring structure. The left multiplication by $\boldsymbol{D}_1$ acts along the U-direction. The original Gentry–Lee framework regards the variables U and V, which have the same relation, as matrix axes and applies $\boldsymbol{D}_1$ across the batch axis-W. By contrast, the right multiplication by $\boldsymbol{D}_2^T$ acts along the W-direction, which serves as the batching axis of $\mathcal{S}^{(\mathrm{i})}$. The reduction over $\mathcal{S}^{(\mathrm{i})}$ therefore does not expose the radix-3 direction as a matrix dimension.

To make the right factor $\boldsymbol{D}_2^T$ amenable to reduction-based MM, we pass to an alternative presentation of R and introduce the *auxiliary trivariate extension*

$$R \cong \mathcal{R}^{(\omega)} := \mathbb{Z}[\omega][W, U]/(W^{n_2/2} - \omega,\ U^{2n_1} + 1),$$
$$\mathcal{S}^{(\omega)} := \mathcal{R}^{(\omega)}[Y]/(Y^{n_2/2} - \omega) = \mathbb{Z}[\omega][W, Y, U]/(W^{n_2/2} - \omega,\ Y^{n_2/2} - \omega,\ U^{2n_1} + 1).$$

Indeed, eliminating ω via $\omega = W^{n_2/2}$ identifies its defining relation $\omega^2 + \omega + 1 = 0$ with $W^{n_2} + W^{n_2/2} + 1 = 0$, yielding $\mathcal{R}^{(\omega)} \cong \mathcal{R}' \cong R$.

In $\mathcal{S}^{(\omega)}$, the variables W and Y serve as the two matrix axes, each of dimension $n_2/2$, while U indexes $2n_1$ batched instances. Under this presentation, right multiplication by $\boldsymbol{D}_2^T$ and by $(\boldsymbol{D}_2^T)^{-1}$ can both be realized as batch PCMMs over $\mathcal{S}^{(\omega)}$. Accordingly, StC and CtS use two algebraically distinct instantiations of reduction-based MM [GL26, CKL26]: the $\boldsymbol{D}_1$ factor and its inverse are evaluated over $\mathcal{S}^{(\mathrm{i})}$, whereas the $\boldsymbol{D}_2^T$ factor and its inverse are evaluated over $\mathcal{S}^{(\omega)}$. Both stages replace per-ciphertext rotations with plaintext-side matrix multiplications amenable to BLAS and FLINT. Section 4 gives the full construction.

1.3 Comparison with Crockett et al.

Concurrently, Crockett et al. [CGK+26] independently proposed a bootstrapping algorithm for Gentry–Lee that applies reduction-based MM to the linear transforms in StC and CtS. The main distinction lies in the auxiliary rings used to realize these matrix multiplications. Both works evaluate the $\boldsymbol{D}_1$ factor in the original Gentry–Lee trivariate ring $\mathcal{S}^{(\mathrm{i})}$. To evaluate the $\boldsymbol{D}_2^T$ factor, however, Crockett et al. introduce the auxiliary ring

$$R^{\mathrm{aux}} = \mathbb{Z}[\mathrm{i}][X, W, V]/(X^n - \mathrm{i}, \Phi_p(W), \Phi_p(V)),$$

whereas our construction uses $\mathcal{S}^{(\omega)}$. Although R^{aux} has a simpler presentation than $\mathcal{S}^{(\omega)}$, it remains distinct from the Gentry–Lee ring $\mathcal{S}^{(\mathrm{i})}$. This representation introduces an additional correction term in the reduction to cleartext MM. For prime p, the correction is a rank-one outer product of cost $O(\phi(p)^2)$, negligible compared to $\phi(p) \times \phi(p)$ MM of cost $O(\phi(p)^3)$.

Their theorem and algorithm are stated only for prime p ([CGK+26, Theorem 5]). For an odd prime power $p = r^k$, they note, without providing a corresponding theorem or algorithm, that the correction term has rank r^{k-1} and incurs a cost of $O(\phi(p)^2 r^{k-1})$. In our setting, where $p = 3^{\beta+1}$, this correction costs $\phi(p)^3/2$, thereby increasing the leading cleartext-MM cost from $\phi(p)^3$ to $3\phi(p)^3/2$. By contrast, our construction handles the $\boldsymbol{D}_2^T$ factor without requiring any correction term.

Moreover, we note that our construction provides finer bootstrapping granularity: $\mathcal{S}^{(\omega)}$ packs $n_2/2$ $\mathcal{R}^{(\mathrm{i})}$ ciphertexts, whereas R^{aux} in [CGK+26] packs n_2. Thus, $\mathcal{S}^{(\omega)}$ has half the dimension, reducing zero-padding for small workloads. We refer to the section 5.2 for more details.

2 Preliminaries

2.1 Notation

We denote by $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ the rings of integers, rational numbers, real numbers, and complex numbers, respectively. For a positive integer d, we write $[d] := \{0, 1, \ldots, d-1\}$. Throughout the paper, we fix integers $\alpha \geq 3$ and $\beta \geq 0$, and set $N := 2^\alpha 3^\beta$, $n := N/2$, $n_1 := 2^{\alpha-2}$, and $n_2 := 2 \cdot 3^\beta$, so that $n = n_1 n_2$. Let $\phi(m)$ be Euler's totient function. Let $\Phi_M(X) \in \mathbb{Z}[X]$ be the M-th cyclotomic polynomial.

We work over the cyclotomic ring $\mathcal{R} := \mathbb{Z}[X]/(\Phi_{3N}(X))$ and define $\omega_M := \exp(2\pi\mathrm{i}/M)$ for a positive integer M, $\zeta = \omega_{3N}^{3^{\beta+1}}$, $\eta = \omega_{3N}^{2^\alpha}$, and $\omega = \exp(2\pi\mathrm{i}/3)$ in particular. For a ring A and a positive integer d, we denote by $\mathcal{M}_d(A)$ the ring of all $d \times d$ matrices over A. Vectors are written in bold lowercase, e.g., $\boldsymbol{a}$, and matrices are written in bold uppercase, e.g., $\boldsymbol{M}$. The (j,k)-th entry of a matrix $\boldsymbol{M}$ is denoted by $M_{j,k}$, and for $\boldsymbol{M} \in M_d(\mathbb{C})$, we denote by $\boldsymbol{M}^*$ its conjugate transpose. We denote the Kronecker product of the matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ by $\boldsymbol{M} \otimes \boldsymbol{N}$. For vectors or matrices of the same shape, $\odot$ denotes the Hadamard (entrywise) product. We write $x \leftarrow D$ to denote sampling x from a distribution D. We use χ_σ to denote the centered discrete Gaussian distribution with standard deviation σ, and χ_h to denote the uniform distribution over ternary vectors of Hamming weight h.

2.2 The CKKS scheme and Bootstrapping

The Cheon-Kim-Kim-Song (CKKS) scheme [CKKS17] is an RLWE-based FHE scheme that supports approximate computations on real or complex numbers. For a plaintext space $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ with a power-of-two N, $\mathsf{DFT} : \mathbb{R}[X]/(X^N+1) \to \mathbb{C}^{N/2}$ is defined as

$$\mathsf{DFT}(m(X)) = (m(\omega_{2N}^{5^j}))_{j \in [N/2]}$$

where $\omega_{2N} = \exp(2\pi\mathrm{i}/2N)$, and iDFT is defined as its inverse. Given an integer $\Delta > 0$, the slot-encoding is defined as: $m = \mathsf{Ecd}(\boldsymbol{z}) = \lfloor \Delta \cdot \mathsf{iDFT}(\boldsymbol{z}) \rceil \in \mathcal{R}$ for a message vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$. The decoding is defined as: $\boldsymbol{z} = \mathsf{Dcd}(m) = \mathsf{DFT}(m)/\Delta \in \mathbb{C}^{N/2}$. Note that DFT can be represented as a matrix vector product with a Vandermonde-like matrix $\boldsymbol{V}$ as

$$\boldsymbol{V} = \left(\omega_{2N}^{5^i \cdot j}\right)_{i,j \in [N/2]} \in \mathbb{C}^{(N/2)\times(N/2)}.$$

For a polynomial $m \in \mathbb{R}[X]/(X^N+1)$, we write its coefficient vector $\boldsymbol{m} = (\boldsymbol{m}_0, \boldsymbol{m}_1)$, and we have

$$\boldsymbol{z} = \mathsf{DFT}(\boldsymbol{m}) = \boldsymbol{V}(\boldsymbol{m}_0 + \mathrm{i}\,\boldsymbol{m}_1) \in \mathbb{C}^{N/2},$$

which implies that encoding and decoding map can be represented as matrix–vector product by $\boldsymbol{V}^{-1}$ and $\boldsymbol{V}$, respectively.

CKKS bootstrapping. When a ciphertext reaches a minimal modulus, bootstrapping is required to restore its modulus. The contemporary CKKS bootstrapping algorithm is made of the following steps:

- ModRaise: Lifts the ciphertext modulus from bottom modulus q_0 to a larger modulus Q, transforming a ciphertext encrypting $m \in \mathcal{R}$ into one encrypting $m + q_0 I$ for some small integer polynomial $I \in \mathcal{R}$.
- CoeffToSlot (CtS): Applies a homomorphic iDFT to convert the coefficient representation into slot representation, resulting in the ciphertext encrypting $\mathsf{Ecd}(\boldsymbol{m} + q_0 \boldsymbol{I})$, where $\boldsymbol{m}$ and $\boldsymbol{I} \in \mathbb{C}^{N/2}$ are coefficient vectors of m and I, respectively.

- EvalMod: Homomorphically removes the error term $q_0\boldsymbol{I}$ modulo q_0, and return the encryption of $\mathsf{Ecd}(\boldsymbol{m})$.
- SlotToCoeff (StC): Applies a homomorphic DFT to map the slot representation back to the coefficient representation, returning a ciphertext encrypting m with refreshed modulus q satisfying $q_0 < q < Q$.

Radix-2 decomposition of DFT. The DFT matrix $\boldsymbol{V}$ admits a sparse factorization that underlies efficient homomorphic evaluation [CCS19, HHC19]. For a bit-reversal permutation matrix $\boldsymbol{P} \in \mathbb{Z}^{(N/2)\times(N/2)}$, one can decompose the matrix $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{P}$ by adopting the lemma. For a power of two k, let rev_k denote the bit-reversal permutation of $[k]$ and set $\boldsymbol{S}_k := \left(\omega_{4k}^{5^i \mathsf{rev}_k(j)}\right)_{i,j\in[k]}$, so that $\boldsymbol{S}_1 = [1]$ and $\boldsymbol{S} = \boldsymbol{S}_{N/2}$.

Lemma 1 (Radix-2 decomposition [HHC19]). *For $k = 2^u$ with $u \geq 1$, we have*

$$\boldsymbol{S}_k = \boldsymbol{F}_k^{(2)} \begin{bmatrix} \boldsymbol{S}_{k/2} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_{k/2} \end{bmatrix}, \qquad \boldsymbol{F}_k^{(2)} := \begin{bmatrix} \boldsymbol{I}_{k/2} & \boldsymbol{W}_{k/2} \\ \boldsymbol{I}_{k/2} & -\boldsymbol{W}_{k/2} \end{bmatrix},$$

where $\boldsymbol{W}_{k/2} = \mathit{diag}(\omega_{4k}^{5^j})_{j\in[k/2]}$.

Then,

$$\boldsymbol{S} = \boldsymbol{V}\cdot\boldsymbol{P} = \boldsymbol{B}_{N/4}\boldsymbol{B}_{N/8}\cdots\boldsymbol{B}_1, \qquad \boldsymbol{B}_k := \boldsymbol{I}_{N/4k} \otimes \boldsymbol{F}_{2k}^{(2)}.$$

We group these butterfly matrices $\boldsymbol{B}_k$ into r blocks and evaluate each block as a plaintext-ciphertext matrix-vector product (PCMV) via the diagonal method with baby-step giant-step (BSGS) optimization [HS18].

Diagonal method and BSGS. Given a plaintext matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ and a ciphertext encrypting a vector $\boldsymbol{z} \in \mathbb{C}^n$, the diagonal method homomorphically evaluates $\boldsymbol{A}\boldsymbol{z} = \sum_{t\in[n]} \mathsf{diag}_t(\boldsymbol{A}) \odot \rho^t(\boldsymbol{z})$, where $\mathsf{diag}_t(\boldsymbol{A}) := (A_{i,i+t})_{i\in[n]}$ denotes the t-th cyclic diagonal and $\rho^t(\boldsymbol{z}) := (z_{i+t})_{i\in[n]}$ denotes cyclic rotation by t positions. If $\boldsymbol{A}$ is sparse, the sum can be restricted to the set of nonzero offsets $\Delta(\boldsymbol{A}) := \{t \in \mathbb{Z}_n : \mathsf{diag}_t(\boldsymbol{A}) \neq \boldsymbol{0}\}$, which reduces the number of rotations to $|\Delta(\boldsymbol{A})|$. When $\Delta(\boldsymbol{A}) \subseteq [T]$, the baby-step giant-step optimization reduces the number of rotations further by evaluating

$$\boldsymbol{A}\boldsymbol{z} = \sum_{\ell\in[n_{\mathsf{gs}}]} \rho^{n_{\mathsf{bs}}\ell}\Big(\sum_{j\in[n_{\mathsf{bs}}]} \rho^{-n_{\mathsf{bs}}\ell}\big(\mathsf{diag}_{n_{\mathsf{bs}}\ell+j}(\boldsymbol{A})\big) \odot \rho^j(\boldsymbol{z})\Big),$$

where $n_{\mathsf{bs}}n_{\mathsf{gs}} \geq T$ and the rotated diagonals are precomputed in the clear. This requires $(n_{\mathsf{bs}} - 1) + (n_{\mathsf{gs}} - 1)$ rotations. The optimizaiton also applies when $\Delta(\boldsymbol{A})$ is contained in an arithmetic progression $\{t_0 + sk : k \in [T]\}$ with *stride* s: one replaces ρ with ρ^s and absorbs ρ^{t_0} into the giant-step rotations.

Integer lift polynomial and EvalMod range. Let $(b, a) \in \mathcal{R}^2$ be a ciphertext at the base modulus q_0 with centered representatives. Then

$$b + as = m + e + q_0 I \quad \text{in } \mathcal{R}, \tag{2}$$

for some integer polynomial $I = \sum_{j\in[N]} I_j X^j$. Under the condition $\|m + e\|_\infty < q_0/2$, this polynomial is given coefficient-wise by rounding $S = (b + as)/q_0$, i.e. $I = \lfloor S \rceil$. EvalMod removes $q_0 I$ by evaluating a polynomial that approximates the mod-q_0 reduction over an interval $[-K, K]$ assumed to contain $\|I\|_\infty$. More precisely, EvalMod evaluates a function $f(t) = \frac{1}{2\pi} cos\big(2\pi(t - \frac{1}{4})\big)$ by polynomial approximation over an interval $[-K, K]$. One can instead evaluate $\frac{1}{2\pi}\cos\big(\frac{2\pi}{2^r}(t - \frac{1}{4})\big)$ and then apply r times double angle formula $\cos(\theta) = 2\cos(\theta/2)^2 - 1$. Thus, the range parameter K determines the cost of EvalMod: the approximation degree, and hence the depth.

For a ternary secret of Hamming weight h, we model the coefficients of a/q_0 and b/q_0 uniformly sampled from $(-1/2, 1/2)$. Then, each coefficient of I is a sum of at most $h+1$ such variables [BMTPH21], yielding a centered Irwin–Hall variable with variance $\sigma_I^2 = \frac{h+1}{12}$. The bootstrapping failure probability is therefore modeled as

$$f(K,h,N) = 1 - \left[\frac{2}{(h+1)!}\sum_{\ell=0}^{h+1}(-1)^{\ell}\binom{h+1}{\ell}\left(K + \frac{h+1}{2} - \ell\right)_+^{h+1} - 1\right]^N, \qquad (3)$$

where $(x)_+ := \max\{x, 0\}$. Sparse-secret encapsulation [BTPH22] reduces the required range K for a given failure probability by switching to a secret of low Hamming weight for the integer lift.

2.3 FHE over Multivariate Cyclotomic Ring

MatHEAAN. Jang et al. [JLK+22] introduced MatHEAAN, a multivariate variant of CKKS for computation over multi-dimensional datasets such as matrices and tensors. For pairwise coprime integers $M_0, \ldots, M_s$, with $M_0 = 2^k$, the cyclotomic ring $\mathbb{Z}[X]/(\Phi_M(X))$, where $M = \prod_{i=0}^{s} M_i$, admits a multivariate representation

$$\mathbb{Z}[X_0, \ldots, X_s]/\big(X_0^{2^{k-1}} + 1, \Phi_{M_1}(X_1), \ldots, \Phi_{M_s}(X_s)\big).$$

This representation identifies the CKKS message space with a multi-dimensional tensor layout, so that automorphisms act dimension-wise rather than as unstructured permutations of a flat SIMD vector. In particular, the case $s = 1$ is naturally suited to matrix-shaped data and matrix-oriented operations.

Gentry–Lee framework. Recently, Gentry–Lee [GL26] proposed reduction-based algorithms for batch matrix multiplication (MM) on encrypted data, replacing rotation-heavy slot manipulations with linear algebra over plaintexts. Gentry–Lee built on the multivariate ring framework of MatHEAAN [JLK+22], specializing to the bivariate setting with $m_1 = 4n_1$ for a power-of-two n_1 and an odd m_2. They use the base ring

$$\mathcal{R}^{(\mathrm{i})} = \mathbb{Z}[\mathrm{i}][U, W]/(U^{n_1} - \mathrm{i}, \Phi_{m_2}(W))$$

and its trivariate extension

$$\mathcal{S}^{(\mathrm{i})} = \mathbb{Z}[\mathrm{i}][U, V, W]/(U^{n_1} - \mathrm{i}, V^{n_1} - \mathrm{i}, \Phi_{m_2}(W)),$$

equipped with the encoding map

$$\begin{aligned}\Psi^{(\mathrm{i})} : \mathbb{C}[U, V, W]/(U^{n_1} - \mathrm{i}, V^{n_1} - \mathrm{i}, \Phi_{m_2}(W)) &\to \mathcal{M}_{n_1}(\mathbb{C})^{\phi(m_2)},\\ m(U, V, W) &\mapsto \Big\{\big(m(\zeta_i, \zeta_j, \eta_k)\big)_{i,j\in[n_1]}\Big\}_{k\in[\phi(m_2)]},\end{aligned}$$

which packs $\phi(m_2)$ batches of $n_1 \times n_1$ matrices. Here ζ_i and η_k are $4n_1$-th and m_2-th primitive roots of unity, respectively. We also define the k-th slice $\Psi_{3,k}^{(\mathrm{i})} : \mathcal{S}^{(\mathrm{i})} \to \mathcal{M}_{n_1}(\mathbb{C})$ along the W-axis, given by

$$\Psi_{3,k}^{(\mathrm{i})}(m) = \big(m(\zeta_i, \zeta_j, \eta_k)\big)_{i,j\in[n_1]}, \quad m \in \mathcal{S}^{(\mathrm{i})}.$$

Given a secret key $s = s(U, W) \in \mathcal{R}^{(\mathrm{i})}$, Gentry–Lee encryption $(b, a) \in (\mathcal{S}_q^{(\mathrm{i})})^2$ of a plaintext $m \in \mathcal{S}^{(\mathrm{i})}$ satisfies

$$b + a \cdot s = m + e \pmod{q}$$

for an error $e \in \mathcal{S}^{(\mathrm{i})}$. Writing $m = \sum_j m_j V^j$ and similarly for a, b, and e, this relation decomposes into n_1 ciphertexts over $\mathcal{R}^{(\mathrm{i})}$:

$$b_j + a_j \cdot s = m_j + e_j \pmod{q}, \quad j \in [n_1].$$

For $a, b \in \mathcal{S}^{(\mathrm{i})}$, define

$$a \circledast b := \mathsf{Tr}_Z\big(a(U, Z, W) \cdot \bar{b}(V^{-1}, Z^{-1}, W^{-1})\big), \qquad \mathcal{T}^{(\mathrm{i})} := \mathbb{Z}[\mathrm{i}][W]/(\Phi_{m_2}(W)),$$

where Tr_Z denotes the degree-zero component with respect to the auxiliary variable Z. The following theorems describe the encrypted batch matrix multiplication algorithms.

Theorem 1 (Reduction to PPMM [GL26, Corollary 3.9]). *For $a = \sum_{i,k} a_{i,k} U^i W^k$, $b' = \bar{b}(U^{-1}, V, W^{-1}) = \sum_{i,k} b'_{i,k} U^i W^k \in \mathcal{S}^{(\mathrm{i})}$, write $A = \big(a_{i,k}\big)$, $B = \big(b'_{i,k}\big) \in \big(\mathcal{T}^{(\mathrm{i})}\big)^{n_1 \times n_1}$. Then for $c = a \circledast b = \sum_{i,k} c_{ik} U^i V^k \in \mathcal{S}^{(\mathrm{i})}$, the matrix $\boldsymbol{C} = \big(c_{ik}\big) \in \big(\mathcal{T}^{(\mathrm{i})}\big)^{n_1 \times n_1}$ satisfies*

$$\boldsymbol{C} = \boldsymbol{A} \cdot \boldsymbol{B}^T \in \big(\mathcal{T}^{(\mathrm{i})}\big)^{n_1 \times n_1}.$$

Theorem 2 (Batch MM [GL26, Theorem 3.8]). *For all $a, b \in \mathcal{S}^{(i)}$, we have*

$$\Psi^{(\mathrm{i})}_{3,k}(a \circledast b) = \frac{1}{n_1} \Psi^{(\mathrm{i})}_{3,k}(a) \cdot \big(\Psi^{(\mathrm{i})}_{3,k}(b)\big)^*, \quad \forall k \in [\phi(m_2)].$$

3 Extended CKKS over Mixed-Radix Cyclotomic Ring

In this section, we extend CKKS to the mixed-radix cyclotomic ring

$$\mathcal{R} = \mathbb{Z}[X]/(\Phi_{3N}(X)) = \mathbb{Z}[X]/(\Phi_6(X^{N/2})) = \mathbb{Z}[X]/(X^N - X^{N/2} + 1).$$

Throughout, the ring degree is of the form $N = 2^\alpha 3^\beta$ with $n_1 = 2^{\alpha-2}$ and $n_2 = 2 \cdot 3^\beta$. Analogously to standard CKKS, we specify the encoding map and characterize the automorphism group of $\mathcal{R}$ in Section 3.1, analyze the error distribution and the bootstrapping range over $\mathcal{R}$ in Section 3.2, and derive the homomorphic mixed-radix FFT for bootstrapping in Section 3.3.

3.1 Encoding Structure and Automorphism Group

We construct a canonical embedding of $\mathbb{R}[X]/(\Phi_{3N}(X))$ into $\mathbb{C}^{n_1 \times n_2}$, which gives a two-dimensional slot structure with independent row and column automorphisms for $\mathcal{R}$. In standard CKKS with cyclotomic order 2^α, one has $\mathbb{Z}^\times_{2^\alpha}/\langle -1\rangle \cong \langle 5\rangle$, and the automorphism $X \mapsto X^5$ generates the cyclic slot rotation.

We similarly seek generators whose powers form a complete set of coset representatives for $\mathbb{Z}^\times_{3N}/\langle -1\rangle$, which will define the rotation structure of our scheme. Note that

$$\mathbb{Z}^\times_{3N} \cong \mathbb{Z}^\times_{2^\alpha} \times \mathbb{Z}^\times_{3^{\beta+1}},$$

and the multiplicative group $\mathbb{Z}^\times_{3^{\beta+1}}$ is cyclic, generated by 2. The following lemma identifies two such generators $g_1, g_2 \in \mathbb{Z}^\times_{3N}$.

Lemma 2. *For every integer $\alpha \geq 2$, $\beta \geq 1$, we have*

$$5^{2^{\alpha-2}} \equiv 1 + 2^\alpha \pmod{2^{\alpha+1}},$$
$$2^{2 \cdot 3^{\beta-1}} \equiv 1 + 3^\beta \pmod{3^{\beta+1}}.$$

In particular, $5 \in \mathbb{Z}^\times_{2^\alpha}$ has order $2^{\alpha-2}$ and $\mathbb{Z}^\times_{2^\alpha} = \langle 5, -1\rangle$. Similarly, $2 \in \mathbb{Z}^\times_{3^{\beta+1}}$ has order $2 \cdot 3^\beta$, and $\mathbb{Z}^\times_{3^{\beta+1}} = \langle 2\rangle$.

Lemma 3. *Set $m_1 := 2^\alpha$ and $m_2 := 3^{\beta+1}$. Let $\mathsf{CRT}_{(m_1,m_2)}(s,t)$ denote the unique class $x \in \mathbb{Z}_{3N}$ satisfying $x \equiv s \pmod{m_1}$ and $x \equiv t \pmod{m_2}$. Then*

$$\mathbb{Z}_{3N}^\times/\langle -1\rangle = \langle g_1\rangle \times \langle g_2\rangle,$$

where $g_1 = \mathsf{CRT}_{(m_1,m_2)}(5,1)$ and $g_2 = \mathsf{CRT}_{(m_1,m_2)}(-1,4)$ have orders n_1 and n_2 in the multiplicative group $\mathbb{Z}_{3N}^\times$, respectively.

Proof. The proofs are given in Appendix A.1. □

Let ω_{3N} be a primitive $3N$-th root of unity. We define the canonical embedding

$$\psi : \mathcal{R} \to \mathbb{C}^{n_1 \times n_2}, \text{ given by } a(X) \mapsto \left(a\left(\omega_{3N}^{g_1^i g_2^j}\right)\right)_{i\in[n_1], j\in[n_2]}.$$

For $a \in \mathcal{R}$, we write $\boldsymbol{Z} = \psi(a)$ for the resulting $n_1 \times n_2$ slot matrix. The direct-product decomposition in Lemma 3 provides the two-dimensional indexing. In particular,

$$\left\{g_1^i g_2^j : i \in [n_1],\, j \in [n_2]\right\}$$

is a complete set of representatives for $\mathbb{Z}_{3N}^\times/\langle -1\rangle$, where $\langle g_1\rangle$ and $\langle g_2\rangle$ independently index the two dimensions of the slot matrix.

Automorphisms as matrix rotations. For $g \in \mathbb{Z}_{3N}^\times$, let $\tau_g : \mathcal{R} \to \mathcal{R}$ denote the automorphism defined by $\tau_g(a)(X) = a(X^g)$. Under the canonical embedding, the automorphisms corresponding to -1, g_1, and g_2 act as

$$\left(\psi(\tau_{-1}(a))\right)_{i,j} = (\overline{\boldsymbol{Z}})_{i,j}, \quad \left(\psi(\tau_{g_1}(a))\right)_{i,j} = (\boldsymbol{Z})_{i+1,j}, \quad \left(\psi(\tau_{g_2}(a))\right)_{i,j} = (\boldsymbol{Z})_{i,j+1},$$

where the row and column indices are taken modulo n_1 and n_2, respectively. Here, τ_{-1} acts as complex conjugation, whereas τ_{g_1} and τ_{g_2} induce independent cyclic row and column rotations, respectively. Thus, the mixed cyclotomic order $3N = 2^\alpha 3^{\beta+1}$ yields a natural two-dimensional slot structure.

Comparison with MatHEAAN. Although MatHEAAN exposes dimension-wise automorphisms through a multivariate representation, we use the equivalent univariate ring $\mathcal{R} = \mathbb{Z}[X]/(X^N - X^{N/2} + 1)$, which is better suited to optimized mixed-radix NTT arithmetic [LS19, KP23]. Its row and column automorphisms are given by $X \mapsto X^{g_1}$ and $X \mapsto X^{g_2}$, where g_1 and g_2 are chosen to align the slot ordering with the optimized radix-3 FFT decomposition detailed in Section 3.3.

Example 1. We illustrate our constructions with toy example $(\alpha,\beta) = (3,1)$, i.e., $(m,N,n) = (72,24,12)$, $(m_1,n_1) = (8,2)$, and $(m_2,n_2) = (9,6)$. Lemma 3 gives the following generators

$$g_1 = \mathsf{CRT}_{(8,9)}(5,1) = 37, \qquad g_2 = \mathsf{CRT}_{(8,9)}(-1,4) = 31$$

with respective orders $n_1 = 2$ and $n_2 = 6$. The slot (i,j) of $\psi(a)$ is $a(\omega_{72}^{e_{i,j}})$ with $e_{i,j} = g_1^i g_2^j \bmod 72$:

$$(e_{i,j})_{i\in[2],\, j\in[6]} = \begin{bmatrix} 1 & 31 & 25 & 55 & 49 & 7 \\ 37 & 67 & 61 & 19 & 13 & 43 \end{bmatrix}.$$

3.2 Error Distribution and Bootstrapping Range

As discussed in Section 2, choosing the EvalMod range K requires estimating the distribution of the integer-lift polynomial

$$I = \lfloor S \rceil \in \mathcal{R}, \qquad S = (b + a \cdot s)/q_0 \in \mathbb{R}[X]/(X^N - X^{N/2} + 1)$$

arising in (2). In the power-of-two setting, multiplication by a monomial only permutes coefficients up to sign. For a ternary secret of Hamming weight h, each coefficient of I is therefore modeled by rounding a sum of $h+1$ independent uniform variables on $(-1/2, 1/2)$, which underlies the Irwin–Hall estimate in (3).

In this subsection, we extend the analysis to the mixed-radix ring $\mathcal{R} = \mathbb{Z}[X]/(X^N - X^{N/2} + 1)$. The relation $X^N = X^{N/2} - 1$ combines coefficients during polynomial multiplication, changing both their variances and their dependence. We then derive a bound for selecting K to meet a target bootstrapping failure probability and show that the increased coefficient variance leaves the normalized slot variance unchanged.

Secret-key sampling and security assumption. We adopt HElib's recryption-key sampler [HS21], which fixes the Hamming weight h before cyclotomic reduction, whereas conventional CKKS samples a weight-h ternary secret directly in the power basis. For the sparse weights considered here, reduction increases the expected number of nonzero coefficients and may produce non-ternary coefficients. The power-of-two failure estimate thus cannot be applied directly with the same h.

We assume that the induced secret distribution is at least as hard against lattice attacks as a weight-h ternary secret at the same dimension, modulus, and error distribution. Our security estimates are conditional on this assumption; a dedicated analysis accounting for the induced coefficient dependencies is left for future work.

3.2.1 Coefficient and Slot Domain Error Variances

We first examine how the secret-key sampling described above affects error variance in the coefficient and slot domains. Our analysis of the secret-product term $d = cs$ shows that the coefficient variance increases by a factor of $4/3$ relative to the power-of-two ring, while the normalized slot variance remains unchanged.

Coefficient-domain variance. We first consider the secret-product term $d = cs$, which plays an important role in FHE noise analysis. We assume that c has independent, zero-mean coefficients with variance σ^2. The following lemma quantifies the coefficient variances of d.

Lemma 4 (Coefficient-domain variance)**.** *Let $c(X) = \sum_{k\in[N]} c_k X^k$ have independent coefficients of mean zero and variance σ^2. Independently of c, sample $\widetilde{s}(X) = \sum_{j\in[3N]} s_j X^j$ by choosing its support uniformly among the h-element subsets of $[3N]$ and assigning independent uniform signs in $\{\pm 1\}$, independently of the support. Set $s = \widetilde{s} \bmod \Phi_{3N}(X)$ and $d = cs \bmod \Phi_{3N}(X) = \sum_{i\in[N]} d_i X^i$. Then*

$$\mathrm{Var}(d_i) = \tfrac{4}{3} h\sigma^2 \qquad (i \in [N]),$$

where the variance is taken over the sampling of both c and s.

Proof. The proof is in Appendix A.2. □

Slot-domain variance. Averaged over secret sampling, the coefficient variance is $\frac{4}{3}h\sigma^2$, compared with $h\sigma^2$ in the power-of-two ring. Nevertheless, the following corollary shows that the normalized slot variance is unchanged, consistent with Table 4.

Corollary 1 (Slot-domain variance)**.** *Let c, s, and d be as in Lemma 4. For every primitive $3N$-th root of unity ξ,*

$$\frac{1}{N}\operatorname{Var}(d(\xi)) = h\sigma^2, \tag{4}$$

where the variance is taken over the sampling of both c and s.

Proof. The proof is provided in Appendix A.2. □

3.2.2 EvalMod Range and Bootstrapping Failure Probability

We now apply this analysis to the integer-lift polynomial $I = \lfloor S \rceil$. Here, we model the coefficients of b/q_0 and a/q_0 as independent uniforms on $(-1/2, 1/2)$. Applying Lemma 4 with $c = a/q_0$ and $\sigma^2 = 1/12$, and adding the independent contribution of b/q_0, gives

$$\sigma_S^2 := \operatorname{Var}(S_j) = \frac{1}{12} + \frac{4h}{3}\cdot\frac{1}{12} = \frac{4h+3}{36}, \qquad j \in [N]. \tag{5}$$

Following the approach in HElib [HS21], one may regard each S_j as a normal variable with this variance and apply a union bound over the N coefficients, which gives the estimate

$$\Pr\big[\|S\|_\infty > K\big] \lesssim N \cdot \operatorname{erfc}\big(K/\sqrt{2\sigma_S^2}\big),$$

where erfc is the complementary error function. This estimate is quite pessimistic, since the Hamming weight h of s is usually small with sparse-secret encapsulation [BTPH22], so S_j is a sum of few bounded uniforms whose tail is much lighter than that of a normal variable with the same variance. We derive a tighter bound for the distribution of S_j from the following theorem, which is proved in Appendix A.2.

Theorem 3 (Bootstrapping failure probability)**.** *Let $s = \sum_{\ell\in[3N]} s_\ell X^\ell \pmod{X^N - X^{N/2}+1} \in \mathcal{R}$ be sampled as in Lemma 4 of Hamming weight h before reduction, and let $S = (b+as)/q_0$ with the coefficients of b/q_0 and a/q_0 independent uniforms on $(-1/2, 1/2)$. Write $(as)_i = \sum_{k\in[N]} (\boldsymbol{M}_s)_{i,k}\, a_k$, where $\boldsymbol{M}_s \in \mathbb{Z}^{N\times N}$ is the multiplication-by-s matrix in the power basis, and define*

$$L_i(s) = \#\{k \in [N] : (\boldsymbol{M}_s)_{i,k} \neq 0\}, \qquad c_i(s) = \sum_{k\in[N]} \big(|(\boldsymbol{M}_s)_{i,k}| - 1\big)_+,$$

$$\pi_{L,c} = \Pr_s\big[L_i(s) = L,\ c_i(s) = c\big],$$

$$\bar{F}_M(t) = \Pr\Big[\Big|\sum_{m=1}^{M} U_m\Big| > t\Big] \text{ where } U_m \overset{iid}{\sim} \mathcal{U}(-\tfrac{1}{2}, \tfrac{1}{2}),$$

where $\pi_{L,c}$ does not depend on i. Then for every integer $K \geq 1$,

$$\Pr\big[\|S\|_\infty > K\big] \ \leq\ N \sum_{L,c} \pi_{L,c}\, \bar{F}_{1+L}\Big(K - \frac{c}{2}\Big). \tag{6}$$

Sparse-secret encapsulation is widely used in CKKS bootstrapping, typically with a Hamming weight $\tilde{h} = 32$ ternary secret and $K = 16$ [BTPH22]. Since bootstrapping failures can enable attacks against IND-CPA^D security [CCP+24], we target a range-failure probability below 2^{-128}. The choice $\tilde{h} = 32$, $K = 16$ meets this target for the power-of-two parameters considered here. However, in our ring, retaining the pre-reduction weight $\tilde{h} = 32$ and $K = 16$ yields a bound in Theorem 3 well above this target. We therefore choose $K = 25$, for which the theorem certifies a range-failure probability below 2^{-128} for our parameters under the independent-uniform coefficient model, averaged over secret sampling. Table 6 (in Section 5) compares the resulting bounds and bootstrapping performance.

3.3 Bootstrapping over the Mixed Cyclotomic Ring

The CtS and StC stages of CKKS bootstrapping evaluate the inverse DFT and DFT associated with the canonical embedding, respectively. For the matrix encoding of Section 3.1, we express these transforms under row-major flattening and derive a mixed-radix FFT decomposition aligned with the two slot dimensions. The resulting radix-2 and radix-3 butterfly layers are grouped and evaluated through the native row and column automorphisms of $\mathcal{R}$, providing a trade-off between multiplicative depth and key-switching cost.

3.3.1 DFT as Matrix–Vector Product

For $\mu(X) = \sum_{j=0}^{2n-1} \mu_j X^j \in \mathbb{R}[X]/(X^{2n} - X^n + 1)$ (recall $N = 2n$, so that $X^{2n} - X^n + 1 = \Phi_{3N}(X)$ and $\omega_{6n} = \omega_{3N}$), we denote its slot matrix $\boldsymbol{Z} = \psi(\mu) \in \mathbb{C}^{n_1 \times n_2}$ and flatten it row-major into $\boldsymbol{z} = (z_i)_{i\in[n]}^T \in \mathbb{C}^n$ by setting $z_{i_1 n_2 + i_2} = Z_{i_1,i_2}$. By the definition of the canonical embedding,

$$z_{i_1 n_2 + i_2} = \mu\big(\omega_{6n}^{g_1^{i_1} g_2^{i_2}}\big), \qquad i_1 \in [n_1], \quad i_2 \in [n_2].$$

For the coefficient vector $\boldsymbol{\mu} = (\mu_0, \ldots, \mu_{2n-1})^T$, the DFT can be represented as the following matrix–vector product:

$$\boldsymbol{z} = \boldsymbol{F}\,\boldsymbol{\mu}, \qquad \boldsymbol{F} = (F_{i,j}) \in \mathbb{C}^{n\times 2n},$$

where $F_{i_1 n_2 + i_2, j} = \omega_{6n}^{g_1^{i_1} g_2^{i_2} j}$ for $i_1 \in [n_1]$, $i_2 \in [n_2]$, and $j \in [2n]$. Let $\boldsymbol{D} \in \mathbb{C}^{n\times n}$ denote the first n columns of $\boldsymbol{F}$. Since $g_1 \equiv g_2 \equiv 1 \pmod 6$ and $\omega_{6n}^n = \omega_6$, we have

$$\boldsymbol{F} = \big[\, \boldsymbol{D} \mid \omega_6 \boldsymbol{D} \,\big], \qquad \boldsymbol{z} = \boldsymbol{F}\cdot\boldsymbol{\mu} = \boldsymbol{D}\,(\boldsymbol{\mu}_0 + \omega_6 \boldsymbol{\mu}_1), \tag{7}$$

where $\boldsymbol{\mu}_0 = (\mu_0, \ldots, \mu_{n-1})^T$ and $\boldsymbol{\mu}_1 = (\mu_n, \ldots, \mu_{2n-1})^T$ with $\boldsymbol{\mu} = (\boldsymbol{\mu}_0^T, \boldsymbol{\mu}_1^T)^T \in \mathbb{R}^{2n}$.

Now we describe CtS and StC over the mixed-radix cyclotomic ring $\mathcal{R}$. Given a ciphertext whose slots encrypt $\boldsymbol{z} \in \mathbb{C}^n$, CtS first homomorphically evaluates

$$\boldsymbol{u} := \boldsymbol{D}^{-1}\boldsymbol{z} = \boldsymbol{t}_0 + \omega_6 \boldsymbol{t}_1, \qquad \boldsymbol{t}_0, \boldsymbol{t}_1 \in \mathbb{R}^n,$$

where $\boldsymbol{D}^{-1} = n^{-1}\boldsymbol{D}^*$ and n^{-1} is absorbed into the plaintext masks, and then separates the two real components with respect to the basis $\{1, \omega_6\}$, yielding ciphertexts ct_0 and ct_1 whose slots encrypt $\boldsymbol{t}_0$ and $\boldsymbol{t}_1$, respectively. After EvalMod is applied independently to these ciphertexts, StC recombines them and evaluates

$$\boldsymbol{z} = \boldsymbol{D}\cdot(\boldsymbol{t}_0 + \omega_6 \boldsymbol{t}_1).$$

We note that $X^{N/2}$ and $2X^{N/2} - 1$ have the constant slot values ω_6 and $\sqrt{3}\mathrm{i}$, respectively. Using conjugation and precomputation of the constant term, neither StC nor CtS requires an additional multiplicative level to handle the ω_6 component.

3.3.2 Homomorphic Mixed-Radix FFT

The StC and CtS transformations reduce to homomorphically evaluating $\boldsymbol{D}$ and $\boldsymbol{D}^{-1}$. Under the slot ordering induced by the CRT generators, mixed-radix digit reversal yields radix-2, transition, and radix-3 recursions for these matrices. We use the resulting FFT factorizations to evaluate StC and CtS over the same univariate representation used for mixed-radix NTT arithmetic.

Mixed-radix FFT factorization. For the radix sequence of $\alpha - 1$ twos followed by β threes, whose product is $n = 2^{\alpha-1}3^\beta$, define the *mixed-radix digit-reversal permutation* recursively. For $k = 2^u 3^v > 1$, let $d_k = 2$ if $u > 0$ and $d_k = 3$ otherwise, and set

$$\mathsf{rev}_1(0) = 0, \qquad \mathsf{rev}_k(j) = d_k \mathsf{rev}_{k/d_k}\big(j \bmod (k/d_k)\big) + \left\lfloor \frac{j}{k/d_k} \right\rfloor.$$

Let $\boldsymbol{P}$ denote its permutation matrix, with $P_{\mathsf{rev}_n(j),j} = 1$ for $j \in [n]$. Its radix order matches the row-major slot ordering: the first $\alpha - 2$ binary stages follow g_1, whereas the transition stage and the subsequent ternary stages follow g_2. Consequently,

$$\boldsymbol{S} := \boldsymbol{D}\boldsymbol{P}, \qquad S_{i_1 n_2 + i_2, j} = D_{i_1 n_2 + i_2, \mathsf{rev}_n(j)} = \omega_{6n}^{i_1 g_1 + i_2 g_2 \, \mathsf{rev}_n(j)}.$$

For each recursive size $k = 2^u 3^v$, let $\boldsymbol{S}_k$ denote the reordered DFT core induced by the same row-wise slot ordering and rev_k at modulus $6k$, with $\boldsymbol{S}_n = \boldsymbol{S}$ and $\boldsymbol{S}_1 = [1]$. Below, g_1 and g_2 are understood through their natural reductions modulo $6k$. The proofs for the following lemmas are provided in Appendix A.3.

Lemma 5 (Radix-**2** butterfly)**.** *For $k = 2^u 3^v$ with $u > 1$,*

$$\boldsymbol{S}_k = \boldsymbol{F}_k^{(2)} \cdot (\boldsymbol{I}_2 \otimes \boldsymbol{S}_{k/2}) \in \mathcal{M}_k(\mathbb{C}), \qquad \boldsymbol{F}_k^{(2)} := \begin{bmatrix} \boldsymbol{I}_{k/2} & \boldsymbol{W}_{k/2} \\ \boldsymbol{I}_{k/2} & -\boldsymbol{W}_{k/2} \end{bmatrix},$$

where $\boldsymbol{W}_{k/2}$ is diagonal, with entries

$$(\boldsymbol{W}_{k/2})_{i,i} := \omega_{6k}^{i_1 g_1 + i_2 g_2}, \qquad i = (2 \cdot 3^v) i_1 + i_2, \quad i_1 \in [2^{u-2}], \quad i_2 \in [2 \cdot 3^v].$$

Lemma 6 (Radix transition)**.** *For $\ell = 3^v$, the transition from the radix-2 to the radix-3 recursion is*

$$\boldsymbol{S}_{2\ell} = \boldsymbol{F}_{2\ell}^{(\mathrm{tr})} \cdot (\boldsymbol{I}_2 \otimes \boldsymbol{S}_\ell) \in \mathcal{M}_{2\ell}(\mathbb{C}), \qquad \boldsymbol{F}_{2\ell}^{(\mathrm{tr})} := \begin{bmatrix} \boldsymbol{I}_\ell & \boldsymbol{T}_\ell \\ \boldsymbol{I}_\ell & -\boldsymbol{T}_\ell \end{bmatrix}, \quad \boldsymbol{T}_\ell := \mathsf{diag}\big(\omega_{12\ell}^{g_2^i}\big)_{i \in [\ell]}.$$

Lemma 7 (Radix-**3** butterfly)**.** *For $k = 3^v$ with $v > 0$,*

$$\boldsymbol{S}_k = \boldsymbol{F}_k^{(3)} (\boldsymbol{I}_3 \otimes \boldsymbol{S}_{k/3}) \in \mathcal{M}_k(\mathbb{C}), \qquad \boldsymbol{F}_k^{(3)} := \begin{bmatrix} \boldsymbol{I}_{k/3} & \boldsymbol{A}_{k/3} & \boldsymbol{A}_{k/3}^2 \\ \boldsymbol{I}_{k/3} & \omega^2 \boldsymbol{A}_{k/3} & \omega \boldsymbol{A}_{k/3}^2 \\ \boldsymbol{I}_{k/3} & \omega \boldsymbol{A}_{k/3} & \omega^2 \boldsymbol{A}_{k/3}^2 \end{bmatrix},$$

where $\boldsymbol{A}_{k/3} := \mathsf{diag}\big(\omega_{6k}^{g_2^i}\big)_{i \in [k/3]}$.

Let $\widetilde{\boldsymbol{F}}_k^{(d)} := \boldsymbol{I}_{n/k} \otimes \boldsymbol{F}_k^{(d)}$ and $\widetilde{\boldsymbol{F}}_{2\ell}^{(\mathrm{tr})} := \boldsymbol{I}_{n/(2\ell)} \otimes \boldsymbol{F}_{2\ell}^{(\mathrm{tr})}$. Recursively applying the three lemmas gives the following complete factorization.

Proposition 1 (Mixed-radix FFT factorization)**.**

$$\begin{aligned} \boldsymbol{S} &= \widetilde{\boldsymbol{F}}_n^{(2)} \cdots \widetilde{\boldsymbol{F}}_{4 \cdot 3^\beta}^{(2)} \, \widetilde{\boldsymbol{F}}_{2 \cdot 3^\beta}^{(\mathrm{tr})} \, \widetilde{\boldsymbol{F}}_{3^\beta}^{(3)} \cdots \widetilde{\boldsymbol{F}}_3^{(3)}, \\ \boldsymbol{S}^{-1} &= \frac{1}{n} \big(\widetilde{\boldsymbol{F}}_3^{(3)}\big)^* \cdots \big(\widetilde{\boldsymbol{F}}_{3^\beta}^{(3)}\big)^* \big(\widetilde{\boldsymbol{F}}_{2 \cdot 3^\beta}^{(\mathrm{tr})}\big)^* \big(\widetilde{\boldsymbol{F}}_{4 \cdot 3^\beta}^{(2)}\big)^* \cdots \big(\widetilde{\boldsymbol{F}}_n^{(2)}\big)^*. \end{aligned} \tag{8}$$

Here, the radix-3 products are empty for $\beta = 0$. Indeed, for $d \in \{2, 3\}$, $(\widetilde{\boldsymbol{F}}_k^{(d)})^{-1} = d^{-1}(\widetilde{\boldsymbol{F}}_k^{(d)})^*$, and $(\widetilde{\boldsymbol{F}}_{2\ell}^{(\mathrm{tr})})^{-1} = \frac{1}{2}(\widetilde{\boldsymbol{F}}_{2\ell}^{(\mathrm{tr})})^*$, so the inverse factorization has the same sparsity.

Proof. The proof is given in Appendix A.3. □

Example 2. Continuing Example 1, we illustrate the mixed-radix FFT for the same 2×6 slot layout. Let $\xi := \omega_{72}$ and let $(e_{i,j})$ be the exponent matrix given there. The corresponding DFT matrix $\boldsymbol{D} \in \mathbb{C}^{12 \times 12}$ is

$$D_{6i+j,k} = \xi^{e_{i,j} k}, \qquad i \in [2], \quad j \in [6], \quad k \in [12].$$

Mixed-radix digit reversal reorders its columns according to

$$\big(\mathsf{rev}_{12}(k)\big)_{k \in [12]} = (0, 4, 8, 2, 6, 10, 1, 5, 9, 3, 7, 11),$$

yielding $\boldsymbol{S}_{12} = \boldsymbol{DP}$. Proposition 1 specializes to one radix-2 layer, one transition layer, and one radix-3 layer:

$$\begin{aligned}\boldsymbol{S}_{12} &= \widetilde{\boldsymbol{F}}_{12}^{(2)}\, \widetilde{\boldsymbol{F}}_{6}^{(\mathrm{tr})}\, \widetilde{\boldsymbol{F}}_{3}^{(3)} \\ &= \begin{bmatrix} \boldsymbol{I}_6 & \boldsymbol{W}_6 \\ \boldsymbol{I}_6 & -\boldsymbol{W}_6 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{T}_3 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{I}_3 & -\boldsymbol{T}_3 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_3 & \boldsymbol{T}_3 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_3 & -\boldsymbol{T}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_3 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_3 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{S}_3 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{S}_3 \end{bmatrix},\end{aligned}$$

where

$$\boldsymbol{W}_6 = \mathsf{diag}\big(\xi^1, \xi^{31}, \xi^{25}, \xi^{55}, \xi^{49}, \xi^7\big), \quad \boldsymbol{T}_3 = \mathsf{diag}\big(\xi^2, \xi^{62}, \xi^{50}\big), \quad \boldsymbol{S}_3 = \begin{bmatrix} 1 & \xi^4 & \xi^8 \\ 1 & \xi^{52} & \xi^{32} \\ 1 & \xi^{28} & \xi^{56} \end{bmatrix}.$$

Since $\boldsymbol{S} = \boldsymbol{DP}$, CtS may evaluate $\boldsymbol{S}^{-1} = \boldsymbol{P}^{-1}\boldsymbol{D}^{-1}$. Slot-wise $\mathsf{EvalMod}$ preserves this permuted order, and the subsequent StC by $\boldsymbol{S}$ cancels $\boldsymbol{P}^{-1}$; hence the reordering is free.

Native row and column rotations. We split (8) into the two rotation axes:

$$\boldsymbol{S} = \boldsymbol{G}_{\mathrm{row}} \cdot \boldsymbol{G}_{\mathrm{col}}, \qquad \boldsymbol{G}_{\mathrm{row}} := \widetilde{\boldsymbol{F}}_{n}^{(2)} \cdots \widetilde{\boldsymbol{F}}_{4\cdot 3^{\beta}}^{(2)}, \qquad \boldsymbol{G}_{\mathrm{col}} := \widetilde{\boldsymbol{F}}_{2\cdot 3^{\beta}}^{(\mathrm{tr})}\, \widetilde{\boldsymbol{F}}_{3^{\beta}}^{(3)} \cdots \widetilde{\boldsymbol{F}}_{3}^{(3)}.$$

Each lifted radix-2 factor, hence $\boldsymbol{G}_{\mathrm{row}}$, has diagonal $n_2 \times n_2$ blocks in its $n_1 \times n_1$ block form. Equivalently, its nonzero cyclic-diagonal offsets are multiples of n_2, so it mixes rows while preserving columns. We note that $\boldsymbol{G}_{\mathrm{col}}$ has the block-diagonal form

$$\boldsymbol{G}_{\mathrm{col}} = \boldsymbol{I}_{n_1} \otimes \boldsymbol{S}_{n_2} = \mathsf{diag}(\boldsymbol{S}_{n_2}, \ldots, \boldsymbol{S}_{n_2}),$$

and thus acts independently within each row.

Grouping to reduce depth. To trade multiplicative depth for rotation cost, we partition $\boldsymbol{G}_{\mathrm{row}}$ and $\boldsymbol{G}_{\mathrm{col}}$ into r_1 and r_2 nonempty groups of consecutive butterfly factors, respectively:

$$\boldsymbol{G}_{\mathrm{row}} = \boldsymbol{R}_{r_1} \cdots \boldsymbol{R}_1, \qquad \boldsymbol{G}_{\mathrm{col}} = \boldsymbol{C}_{r_2} \cdots \boldsymbol{C}_1 \qquad (1 \le r_1 \le \alpha - 2,\ 1 \le r_2 \le \beta + 1).$$

Let $\Delta_{\mathrm{row}}(\boldsymbol{R}) \subseteq \mathbb{Z}_{n_1}$ and $\Delta_{\mathrm{col}}(\boldsymbol{C}) \subseteq \mathbb{Z}_{n_2}$ denote the set of shifts a and b inducing nonzero coefficients from $(i+a, j)$ and $(i, j+b)$, respectively, to (i, j) for some i, j.

Lemma 8 (Grouped row and column diagonals)**.** *Let $\boldsymbol{R}$ and $\boldsymbol{C}$ be nonempty consecutive subproducts of $\boldsymbol{G}_{\mathrm{row}}$ and $\boldsymbol{G}_{\mathrm{col}}$ containing $k_1 \ge 1$ radix-2 and $k_2 \ge 0$ radix-3 factors, respectively, with $\boldsymbol{C}$ possibly including the transition factor. Their supports lie in cyclic arithmetic progressions of distinct offsets, with lengths T_{row} and T_{col} satisfying*

$$|\Delta_{\mathrm{row}}(\boldsymbol{R})| \le T_{\mathrm{row}} \le 2^{k_1+1} - 1, \qquad |\Delta_{\mathrm{col}}(\boldsymbol{C})| \le T_{\mathrm{col}} \le 2 \cdot 3^{k_2}, \quad 0 \le k_2 \le \beta.$$

The same statement holds for the corresponding inverse products.

Proof. The proof is given in Appendix A.3. □

Homomorphic Evaluation and Cost. Under row-major flattening, i.e., mapping slot (i, j) to index $in_2 + j$, define the plaintext mask matrices $\boldsymbol{\Gamma}_{u,a}, \boldsymbol{\Lambda}_{v,b} \in \mathbb{C}^{n_1 \times n_2}$ by

$$(\boldsymbol{\Gamma}_{u,a})_{i,j} := (\boldsymbol{R}_u)_{in_2+j,\ ((i+a) \bmod n_1)n_2+j}, \quad (\boldsymbol{\Lambda}_{v,b})_{i,j} := (\boldsymbol{C}_v)_{in_2+j,\ in_2+((j+b) \bmod n_2)},$$

for $i \in [n_1]$, $j \in [n_2]$. Since $\tau_{g_1^a}$ and $\tau_{g_2^b}$ induce row and column rotations, respectively, the groups admit the following diagonal representations on input $\psi(\nu)$:

$$\psi(\nu) \xmapsto{\boldsymbol{R}_u} \sum_{a \in \Delta_{\mathrm{row}}(\boldsymbol{R}_u)} \boldsymbol{\Gamma}_{u,a} \odot \psi\big(\tau_{g_1^a}(\nu)\big), \quad \psi(\nu) \xmapsto{\boldsymbol{C}_v} \sum_{b \in \Delta_{\mathrm{col}}(\boldsymbol{C}_v)} \boldsymbol{\Lambda}_{v,b} \odot \psi\big(\tau_{g_2^b}(\nu)\big).$$

Each group consumes one multiplicative level. To reduce the rotation cost using BSGS, for each group choose a cyclic arithmetic progression $\{a_0+sk : k \in [T]\}$ of T distinct offsets with stride s containing its support. Masks outside the support are zero. Set $n_{\mathsf{gs}} = \lceil T/n_{\mathsf{bs}} \rceil$ and $h_\ell = a_0 + sn_{\mathsf{bs}}\ell$. Using τ_g also for its induced action on slot matrices, BSGS gives the following expressions, with the rotated plaintext masks precomputed in the clear:

$$\sum_{\ell\in[n_{\mathsf{gs}}]} \tau_{g_1^{h_\ell}}\Big(\sum_{\substack{j\in[n_{\mathsf{bs}}]\\ n_{\mathsf{bs}}\ell+j<T}} \tau_{g_1^{-h_\ell}}(\mathbf{\Gamma}_{u,h_\ell+sj}) \odot \psi(\tau_{g_1^{sj}}(\nu))\Big),$$

$$\sum_{\ell\in[n_{\mathsf{gs}}]} \tau_{g_2^{h_\ell}}\Big(\sum_{\substack{j\in[n_{\mathsf{bs}}]\\ n_{\mathsf{bs}}\ell+j<T}} \tau_{g_2^{-h_\ell}}(\mathbf{\Lambda}_{v,h_\ell+sj}) \odot \psi(\tau_{g_2^{sj}}(\nu))\Big).$$

The balanced choice $n_{\mathsf{bs}} \approx n_{\mathsf{gs}} \approx \sqrt{T}$ minimizes the BSGS bound of $n_{\mathsf{bs}} - 1 + n_{\mathsf{gs}}$ rotations. When the butterfly factors are evenly grouped along each axis, Lemma 8 gives $T = O(n_1^{1/r_1})$ and $T = O(n_2^{1/r_2})$ for row and column groups, respectively. Thus the FFT core of each of StC and CtS requires $O(r_1 n_1^{1/(2r_1)} + r_2 n_2^{1/(2r_2)})$ rotations and key switches at multiplicative depth $r_1 + r_2$.

4 Faster Bootstrapping for Gentry–Lee Framework

In this section, we describe our novel bootstrapping algorithm for Gentry-Lee [GL26]. The homomorphic DFT evaluation of DFT decomposes into encrypted batch MMs with the two smaller DFT matrices $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, so that StC and CtS are performed via reduction-based MM algorithms, which can be accelerated by high-performance libraries such as BLAS and FLINT. In what follows, we introduce auxiliary bivariate and trivariate ring representations and show how they reduce the homomorphic DFT to batch MMs. We summarize the notation used in this section in Table 1.

4.1 Auxiliary Ring Structure and Extension Ring

We now introduce auxiliary analogues of $\mathcal{R}^{(\mathrm{i})}$ and $\mathcal{S}^{(\mathrm{i})}$, which will be used in our bootstrapping construction for Gentry–Lee. We begin by observing that the proposed ring $\mathcal{R} = \mathbb{Z}[X]/(\Phi_{3N}(X))$ admits an equivalent bivariate presentation

$$\mathcal{R}' = \mathbb{Z}[U,W]/(\Phi_{2^\alpha}(U), \Phi_{3^{\beta+1}}(W)) = \mathbb{Z}[U,W]/(U^{2n_1}+1,\ W^{n_2}+W^{n_2/2}+1) \cong \mathcal{R},$$

where the isomorphism $\tau : \mathcal{R}' \to \mathcal{R}$ is induced by $\tau(U) = X^{3^{\beta+1}}$ and $\tau(W) = X^{2^\alpha}$. While $\mathcal{R}^{(\mathrm{i})}$ provides a Gaussian-integer presentation isomorphic to $\mathcal{R}'$, we introduce its ω-side companion

$$\mathcal{R}^{(\omega)} := \mathbb{Z}[\omega][W,U]/(W^{n_2/2} - \omega,\ U^{2n_1}+1),$$

where $\omega = \exp(2\pi\mathrm{i}/3)$ plays the role of i in $\mathcal{R}^{(\mathrm{i})}$. To define the corresponding canonical embedding, we extend the radix-2 and radix-3 evaluation points by

$$\zeta_i = \begin{cases} \zeta^{5^i} & (0 \le i < n_1), \\ \zeta^{-5^{i-n_1}} & (n_1 \le i < 2n_1), \end{cases} \qquad \eta_j = \begin{cases} \eta^{4^j} & (0 \le j < n_2/2), \\ \eta^{-4^{j-n_2/2}} & (n_2/2 \le j < n_2), \end{cases}$$

yielding the embedding maps

$$\psi^{(\mathrm{i})} : \mathbb{C}[U,W]/(U^{n_1} - \mathrm{i},\ W^{n_2}+W^{n_2/2}+1) \to \mathbb{C}^{n_1\times n_2},$$
$$\alpha'(U,W) \mapsto \big(\alpha'(\zeta_i,\eta_j)\big)_{i\in[n_1],\, j\in[n_2]},$$
$$\psi^{(\omega)} : \mathbb{C}[W,U]/(W^{n_2/2} - \omega,\ U^{2n_1}+1) \to \mathbb{C}^{n_2/2\times 2n_1},$$
$$\beta'(W,U) \mapsto \big(\beta'(\eta_i,\zeta_j)\big)_{i\in[n_2/2],\, j\in[2n_1]}.$$

Note that both $\mathcal{R}^{(i)}$ and $\mathcal{R}^{(\omega)}$ are naturally identified with the same base ring $\mathcal{R}'$, and hence with $\mathcal{R}$. From the definition, the following theorem describes the relationship between $\psi^{(\mathrm{i})}$ and $\psi^{(\omega)}$.

Theorem 4. *Let $m = m(U, W) \in \mathcal{R}'$, and suppose that*

$$\psi^{(i)}(m) = \left[\boldsymbol{E}_1 \mid \boldsymbol{E}_2\right] \in \mathbb{C}^{n_1 \times n_2},$$

where $\boldsymbol{E}_1, \boldsymbol{E}_2 \in \mathbb{C}^{n_1 \times (n_2/2)}$. If we define $\widetilde{m}(W, U) = m(U, W) \in \mathcal{R}'$, then

$$\psi^{(\omega)}(\widetilde{m}) = \left[\boldsymbol{E}_1^T \mid \overline{\boldsymbol{E}}_2^T\right] \in \mathbb{C}^{\frac{n_2}{2} \times 2n_1}.$$

Proof. We refer to the Appendix B.1. □

Following Theorem 4, we define the transfer map

$$\mathcal{T}_{i\to\omega} : \mathbb{C}^{n_1 \times n_2} \to \mathbb{C}^{\frac{n_2}{2} \times 2n_1}, \qquad \mathcal{T}_{i\to\omega}(\psi^{(\mathrm{i})}(m)) := \psi^{(\omega)}(\widetilde{m}),$$

for $m \in \mathcal{R}'$, where $\widetilde{m}(W, U) = m(U, W)$. Equivalently, if $\boldsymbol{M} = [\boldsymbol{E}_1 \mid \boldsymbol{E}_2]$, then $\mathcal{T}_{i\to\omega}(\boldsymbol{M}) = [\boldsymbol{E}_1^T \mid \overline{\boldsymbol{E}}_2^T]$. For bootstrapping construction, we further extend $\mathcal{R}^{(\omega)}$ to its trivariate counterpart, in parallel with $\mathcal{S}^{(\mathrm{i})}$:

$$\mathcal{S}^{(\omega)} := \mathbb{Z}[\omega][W, Y, U]/(W^{n_2/2} - \omega,\ Y^{n_2/2} - \omega,\ U^{2n_1} + 1),$$

equipped with the encoding map

$$\Psi^{(\omega)} : \mathbb{C}[W, Y, U]/(W^{n_2/2} - \omega,\ Y^{n_2/2} - \omega,\ U^{2n_1} + 1) \to \mathcal{M}_{\frac{n_2}{2}}(\mathbb{C})^{2n_1},$$
$$m(W, Y, U) \mapsto \big(m(\eta_i, \eta_j, \zeta_k)\big)_{i,j \in [n_2/2],\, k \in [2n_1]}.$$

Table 1: Ring definitions and slot embeddings used in our construction, together with the DFT matrices $\boldsymbol{D}_1$, $\boldsymbol{D}_2$, and $\boldsymbol{D}_2'$ defined in Section 4.2.

	Definition	Description
$\mathcal{R}^{(\mathrm{i})}$	embedding $\mathbb{Z}[\mathrm{i}][U, W]/(U^{n_1} - \mathrm{i},\ W^{n_2} + W^{n_2/2} + 1)$	original bivariate ring
$\mathcal{R}^{(\omega)}$	embedding $\mathbb{Z}[\omega][W, U]/(W^{n_2/2} - \omega,\ U^{2n_1} + 1)$	auxiliary bivariate ring
$\psi^{(\mathrm{i})}$	$\mathcal{R}^{(\mathrm{i})} \to \mathbb{C}^{n_1 \times n_2}$	encoding map to $n_1 \times n_2$ slots
$\psi^{(\omega)}$	$\mathcal{R}^{(\omega)} \to \mathbb{C}^{(n_2/2) \times (2n_1)}$	encoding map to $(n_2/2) \times (2n_1)$ slots
$\boldsymbol{D}_1$	U-axis DFT matrix of size $n_1 \times n_1$	
$\boldsymbol{D}_2$	V-axis DFT matrix of size $n_2 \times n_2$	$\boldsymbol{D}_2 = \begin{bmatrix} \boldsymbol{D}_2' & \omega\boldsymbol{D}_2' \\ \overline{\boldsymbol{D}_2'} & \omega^2\overline{\boldsymbol{D}_2'} \end{bmatrix}$, $\boldsymbol{D}_2' \in \mathbb{C}^{\frac{n_2}{2} \times \frac{n_2}{2}}$
$\mathcal{S}^{(\mathrm{i})}$	$\mathcal{R}^{(\mathrm{i})}[V]/(V^{n_1} - \mathrm{i}) =$ $\mathbb{Z}[\mathrm{i}][U, V, W]/(U^{n_1} - \mathrm{i}, V^{n_1} - \mathrm{i}, W^{n_2} + W^{n_2/2} + 1)$	trivariate extension ring over $\mathcal{R}^{(\mathrm{i})}$
$\mathcal{S}^{(\omega)}$	$\mathcal{R}^{(\omega)}[Y]/(Y^{n_2/2} - \omega) =$ $\mathbb{Z}[\omega][W, Y, U]/(W^{n_2/2} - \omega, Y^{2n_1} - \omega, U^{2n_1} + 1)$	auxiliary extension ring over $\mathcal{R}^{(\omega)}$ for bootstrapping.
$\Psi^{(\mathrm{i})}$	embedding $\mathcal{S}^{(\mathrm{i})} \to \mathcal{M}_{n_1}(\mathbb{C})^{n_2}$	n_2 matrices of size $n_1 \times n_1$.
$\Psi^{(\omega)}$	embedding $\mathcal{S}^{(\omega)} \to \mathcal{M}_{n_2/2}(\mathbb{C})^{2n_1}$	embedding $2n_1$ matrices of size $(n_2/2) \times (n_2/2)$.

Importantly, $\mathcal{S}^{(\omega)}$ mirrors the algebraic structure of $\mathcal{S}^{(\mathrm{i})}$ under the substitution $(U, V, W, \mathrm{i}) \leftrightarrow (W, Y, U, \omega)$. Defining analogously for $\alpha, \beta \in \mathcal{S}^{(\omega)}$,

$$\alpha \circledast \beta = \mathsf{Tr}_Z\big(\alpha(W, Z, U) \cdot \overline{\beta}(Y^{-1}, Z^{-1}, U^{-1})\big),$$

together with the coefficient ring $\mathcal{T}^{(\omega)} := \mathbb{Z}[\omega][U]/(U^{2n_1} + 1)$, we obtain the analogous batch matrix multiplication property over $\mathcal{M}_{n_2/2}(\mathbb{C})$ with batch size $2n_1$.

4.2 Reduction-based DFT over Extension Rings

We recall that homomorphic DFT decomposes into two smaller DFT layers, which we now formalize. Denote by $\boldsymbol{D}_1 = (\zeta_i^j)_{i,j} \in \mathbb{C}^{n_1 \times n_1}$ and $\boldsymbol{D}_2 = (\eta_i^j)_{i,j} \in \mathbb{C}^{n_2 \times n_2}$ the n_1-dimensional and n_2-dimensional DFT matrices, respectively. The following theorem and corollary make this precise, serving as the starting point of our construction.

Theorem 5 (Two-dimensional DFT)**.** *Let $m = \sum_{i,k} m_{i,k} U^i W^k \in \mathcal{R}^{(\mathrm{i})}$, and write $\boldsymbol{M} = (m_{i,k})_{i,k} \in \mathbb{Z}[\mathrm{i}]^{n_1 \times n_2}$. Then*

$$\psi^{(\mathrm{i})}(m) = \boldsymbol{D}_1 \cdot \boldsymbol{M} \cdot \boldsymbol{D}_2^T.$$

Proof. By definition, $m(\zeta_i, \eta_k) = \sum_{j,\ell} m_{j,\ell} (\zeta_i)^j (\eta_k)^\ell$ and we obtain

$$m(\zeta_i, \eta_k) = \sum_{j,\ell} (\boldsymbol{D}_1)_{i,j}\, m_{j,\ell}\, (\boldsymbol{D}_2^T)_{\ell,k} = (\boldsymbol{D}_1 \boldsymbol{M} \boldsymbol{D}_2^T)_{i,k},$$

which proves the claim. □

Corollary 2. *Let $m \in \mathcal{R}^{(\mathrm{i})}$, and let $m_{\mathrm{stc}}, m_{\mathrm{cts}} \in \mathcal{R}^{(\mathrm{i})}$ denote the outputs of StC and CtS applied to m, respectively. Then we have*

$$\psi^{(\mathrm{i})}(m_{\mathrm{stc}}) = \boldsymbol{D}_1 \cdot \psi^{(\mathrm{i})}(m) \cdot \boldsymbol{D}_2^T, \qquad \psi^{(\mathrm{i})}(m_{\mathrm{cts}}) = \boldsymbol{D}_1^{-1} \cdot \psi^{(\mathrm{i})}(m) \cdot (\boldsymbol{D}_2^T)^{-1}.$$

From Corollary 2, one can conclude that StC and CtS can be realized as matrix multiplications on the coefficient matrices of the input polynomials. In particular, StC turns the slot matrix of m into the coefficient matrix of m_{stc}, i.e., $\boldsymbol{M}_{\mathrm{stc}} = \psi^{(\mathrm{i})}(m)$, so $\psi^{(\mathrm{i})}(m_{\mathrm{stc}}) = \boldsymbol{D}_1 \boldsymbol{M}_{\mathrm{stc}} \boldsymbol{D}_2^T$; CtS is the inverse map.

Our goal is to realize Corollary 2 as batch matrix multiplication. Suppose we have a ciphertext (b, a) over $\mathcal{S}^{(\mathrm{i})}$ of the form

$$b + a \cdot s \approx m \in \mathcal{S}_q^{(\mathrm{i})} \quad \Leftrightarrow \quad b_j + a_j \cdot s \approx m_j \in \mathcal{R}_q^{(\mathrm{i})} \ \text{ for } \ j \in [n_1],$$

where $m = \sum_j m_j V^j$ for $m_j \in \mathcal{R}^{(\mathrm{i})}$, and similarly for $a, b \in \mathcal{S}^{(\mathrm{i})}$. We aim to compute $m' = \sum_j m'_j V^j$ such that

$$\psi^{(\mathrm{i})}(m'_j) = \boldsymbol{D}_1 \cdot \psi^{(\mathrm{i})}(m_j) \cdot \boldsymbol{D}_2^T,$$

and CtS aims to obtain $m'' = \sum_j m''_j V^j$ such that

$$\psi^{(\mathrm{i})}(m''_j) = \boldsymbol{D}_1^{-1} \cdot \psi^{(\mathrm{i})}(m_j) \cdot (\boldsymbol{D}_2^T)^{-1}.$$

Batch Left Multiplication by $\boldsymbol{D}_1$. We first observe that the batch PCMM framework of $\mathcal{S}^{(\mathrm{i})}$ is formulated as left multiplication on (U, V)-matrices batched along the W-axis, whereas our goal is to perform left multiplication by $\boldsymbol{D}_1$ along the V-axis. Since the same $\boldsymbol{D}_1$ is applied to every batch, the left multiplier does not depend on the batching axis, and the batch MM framework directly handles this multiplication, as we now make precise.

For $a \in \mathcal{S}^{(\mathrm{i})}$, define $a^*(U, V, W) := \bar{a}(V^{-1}, U^{-1}, W^{-1})$. Then it follows that

$$\Psi^{(\mathrm{i})}(a^*) = \big(\bar{a}(\zeta_j^{-1}, \zeta_i^{-1}, \eta_k^{-1})\big)_{i,j,k} = \big(\bar{a}(\bar{\zeta}_j, \bar{\zeta}_i, \bar{\eta}_k)\big)_{i,j,k} = \big(\overline{a(\zeta_j, \zeta_i, \eta_k)}\big)_{i,j,k}$$

so $\Psi^{(\mathrm{i})}_{3,k}(a^*) = \big(\Psi^{(\mathrm{i})}_{3,k}(a)\big)^*$. We also denote

$$d_1 := \big(\Psi^{(\mathrm{i})}\big)^{-1} (\{n_1 \cdot \boldsymbol{D}_1\}_{k=1}^{n_2}) \in \mathcal{S}^{(\mathrm{i})}.$$

The following proposition shows how to multiply $\boldsymbol{D}_1$ on the left in a batched manner.

Proposition 2 (Batch Left-multiplication by $\boldsymbol{D_1}$)**.** *For a polynomial* $m = \sum m_j(U,W)V^j \in \mathcal{S}^{(\mathrm{i})}$*, we denote* $m' = d_1 \circledast m^* \in \mathcal{S}^{(\mathrm{i})}$ *and write* $m' = \sum_j m'_j(U,W)V^j$*. Then, we have*

$$\psi^{(\mathrm{i})}(m'_j) = \boldsymbol{D}_1 \cdot \psi^{(\mathrm{i})}(m_j), \qquad \forall j \in [n_1].$$

Proof. We provide the proof in Appendix B.2. □

Hence, the $\boldsymbol{D}_1$ part of the homomorphic DFT reduces to a batch MM over plaintexts, accelerated by high-performance libraries at the cost of one multiplicative level. The only remaining key-switching costs come from (1) homomorphically evaluating $(\cdot)^*$, and (2) the $d \circledast (\cdot)$ operation itself, where the secret $s \in \mathcal{R}^{(\mathrm{i})}$ switches to $s^* := \overline{s}(V^{-1}, W^{-1})$ and must be switched back. We note that Proposition 2 extends to the inverse $\boldsymbol{D}_1^{-1}$ by replacing d_1 with

$$d_1^{(-1)} := (\Psi^{(\mathrm{i})})^{-1}\big(\{n_1 \cdot \boldsymbol{D}_1^{-1}\}_{k=1}^{n_2}\big) \in \mathcal{S}^{(\mathrm{i})}.$$

We refer to Algorithm 1 for the entire procedure.

Algorithm 1: $\mathsf{PCMM}(d, \mathsf{ct})$

Input: $\mathsf{ct} = (b,a) = \mathsf{Enc}_s(m) \in \mathcal{S}_Q^2$, $d \in \mathcal{S}$, where $\mathcal{S}$ is either $\mathcal{S}^{(\mathrm{i})}$ or $\mathcal{S}^{(\omega)}$
Output: $\mathsf{ct}_{\mathrm{out}} = \mathsf{Enc}_s(d \circledast m^*) \in \mathcal{S}^2_{Q/\Delta}$.

$\mathsf{ct}^* \leftarrow (b^*, a^*)$
$\mathsf{ct}_1 = (b_1, a_1) \leftarrow \mathsf{Switch}_{s^* \to s}(\mathsf{ct}^*)$
$\mathsf{ct}_2 \leftarrow (d \circledast b_1, d \circledast a_1)$ ▷ Reduction to PPMM
$\mathsf{ct}_3 = (b_3, a_3) \leftarrow \mathsf{Switch}_{s^* \to s}(\mathsf{ct}_2)$
$\mathsf{ct}' \leftarrow \mathsf{Rescale}(\mathsf{ct}_3; \Delta) \in \mathcal{S}^2_{Q/\Delta}$
return ct'

Batch Right Multiplication by $\boldsymbol{D_2^T}$ and $(\boldsymbol{D_2^T})^{-1}$. Right multiplications do not admit the same axis-invariance: the right multiplication by $\boldsymbol{D}_2^T$ acts along the W-axis, which coincides with the batching axis of $\mathcal{S}^{(\mathrm{i})}$ and cannot be reformulated as a left multiplication within $\mathcal{S}^{(\mathrm{i})}$. We resolve this asymmetry by introducing the auxiliary ring structures $\mathcal{R}^{(\omega)}$ and $\mathcal{S}^{(\omega)}$ in Section 4.1, which allow the right multiplication to be realized as left multiplication by block $\boldsymbol{D}'_2 \in \mathcal{M}_{n_2/2}(\mathbb{C})$. More precisely, we can decompose the matrices as:

$$\boldsymbol{D}_2 = \begin{bmatrix} \boldsymbol{D}'_2 & \omega \boldsymbol{D}'_2 \\ \overline{\boldsymbol{D}'_2} & \omega^2 \overline{\boldsymbol{D}'_2} \end{bmatrix}, \quad \boldsymbol{D}_2^{-1} = \frac{2}{n_2(\omega^2 - \omega)} \begin{bmatrix} \omega^2 \overline{\boldsymbol{D}'_2}^T & -\omega (\boldsymbol{D}'_2)^T \\ -\overline{\boldsymbol{D}'_2}^T & (\boldsymbol{D}'_2)^T \end{bmatrix} \in \mathcal{M}_{n_2}(\mathbb{C}).$$

The following theorem gives both transfer identities.

Theorem 6 (Transfer of right multiplication by $\boldsymbol{D_2^T}$ and $\boldsymbol{(D_2^T)^{-1}}$)**.** *Let* $m \in \mathcal{R}^{(\mathrm{i})}$*, and set*

$$m'(U,W) = m(U, W^{-1}) \in \mathcal{R}^{(\mathrm{i})}, \quad \widetilde{m}(W,U) = m(U,W) \in \mathcal{R}^{(\omega)},$$
$$\widetilde{m}'(W,U) = m'(W,U) \in \mathcal{R}^{(\omega)}.$$

Then the following identities hold, where $\mathbf{1}_{s,t}$ *denotes the all-one matrix of size* $s \times t$*.*

(i) *Let* $\mathsf{pt}_{stc} \in \mathcal{R}^{(\omega)}$ *satisfy*

$$\psi^{(\omega)}(\mathsf{pt}_{stc}) = \begin{bmatrix} \mathbf{1}_{\frac{n_2}{2} \times n_1} & \omega \mathbf{1}_{\frac{n_2}{2} \times n_1} \end{bmatrix} \in \mathbb{C}^{\frac{n_2}{2} \times 2n_1}.$$

Define

$$\mathsf{pt}'_{stc}(W,U) = \mathsf{pt}_{stc}(W, U^{-1}) \in \mathcal{R}^{(\omega)}.$$

Then

$$\mathcal{T}_{i \to \omega}\left(\psi^{(\mathrm{i})}(m)\boldsymbol{D}_2^T\right) = \boldsymbol{D}'_2\, \psi^{(\omega)}(\mathsf{pt}_{stc} \cdot \widetilde{m}) + \boldsymbol{D}'_2\, \psi^{(\omega)}(\mathsf{pt}'_{stc} \cdot \widetilde{m}').$$

(ii) *Choose* $\mathsf{pt}_{cts} \in \mathcal{R}^{(\omega)}$ *such that*

$$\psi^{(\omega)}(\mathsf{pt}_{cts}) = \frac{2}{n_2(\omega^2-\omega)} \left[\omega^2 \mathbf{1}_{\frac{n_2}{2}\times n_1} \quad -\mathbf{1}_{\frac{n_2}{2}\times n_1}\right].$$

Define

$$\mathsf{pt}'_{cts}(W,U) := -\mathsf{pt}_{cts}(W^{-1},U^{-1}).$$

Then

$$\mathcal{T}_{i\to\omega}\left(\psi^{(\mathrm{i})}(m)(\boldsymbol{D}_2^T)^{-1}\right) = (\boldsymbol{D}'_2)^*\,\psi^{(\omega)}(\mathsf{pt}_{cts}\widetilde{m}) - (\boldsymbol{D}'_2)^T\,\psi^{(\omega)}(\mathsf{pt}'_{cts}\widetilde{m}').$$

Proof. The proof is given in Appendix B.3. □

With Theorem 6 in place, DFT_{m_2} and its inverse can now be evaluated as a batch PCMM over $\mathcal{S}^{(\omega)}$. The only remaining technicality is the handling of the plaintext terms $\mathsf{pt}_{\mathrm{stc}}, \mathsf{pt}'_{\mathrm{stc}}$ appearing in the theorem, which we address as follows. We start by observing the following chain:

$$\begin{aligned} b + a\cdot s \approx m \in \mathcal{S}^{(\mathrm{i})} &\Leftrightarrow b_j + a_j \cdot s \approx m_j \in \mathcal{R}^{(\mathrm{i})}, \quad \forall j \in [n_1], \\ &\Leftrightarrow \widetilde{b}_j + \widetilde{a}_j \cdot s \approx \widetilde{m}_j \in \mathcal{R}^{(\omega)}, \quad \forall j \in [n_1] \\ &\Leftrightarrow \lceil 2n_1/n_2 \rceil \ \text{ ciphertexts } \ \{\widetilde{b} + \widetilde{a}\cdot s \approx \widetilde{m} \in \mathcal{S}^{(\omega)}\}, \end{aligned}$$

where $\widetilde{m}_j(W,U) := m_j(U,W) \in \mathcal{R}^{(\omega)}$ and similarly for a_j, b_j. The tilde notation permutes the coefficient ordering under the variable swap $(U,W) \leftrightarrow (W,U)$, leaving the secret invariant across the ring presentations. We denote the conversion from $\mathcal{S}^{(\mathrm{i})}$ to $\mathcal{S}^{(\omega)}$ by $\mathsf{RingConv}_{\mathrm{i}\to\omega}$, and its inverse from $\mathcal{S}^{(\omega)}$ to $\mathcal{S}^{(\mathrm{i})}$ by $\mathsf{RingConv}_{\omega\to\mathrm{i}}$.

The final step partitions the $\widetilde{m}_j$'s into blocks of size $n_2/2$ along the V-axis, packing each block into a single ciphertext over $\mathcal{S}^{(\omega)}$. We denote this conversion procedure by $\mathsf{RingConv}_{\mathrm{i}\to\omega}$ and its inverse by $\mathsf{RingConv}_{\omega\to\mathrm{i}}$. Figure 1 illustrates this ring conversion and the resulting factorization of the forward DFT.

(a) $\mathsf{RingConv}_{\mathrm{i}\to\omega}$

$$\begin{array}{ccccccc} & & \overset{\text{split in } V}{\longrightarrow} & & \overset{\text{Ring Conv.}}{\longrightarrow} & & \overset{\text{pack in } Y}{\longrightarrow} \\ \mathcal{S}^{(\mathrm{i})} & & (\mathcal{R}^{(\mathrm{i})})^{n_1} & & (\mathcal{R}^{(\omega)})^{n_1} & & (\mathcal{S}^{(\omega)})^{L} \\ m = \sum_{j=0}^{n_1-1} m_j V^j & & (m_0,\ldots,m_{n_1-1}) & & (\widetilde{m}_0,\ldots,\widetilde{m}_{n_1-1}) & & \left(\widetilde{m}^{(t)}\right)_{0\le t<L} \end{array}$$

$$\psi^{(\mathrm{i})}(m_j) = \underbrace{[\boldsymbol{E}_{j,1} \mid \boldsymbol{E}_{j,2}]}_{n_1\times n_2} \xrightarrow{\mathcal{T}_{i\to\omega}} \psi^{(\omega)}(\widetilde{m}_j) = \underbrace{[\boldsymbol{E}_{j,1}^T \mid \overline{\boldsymbol{E}}_{j,2}^T]}_{(n_2/2)\times 2n_1}$$

(b) $\mathsf{DFT}(\mu)$

$$\boldsymbol{M} \xrightarrow[\text{over } \mathcal{S}^{(\mathrm{i})}]{\boldsymbol{D}_1\cdot} \boldsymbol{D}_1\boldsymbol{M} = [\boldsymbol{E}_1 \mid \boldsymbol{E}_2] \xrightarrow{\mathcal{T}_{i\to\omega}} [\boldsymbol{E}_1^T \mid \overline{\boldsymbol{E}}_2^T] \xrightarrow{\text{lin. sum}} \boldsymbol{M}' \xrightarrow[\text{over } \mathcal{S}^{(\omega)}]{\boldsymbol{D}'_2\cdot} \boldsymbol{D}'_2\boldsymbol{M}' \xrightarrow{\mathcal{T}^{-1}_{i\to\omega}} \boldsymbol{D}_1\boldsymbol{M}\boldsymbol{D}_2^T$$

Figure 1: Illustration of (a) Ring conversion from $\mathcal{S}^{(\mathrm{i})}$ to $\mathcal{S}^{(\omega)}$, and (b) Factorization of the forward DFT through the auxiliary ring. Here $\boldsymbol{M}' = [\boldsymbol{E}_1^T \mid \omega\overline{\boldsymbol{E}}_2^T] + [\omega\boldsymbol{E}_2^T \mid \overline{\boldsymbol{E}}_1^T]$.

Analogously to the $\mathcal{S}^{(\mathrm{i})}$, we define

$$\beta^*(W,Y,U) := \bar{\beta}(Y^{-1},W^{-1},U^{-1}),$$

which gives $\Psi^{(\omega)}_{3,k}(\beta^*) = \left(\Psi^{(\omega)}_{3,k}(\beta)\right)^*$, and let

$$d'_2 = \left(\Psi^{(\omega)}\right)^{-1}\left(\{(n_2/2)\cdot \boldsymbol{D}'_2\}_{k=1}^{2n_1}\right) \in \mathcal{S}^{(\omega)}.$$

The batch left multiplication by $\boldsymbol{D}'_2$ is then realized as follows.

Algorithm 2(a): StC

Input: $\mathsf{ct} \in \left(\mathcal{S}_Q^{(\mathrm{i})}\right)^2$, $\mathsf{aux}^{(\mathsf{stc})} = \left(d_1, d_2', \mathsf{pt}_{\mathrm{stc}}, \mathsf{pt}_{\mathrm{stc}}'\right) \in \mathcal{S}^{(\mathrm{i})} \times (\mathcal{S}^{(\omega)})^3$

Output: $\mathsf{ct}_{\mathrm{out}} \in \left(\mathcal{S}_{Q/\Delta^2}^{(\mathrm{i})}\right)^2$.

$\mathsf{ct}' = (b = \sum_j b_j V^j, a = \sum_j a_j V^j) \leftarrow \mathsf{PCMM}(d_1, \mathsf{ct})$
$\mathsf{ct}'' \leftarrow \mathsf{EvalAuto}_{W \mapsto W^{-1}}(\mathsf{ct}')$
$L \leftarrow \lceil 2n_1/n_2 \rceil$
$\widetilde{\mathsf{ct}} = \{\widetilde{\mathsf{ct}}^{(t)}\}_{t\in[L]} \leftarrow \mathsf{RingConv}_{\mathrm{i}\to\omega}(\mathsf{ct}')$
$\widetilde{\mathsf{ct}}' = \{(\widetilde{\mathsf{ct}}^{(t)})'\}_{t\in[L]} \leftarrow \mathsf{RingConv}_{\mathrm{i}\to\omega}(\mathsf{ct}'')$
for $t \leftarrow 0$ **to** $L-1$ **do**
  $\widetilde{\mathsf{ct}}_1^{(t)} \leftarrow \mathsf{PCMM}\left(\mathsf{pt}_{\mathrm{stc}} \cdot d_2', \widetilde{\mathsf{ct}}^{(t)}\right)$
  $\widetilde{\mathsf{ct}}_2^{(t)} \leftarrow \mathsf{PCMM}\left(\mathsf{pt}_{\mathrm{stc}}' \cdot d_2', (\widetilde{\mathsf{ct}}^{(t)})'\right)$
  $\widetilde{\mathsf{ct}}^{(t)\prime\prime} \leftarrow \widetilde{\mathsf{ct}}_1^{(t)} + \widetilde{\mathsf{ct}}_2^{(t)}$
end for
$\widetilde{\mathsf{ct}}'' \leftarrow \{\widetilde{\mathsf{ct}}^{(t)\prime\prime}\}_{t\in[L]}$
$\mathsf{ct}_{\mathrm{out}} \leftarrow \mathsf{RingConv}_{\omega\to\mathrm{i}}\left(\widetilde{\mathsf{ct}}''\right)$
return ct_{out};

Algorithm 2(b): CtS

Input: $\mathsf{ct} \in \left(\mathcal{S}_Q^{(\mathrm{i})}\right)^2$, $\mathsf{aux}^{(\mathsf{cts})} = \left(d_1^{(-1)}, d_2^{(*)}, \mathsf{pt}_{\mathrm{cts}}, \mathsf{pt}_{\mathrm{cts}}', d_2^{(T)}\right) \in \mathcal{S}^{(\mathrm{i})} \times (\mathcal{S}^{(\omega)})^4$

Output: $\mathsf{ct}_{\mathrm{out}} \in \left(\mathcal{S}_{Q/\Delta^2}^{(\mathrm{i})}\right)^2$.

$\mathsf{ct}' = (b = \sum_j b_j V^j, a = \sum_j a_j V^j) \leftarrow \mathsf{PCMM}(d_1^{(-1)}, \mathsf{ct})$
$\mathsf{ct}'' \leftarrow \mathsf{EvalAuto}_{W \mapsto W^{-1}}(\mathsf{ct}')$
$L \leftarrow \lceil 2n_1/n_2 \rceil$
$\widetilde{\mathsf{ct}} = \{\widetilde{\mathsf{ct}}^{(t)}\}_{t\in[L]} \leftarrow \mathsf{RingConv}_{\mathrm{i}\to\omega}(\mathsf{ct}')$
$\widetilde{\mathsf{ct}}' = \{(\widetilde{\mathsf{ct}}^{(t)})'\}_{t\in[L]} \leftarrow \mathsf{RingConv}_{\mathrm{i}\to\omega}(\mathsf{ct}'')$
for $t \leftarrow 0$ **to** $L-1$ **do**
  $\widetilde{\mathsf{ct}}_1^{(t)} \leftarrow \mathsf{PCMM}\left(\mathsf{pt}_{\mathrm{cts}} \cdot d_2^{(*)}, \widetilde{\mathsf{ct}}^{(t)}\right)$
  $\widetilde{\mathsf{ct}}_2^{(t)} \leftarrow \mathsf{PCMM}\left(\mathsf{pt}_{\mathrm{cts}}' \cdot d_2^{(T)}, (\widetilde{\mathsf{ct}}^{(t)})'\right)$
  $\widetilde{\mathsf{ct}}^{(t)\prime\prime} \leftarrow \widetilde{\mathsf{ct}}_1^{(t)} - \widetilde{\mathsf{ct}}_2^{(t)}$
end for
$\widetilde{\mathsf{ct}} \leftarrow \{\widetilde{\mathsf{ct}}^{(t)\prime\prime}\}_{t\in[L]}$
$\mathsf{ct}_{\mathrm{out}} \leftarrow \mathsf{RingConv}_{\omega\to\mathrm{i}}\left(\widetilde{\mathsf{ct}}\right)$
return ct_{out};

Proposition 3 (Batch Left-multiplication by $\boldsymbol{D}_2'$)**.** *For a polynomial $m = \sum m_j(W,U)Y^j \in \mathcal{S}^{(\omega)}$, we denote $m' = d_2' \circledast m^* \in \mathcal{S}^{(\omega)}$ and write $m' = \sum_j m_j'(W,U)Y^j$. Then, we have*

$$\psi^{(\omega)}(m_j') = \boldsymbol{D}_2' \cdot \psi^{(\omega)}(m_j)$$

for all $j \in [n_2/2]$.

Proof. We refer the proof to Appendix B.2. □

To merge the plaintext multiplication with the $\circledast$ operation, note that $\mathsf{pt}_{\mathrm{stc}} \in \mathcal{R}^{(\omega)}$ depends solely on U, i.e., $\mathsf{pt}_{\mathrm{stc}} = \mathsf{pt}_{\mathrm{stc}}(U)$. A direct computation gives

$$(\mathsf{pt}_{\mathrm{stc}} \cdot d_2') \circledast \widetilde{m}^* = d_2' \circledast (\mathsf{pt}_{\mathrm{stc}} \cdot \widetilde{m})^*, \quad (\mathsf{pt}_{\mathrm{stc}}' \cdot d_2') \circledast \widetilde{m}'^* = d_2' \circledast (\mathsf{pt}_{\mathrm{stc}}' \cdot \widetilde{m}')^*,$$

where an encryption of $\widetilde{m}'$ is obtained by evaluating the automorphism $W \mapsto W^{-1}$,

$$\widetilde{b}' + \widetilde{a}' \cdot s \approx \widetilde{m}'(W,Y,U) = \widetilde{m}(W^{-1},Y,U). \tag{9}$$

Hence, the operation $\circledast$ and the PCMM merge into a single step, consuming only one ciphertext level. Applying Proposition 3 to each block and summing the two resulting ciphertexts within that block, we obtain

$$\widetilde{b}_{\mathrm{out}} + \widetilde{a}_{\mathrm{out}} \cdot \widetilde{s} \approx d_2' \circledast (\mathsf{pt}_{\mathrm{stc}} \cdot \widetilde{m})^* + d_2' \circledast (\mathsf{pt}_{\mathrm{stc}}' \cdot \widetilde{m}')^* \in \mathcal{S}^{(\omega)}.$$

Reversing the above packing chain yields $b_{\mathrm{out}} + a_{\mathrm{out}} \cdot s \approx m_{\mathrm{out}} \in \mathcal{S}^{(\mathrm{i})}$ with $m_{\mathrm{out}} = \sum_j (m_{\mathrm{out}})_j V^j$ satisfying

$$\psi^{(\mathrm{i})}\left((m_{\mathrm{out}})_j\right) = \psi^{(\mathrm{i})}(m_j) \cdot \boldsymbol{D}_2^T$$

by Theorem 6. For the case of CtS, we denote

$$d_2^{(T)} = \left(\Psi^{(\omega)}\right)^{-1}\left(\{(n_2/2) \cdot \boldsymbol{D}_2'^T\}_{k=1}^{2n_1}\right), \quad d_2^{(*)} = \left(\Psi^{(\omega)}\right)^{-1}\left(\{(n_2/2) \cdot \boldsymbol{D}_2'^*\}_{k=1}^{2n_1}\right) \in \mathcal{S}^{(\omega)}.$$

The resulting StC and CtS are summarized side by side in Algorithms 2(a) and 2(b).

4.3 Bootstrapping Optimizations

We optimize the complete bootstrapping circuit while preserving the transforms of Sections 4.1–4.2 and their multiplicative depth. In our experiments, the boundary and modulus-raising optimizations of [CGK+26] are applied to both our construction and theirs, whereas branch fusion and the associated hoisting are specific to our construction, as they exploit its auxiliary-ring structure.

Canceling boundary key-switchings. Following [CGK+26], we cancel a pair of key-switchings at the boundaries of the circuit. First, since $\boldsymbol{D}_1$ and $\boldsymbol{D}_2^T$ multiply from opposite sides, we may reorder the two stages of each transform so that the $\mathcal{S}^{(\mathrm{i})}$ stages are exposed at the boundaries: CtS applies $\boldsymbol{D}_1^{-1}$ before $(\boldsymbol{D}_2^T)^{-1}$, and StC applies $\boldsymbol{D}_2^T$ before $\boldsymbol{D}_1$. Each PCMM over $\mathcal{S}^{(\mathrm{i})}$ evaluates $\boldsymbol{x} \mapsto d \circledast \boldsymbol{x}^*$ with two key-switchings (Algorithm 1): one after forming ct^* (line 2), and one after the product (line 4), since $\circledast$ changes the secret of its right operand. To cancel some of these key-switchings, we refresh $\boldsymbol{m}^*$ and transpose the result back. Its first and last $\mathcal{S}^{(\mathrm{i})}$ products then evaluate

$$\boldsymbol{m} \mapsto d_1^{(-1)} \circledast (\boldsymbol{m}^*)^* = d_1^{(-1)} \circledast \boldsymbol{m}, \qquad \boldsymbol{y} \mapsto (d_1 \circledast \boldsymbol{y}^*)^* = \boldsymbol{y}^* \circledast d_1,$$

where the second identity follows from Theorem 2. In the first, the lines 1–2 of PCMM are skipped, sparing a key-switch. The second, after lines 1–2, is a product with the ciphertext as the left operand of $\circledast$, which leaves its secret unchanged. Hence, key-switching of line 4 is skipped as well. For a full block of n_1 components, this saves one $\mathcal{S}^{(\mathrm{i})}$ key-switching in each transform (Algorithm 3, lines 3–4 and 28–29).

Raising the modulus to pQ. After the above optimization, no key-switching precedes the first CtS product. Following [CGK+26], we raise modulus to pQ, encode the plaintext of this product at scale p, and rescale by p, for a prime $p = P_b$ of the auxiliary modulus $P = \prod_b P_b$. The other key-switch in $\mathcal{S}^{(\mathrm{i})}$ stage also restores the dense secret after sparse-secret encapsulation [BTPH22]:

$$\begin{aligned} \mathsf{ct}_{\mathrm{mr}} &= \mathsf{ModRaise}_{pQ}\big(\mathsf{KeySwitch}_{s \to s'}(\mathsf{ct})\big), \\ \mathsf{ct}' &= \mathsf{KeySwitch}_{\bar{s}'(V^{-1}, W^{-1}) \to s}\big(\mathsf{Rescale}(d_1^{(-1)} \circledast \mathsf{ct}_{\mathrm{mr}};\, p)\big). \end{aligned}$$

Here s' is sparse to reduce the overflow removed by $\mathsf{EvalMod}$, and the dense-to-sparse key-switch happens at the base modulus q_0. This optimization saves one level and a separate sparse-to-dense key-switch, without any switching key at pQ.

Sharing work between the auxiliary-ring branches. By Theorem 6, right multiplications by $\boldsymbol{D}_2^T$ and $(\boldsymbol{D}_2^T)^{-1}$ in StC and CtS is computed as

$$\begin{aligned} \mathcal{T}_{i \to \omega}\big(\psi^{(i)}(m)\, \boldsymbol{D}_2^T\big) &= \boldsymbol{D}_2'\, \psi^{(\omega)}(\mathsf{pt}_{\mathrm{stc}}\, \tilde{m}) + \boldsymbol{D}_2'\, \psi^{(\omega)}(\mathsf{pt}'_{\mathrm{stc}}\, \tilde{m}'), \\ \mathcal{T}_{i \to \omega}\big(\psi^{(i)}(m)\, (\boldsymbol{D}_2^T)^{-1}\big) &= (\boldsymbol{D}_2')^*\, \psi^{(\omega)}(\mathsf{pt}_{\mathrm{cts}}\, \tilde{m}) - (\boldsymbol{D}_2')^T\, \psi^{(\omega)}(\mathsf{pt}'_{\mathrm{cts}}\, \tilde{m}'), \end{aligned}$$

where $\tilde{m}' = \tilde{m}(W^{-1}, U)$. Each consists of two branches, one acting on $\tilde{m}$ and one on $\tilde{m}'$. Evaluated separately, as in Algorithms 2(a) and 2(b), each transform costs a W-inversion automorphism and two PCMMs, i.e., two matrix products and four $\mathcal{S}^{(\omega)}$ key-switchings per block. In CtS, different matrices are multiplied in the branches, so we keep both products but share the work around them; in StC, both branches multiply $\boldsymbol{D}_2'$, so we add the inputs before the product.

First, we reduce the input side and output side work of CtS as follows. On the input side, the second branch applies the W-inversion then computes the map $m \mapsto m^*$ in PCMM, each involving a key-switching. We fuse the two into a single automorphism, so that its input becomes

$$\widetilde{\mathsf{ct}}_2 = \mathsf{KeySwitch}_{s(U^{-1}, Y) \to s}\big(\widetilde{\mathsf{ct}}(Y, W^{-1}, U^{-1})\big),$$

with one key-switching instead of two. On the output side, the final key-switchings of the two PCMMs (Algorithm 1, line 4) switch from the same secret $\bar{s}(U^{-1}, Y^{-1})$ to s, so we defer it until after the subtraction and perform it once.

In StC, both branches multiply by $\boldsymbol{D}'_2$, so by linearity they can share a single product. We compute the sum of inputs, pack it into $\mathcal{S}^{(\omega)}$, then apply one PCMM with d'_2. This reduces the operation count to one matrix product and two $\mathcal{S}^{(\omega)}$ key-switchings per block. However, $\mathsf{pt}_{\mathrm{stc}}$ and $\mathsf{pt}'_{\mathrm{stc}}$ can then no longer be merged into separate PCMM plaintexts, and applying them to the inputs would cost a level. We avoid this by forming the sum from the EvalMod outputs as follows. The sum corresponds to $[\boldsymbol{E}_1 + \omega\boldsymbol{E}_2 \mid \boldsymbol{E}_1 + \omega^2\boldsymbol{E}_2]$, where $\boldsymbol{E} = [\boldsymbol{E}_1 \mid \boldsymbol{E}_2] = \psi^{(i)}(m)$ is the slot matrix of the message m refreshed by EvalMod. Since EvalMod handles the real and imaginary parts separately, we absorb constants into its final coefficients and let it output $m_{\mathrm{re}}, m_{\mathrm{im}}$ with $\psi^{(i)}(m_{\mathrm{re}}) = -\omega \operatorname{Re}\boldsymbol{E}$ and $\psi^{(i)}(m_{\mathrm{im}}) = -\mathrm{i}\omega \operatorname{Im}\boldsymbol{E}$. With $\tau_U \colon U \mapsto U^{-1}$, we then compute

$$\psi^{(i)}\Big(X^{N/2}(m_{\mathrm{re}}+m_{\mathrm{im}})+\tau_U\big(X^{N/2}(m_{\mathrm{re}}-m_{\mathrm{im}})\big)\Big) = [1 \mid \omega^2]\odot[\boldsymbol{E}_1 \mid \boldsymbol{E}_2]+[\omega \mid 1]\odot[\boldsymbol{E}_2 \mid \boldsymbol{E}_1],$$

which is the desired sum. This consumes no extra level, and the W-inversion is replaced by τ_U, which costs one base-ring key-switching per component (Algorithm 3, line 18).

Hoisting the initial key-switchings. The two initial key-switchings of each CtS block act on images of the same ciphertext $\widetilde{\mathsf{ct}}$: under $(\cdot)^*$ for the first branch, and under $m \mapsto m(Y, W^{-1}, U^{-1})$ for the second. Both maps are linear on coefficients, so, as in the hoisting of [HS18], we decompose $\tilde{a}$ in $\widetilde{\mathsf{ct}} = (\tilde{b}, \tilde{a})$ once and apply both maps to the resulting digits. This differs from the usual hoisting of rotations in two ways. First, the maps act on each coefficient in $\mathbb{Z}[\omega]$ by a unit $\pm\omega^k$ and possibly conjugation, which mix the coordinates in the $(1, \omega)$-basis; the mapped digits may thus exceed the usual digit bound by up to a factor of two, which at most doubles the key-switching noise contributed by the digits. Second, the NTT of a digit d evaluates its Y-variable at the roots of $Y^{n_2/2} = \omega$, but the second map requires $d(y, w^{-1}, u^{-1})$, where $(w^{-1})^{n_2/2} = \omega^2$; this value is thus not already computed. We therefore share the W- and U-passes and add one Y-pass on the other root set, so each digit needs one full NTT plus one Y-pass instead of two full NTTs. Thus, hoisting shares the decomposition completely and the NTTs almost completely, while the number of key-switchings is unchanged.

Algorithm 3 gives the resulting circuit for one $\mathcal{S}^{(\mathrm{i})}$ ciphertext packing n_1 base-ring ciphertexts, combining the boundary and modulus-raising optimizations, the shared work between the auxiliary-ring branches, and the hoisted initial switches.

5 Implementation Results

Runtime benchmarks were conducted on an Intel Xeon Gold 6542Y processor using a single thread, except for the 16-thread results reported in Table 7. Table 2 summarizes the performance-benchmark parameter sets; each provides 128-bit security according to the lattice estimator [APS15] (commit `8d38f52`). [1] Here, the notation (a, b) in each parameter denotes the ring degree N satisfies $N = 2^a 3^b$.

Throughout this section, *latency* denotes the execution time of a single operation, while *throughput* denotes the amortized runtime per slot, computed by dividing the latency by the number of slots processed by the operation. Note that *amortized time* represents

[1] Module-BKZ may improve the reduction quality over cyclotomic fields with non-power-of-two conductors [DEdP25], leading to a smaller equivalent blocksize than unstructured BKZ. Since existing estimators do not yet provide a complete concrete cost model for such module-aware attacks—including SVP-oracle costs, dimensions-for-free, and convergence effects—we report security with respect to the standard lattice-estimator model.

Algorithm 3: Complete bootstrapping circuit for Gentry–Lee

Input: $\mathsf{ct} = \mathsf{Enc}_s(m) \in \big(\mathcal{S}^{(\mathrm{i})}_{q_0}\big)^2$; auxiliary plaintexts $d_1, d_1^{(-1)} \in \mathcal{S}^{(\mathrm{i})}$, $d'_2, d_2'^{(*)} = \mathsf{pt}_{\mathrm{cts}} \cdot d_2^{(*)}$, $d_2'^{(T)} = \mathsf{pt}'_{\mathrm{cts}} \cdot d_2^{(T)} \in \mathcal{S}^{(\omega)}$; switching keys $\mathsf{swk}_{s\to s'}$, $\mathsf{swk}_{\bar{s}(U^{-1},W^{-1})\to s}$, $\mathsf{swk}_{s(U^{-1},W)\to s}$, $\mathsf{swk}_{s^2\to s}$ over $\mathcal{R}^{(\mathrm{i})}$ for a sparse secret s'; $\mathsf{swk}_{\bar{s}'(V^{-1},W^{-1})\to s}$, $\mathsf{swk}_{\bar{s}(V^{-1},W^{-1})\to s}$ over $\mathcal{S}^{(\mathrm{i})}$; $\mathsf{swk}_{\bar{s}(U^{-1},Y^{-1})\to s}$, $\mathsf{swk}_{s(U^{-1},Y)\to s}$ over $\mathcal{S}^{(\omega)}$.

Output: $\mathsf{ct}_{\mathrm{out}} = \mathsf{Enc}_s(m) \in \big(\mathcal{S}^{(\mathrm{i})}_{q}\big)^2$ for larger modulus $q > q_0$.

$\mathsf{ct}_0 \leftarrow \mathsf{KeySwitch}_{s\to s'}(\mathsf{ct}) \in \big(\mathcal{S}^{(\mathrm{i})}_{q_0}\big)^2$

$\mathsf{ct}_{\mathrm{mr}} = (b_{\mathrm{mr}}, a_{\mathrm{mr}}) \leftarrow \mathsf{ModRaise}(\mathsf{ct}_0) \in \big(\mathcal{S}^{(\mathrm{i})}_{pQ}\big)^2$

$\mathsf{ct}_1 \leftarrow \big(d_1^{(-1)} \circledast b_{\mathrm{mr}},\ d_1^{(-1)} \circledast a_{\mathrm{mr}}\big)$

$\mathsf{ct}_1 \leftarrow \mathsf{KeySwitch}_{\bar{s}'(V^{-1},W^{-1})\to s}\big(\mathsf{Rescale}(\mathsf{ct}_1;\, p)\big) \in \big(\mathcal{S}^{(\mathrm{i})}_{Q}\big)^2$

$L \leftarrow \lceil 2n_1/n_2 \rceil$; $\quad \{\widetilde{\mathsf{ct}}^{(\ell)} = (\tilde{b}_\ell, \tilde{a}_\ell) \in (\mathcal{S}^{(\omega)}_Q)^2\}_{\ell\in[L]} \leftarrow \mathsf{RingConv}_{\mathrm{i}\to\omega}(\mathsf{ct}_1)$

for $\ell \leftarrow 0$ **to** $L-1$ **do**

  $\widetilde{\mathsf{ct}}_1 \leftarrow \mathsf{KeySwitch}_{\bar{s}(U^{-1},Y^{-1})\to s}(\tilde{b}^*_\ell, \tilde{a}^*_\ell)$

  $\widetilde{\mathsf{ct}}_2 \leftarrow \mathsf{KeySwitch}_{s(U^{-1},Y)\to s}\big(\tilde{b}_\ell(Y, W^{-1}, U^{-1}), \tilde{a}_\ell(Y, W^{-1}, U^{-1})\big)$

  $\widetilde{\mathsf{ct}}_{\mathrm{acc}} \leftarrow d_2'^{(*)} \circledast \widetilde{\mathsf{ct}}_1 - d_2'^{(T)} \circledast \widetilde{\mathsf{ct}}_2$

  $\widetilde{\mathsf{ct}}''_\ell \leftarrow \mathsf{Rescale}\big(\mathsf{KeySwitch}_{\bar{s}(U^{-1},Y^{-1})\to s}(\widetilde{\mathsf{ct}}_{\mathrm{acc}})\big)$

end for

$\mathsf{ct}_2 = \big(\sum_j b_j V^j, \sum_j a_j V^j\big) \leftarrow \mathsf{RingConv}_{\omega\to\mathrm{i}}\big(\{\widetilde{\mathsf{ct}}''_\ell\}_{\ell\in[L]}\big)$

for $j \leftarrow 0$ **to** $n_1 - 1$ **do**

  $\mathsf{ct}_{\mathrm{conj}} \leftarrow \mathsf{KeySwitch}_{\bar{s}(U^{-1},W^{-1})\to s}\big(\bar{b}_j(U^{-1}, W^{-1}),\, \bar{a}_j(U^{-1}, W^{-1})\big)$

  $\mathsf{ct}_{\mathrm{re}} \leftarrow (b_j, a_j) + \mathsf{ct}_{\mathrm{conj}}$; $\quad \mathsf{ct}_{\mathrm{im}} \leftarrow \mathrm{i} \cdot \big(\mathsf{ct}_{\mathrm{conj}} - (b_j, a_j)\big)$

  $\mathsf{ct}'_{\mathrm{re}} \leftarrow -\omega \cdot \mathsf{EvalMod}(\mathsf{ct}_{\mathrm{re}})$; $\quad \mathsf{ct}'_{\mathrm{im}} \leftarrow -\mathrm{i}\,\omega \cdot \mathsf{EvalMod}(\mathsf{ct}_{\mathrm{im}})$

  $\mathsf{ct}_+ \leftarrow X^{N/2}\,(\mathsf{ct}'_{\mathrm{re}} + \mathsf{ct}'_{\mathrm{im}})$, $\quad \mathsf{ct}_- = (b_j^-, a_j^-) \leftarrow X^{N/2}\,(\mathsf{ct}'_{\mathrm{re}} - \mathsf{ct}'_{\mathrm{im}})$

  $(b'_j, a'_j) \leftarrow \mathsf{ct}_+ + \mathsf{KeySwitch}_{s(U^{-1},W)\to s}\big(b_j^-(U^{-1}, W), a_j^-(U^{-1}, W)\big)$

end for

$\mathsf{ct}_3 \leftarrow \big(\sum_j b'_j V^j, \sum_j a'_j V^j\big)$

$\{\widetilde{\mathsf{ct}}^{(\ell)} = (\tilde{b}_\ell, \tilde{a}_\ell) \in (\mathcal{S}^{(\omega)})^2\}_{\ell\in[L]} \leftarrow \mathsf{RingConv}_{\mathrm{i}\to\omega}(\mathsf{ct}_3)$

for $\ell \leftarrow 0$ **to** $L-1$ **do**

  $(\tilde{b}_\ell, \tilde{a}_\ell) \leftarrow \mathsf{KeySwitch}_{\bar{s}(U^{-1},Y^{-1})\to s}(\tilde{b}^*_\ell, \tilde{a}^*_\ell)$

  $\widetilde{\mathsf{ct}}^{(\ell)} \leftarrow (d'_2 \circledast \tilde{b}_\ell, d'_2 \circledast \tilde{a}_\ell)$

  $\widetilde{\mathsf{ct}}^{(\ell)\prime} \leftarrow \mathsf{Rescale}\big(\mathsf{KeySwitch}_{\bar{s}(U^{-1},Y^{-1})\to s}\big(\widetilde{\mathsf{ct}}^{(\ell)}\big)\big)$

end for

$(b_4, a_4) \leftarrow \mathsf{RingConv}_{\omega\to\mathrm{i}}\big(\{\widetilde{\mathsf{ct}}^{(\ell)\prime}\}_{\ell\in[L]}\big)$

$(b_5, a_5) \leftarrow \mathsf{KeySwitch}_{\bar{s}(V^{-1},W^{-1})\to s}(b^*_4, a^*_4)$

$\mathsf{ct}_{\mathrm{out}} \leftarrow \mathsf{Rescale}\big(b_5 \circledast d_1,\ a_5 \circledast d_1\big) \in \big(\mathcal{S}^{(\mathrm{i})}_{q}\big)^2$

return ct_{out}

the time per matrix in batched evaluation. All experiment results are the average of 100 repetitions unless otherwise specified.

HE operations benchmark. We benchmark several core HE operations in Table 3, including homomorphic addition, plaintext-ciphertext multiplication (PCMult), homomorphic multiplication (Mult), homomorphic rotations (row-wise rotation Rot_r, column-wise rotation Rot_c), and NTT/iNTT.

Table 2: Parameters for the SHE and FHE benchmarks. The two SHE sets are used for CCMM (Table 5): the parameters of [GL26, Table 1] and their $p = 27$ counterpart over our ring, with $N = 2n\varphi(p)$ in both cases. Here, N denotes the ring degree, $(h, \tilde{h})$ are the Hamming-weight parameters for the original secret s and sparse secret s', respectively [BTPH22], and dnum is the number of RNS decomposition digits used in key switching. The columns $\log PQ$, $\log q_i$, and $\log p_i$ give the bit-lengths of the maximum combined modulus PQ, ciphertext-modulus primes, and auxiliary primes, respectively.

Parameter Set	N	$\log PQ$	$(h, \tilde{h})$	$\log q_i$					$\log p_i$	dnum
				Base	Mult	StC	EvalMod	CtS		
SHE-n256p17	8192	200	$(4096, –)$	60	40×2	–	–	–	60	3
SHE(10,2)	9216	200	$(4608, –)$	60	40×2	–	–	–	60	3
FHE(15)-S	32768	768	$(192, 32)$	33	$\{50, 25\}$	60	50×8	49×2	51×2	7
FHE(15)-D	32768	880	$(16384, 32)$	40	31×4	60	55×8	52×2	56×2	8
FHE(16)	65536	1767	$(32768, 32)$	60	40×13	39×3	60×8	56×4	61×6	5
FHE(8,4)-D	20736	534	$(1024, 32)$	33	30	32×2	37×8	33×2	45	14
FHE(10,3)-D	27648	714	$(1024, 32)$	40	32×3	38×2	40×8	40×2	51×2	8
FHE(9,4)-S	41472	1001	$(192, 32)$	40	31×2	39×3	45×8	43×4	50×5	4
FHE(9,4)-D	41472	1063	$(20736, 32)$	40	31×4	39×3	45×8	43×4	50×5	4
FHE(8,5)	62208	1658	$(31104, 32)$	60	40×11	48×3	56×8	50×4	61×6	5

Table 3: Core HE-operation timings (ms), averaged over 100 runs. For standard CKKS, its sole rotation direction is reported in the Rot_r column.

Parameter Set	Add	PCMult	Mult	Rot_r	Rot_c	NTT	iNTT
FHE(15)-S	0.144	0.246	9.08	8.57	-	0.286	0.293
FHE(9,4)-S	0.161	0.307	16.34	15.70	15.75	0.450	0.491
FHE(15)-D	0.239	0.427	15.76	14.91	-	0.286	0.293
FHE(9,4)-D	0.261	0.507	21.26	20.10	20.11	0.450	0.490
FHE(16)	1.415	2.395	113.16	103.15	-	0.605	0.621
FHE(8,5)	0.958	1.840	91.51	81.30	81.29	0.708	0.767

Noise analysis on coefficients and slots. We experimentally validate the variance predictions of Lemma 4 and Corollary 1 using encryptions of zero. Both experiments use $q_0 \approx 2^{60}$, a scale of $\Delta = 2^{40}$, secrets sampled from χ_h, and errors sampled from χ_σ with $\sigma = 3.2$. In the encryption experiment, we perform public-key encryption at modulus q_0 and measure the decryption error afterward. In the rescaling experiment, we generate ciphertexts at $Q = q_0 q_1$, where $q_1 \approx 2^{40}$, using direct public-key encryption without an auxiliary modulus. We then multiply by 10^{-4}, rescale to q_0, and measure the full output error. For these zero-mean errors, Table 4 estimates the coefficient variance by $\|e\|_2^2/N$ and the normalized slot variance by $2\|\mathsf{DFT}(e)\|_2^2/N^2$. The former averages squared error over N coefficients; the latter averages over $N/2$ complex slots and divides by N to compare variances across different ring degrees.

For direct public-key encryption at q_0, the error is $ue_{\mathrm{pk}} + e_0 + e_1 s$, where $u, s \leftarrow \chi_h$ and $e_{\mathrm{pk}}, e_0, e_1 \leftarrow \chi_\sigma$. The implemented sampler rounds Gaussian samples to integers, giving coefficient variance approximately $\sigma^2 + 1/12$. Applying Lemma 4 to both product terms predicts coefficient variances of approximately $(8h/3+1)(\sigma^2+1/12)$ in our ring and $(2h+1)(\sigma^2+1/12)$ in the power-of-two ring. In the rescaling experiment, multiplication by 10^{-4} suppresses the inherited encryption error, so the output error is dominated by $r_0 + r_1 s$.

Table 4: Coefficient and normalized slot error variances. Entries are means ± standard errors across 1,000 independent keys, using eight-ciphertext averages for each key. For our ring, h is the Hamming weight before cyclotomic reduction; for the power-of-two ring, it is the ordinary ternary Hamming weight.

Ring	N	h	$\|e\|_2^2/N$	$2\,\|\mathsf{DFT}(e)\|_2^2/N^2$
(a) Public-key encryption				
Our ring	41,472	192	5295.6 ± 1.2	3973.92 ± 0.71
Power-of-two	65,536	192	3974.49 ± 0.45	3974.49 ± 0.45
Our ring	20,736	1,024	28194.7 ± 7.6	21146.2 ± 5.3
Power-of-two	32,768	1,024	21152.6 ± 3.3	21152.6 ± 3.3
Our ring	27,648	1,024	28195.4 ± 6.7	21150.0 ± 4.7
Power-of-two	32,768	1,024	21154.3 ± 3.3	21154.3 ± 3.3
(b) Multiplication by 10^{-4} followed by rescaling				
Our ring	41,472	192	21.4222 ± 0.0078	16.0833 ± 0.0035
Power-of-two	65,536	192	16.0837 ± 0.0011	16.0837 ± 0.0011
Our ring	20,736	1,024	113.879 ± 0.038	85.407 ± 0.025
Power-of-two	32,768	1,024	85.4158 ± 0.0088	85.4158 ± 0.0088
Our ring	27,648	1,024	113.884 ± 0.034	85.416 ± 0.022
Power-of-two	32,768	1,024	85.4241 ± 0.0089	85.4241 ± 0.0089

Table 5: CCMM performance at $n = 256$ using the SHE parameter sets of Table 2: SHE-n256p17 for the baseline and SHE(10,2) for ours. Entries give latency (s) and amortized runtime (ms/matrix); speedups use the latter. Results are averaged over 100 runs.

Component	[GL26]	Ours	
	Lat. (Amort.)	Lat. (Amort.)	Speedup
MM	4.673 (292.0)	3.459 (192.1)	1.52×
KS	3.945 (246.6)	1.901 (105.6)	2.33×
Total	8.630 (539.4)	5.374 (298.6)	1.81×

Under the uniform rounding model, the coefficients of r_0 and r_1 are independent samples from $\mathcal{U}(-1/2, 1/2)$, each with variance $1/12$. The corresponding coefficient variances are $(4h+3)/36$ in our ring and $(h+1)/12$ in the power-of-two ring. Corollary 1 predicts equal normalized slot variances in both rings: approximately $(2h+1)(\sigma^2 + 1/12)$ for encryption and $(h+1)/12$ for rescaling. The measurements in Table 4 agree with these predictions.

Batched CCMM benchmark. We also benchmark batched ciphertext–ciphertext matrix multiplication over our mixed-radix cyclotomic ring against the Gentry–Lee baseline in Table 5. Faster key switching yields a 2.33× speedup in the key-switching component and contributes to a 1.81× improvement in overall throughput.

5.1 Bootstrapping for Mixed Cyclotomic Ring

We compare the bootstrapping performance of the standard CKKS scheme with that of our method. Since the ring degrees are not identical across the compared instances, we report both latency and throughput, and use throughput as the primary comparison metric. For FHE(9,4)-S and FHE(9,4)-D, the comparisons with the corresponding power-of-two baselines preserve the number of multiplication primes available after bootstrapping (Table 2). The scaling factors and prime sizes of the remaining levels differ between

the two rings. In addition, under the model in Section 3.2, the integer-lift inputs have approximately 4/3 times the variance in our ring. As CtS places these coefficient-domain values into slots for EvalMod, this broader input distribution can increase the error in the polynomial approximation of the sine function during EvalMod. Therefore, the reduced precision reported in Table 6 stems from both the parameter choices and the different input distributions.

EvalMod **configurations**. We fix $\tilde{h} = 32$ for all parameter sets and evaluate two EvalMod ranges for our ring: $K = 16$, as used in sparse-secret encapsulation over power-of-two rings [BTPH22], and $K = 25$, as certified by Theorem 3. Table 6 shows the resulting trade-off between runtime and the certified failure bound. Increasing K from 16 to 25 improves this bound on our ring from 2^{-43} to 2^{-140}, comparable to that of the power-of-two configuration. Both configurations use eight levels for EvalMod; the larger range reallocates one level from the double-angle iterations to the polynomial approximation, increasing EvalMod latency by 19–20% and total bootstrapping latency by approximately 6%.

Experimental Results. With $K = 16$, FHE(9,4)-S and FHE(9,4)-D achieve 24–29% lower CtS latency than the corresponding power-of-two baselines, despite their approximately 27% larger ring degree and slot count (Table 6). The flexibility in ring degree allows these sets to support a larger modulus budget at the same 128-bit security target. We allocate part of this additional budget to a larger auxiliary modulus P, reducing the number of decomposition digits from dnum $= 7/8$ to 4. Specifically, $\log P$ increases from 102/112 to 250, while $\log Q$ increases from 666/768 to 751/813 (Table 2).

Although the basic HE operations are generally slower for these mixed-radix parameter sets (Table 3), the overall CtS latency is lower. This suggests that the gains arise from the combined effect of the homomorphic mixed-radix FFT of Section 3.3 and the parameter choices above, including the smaller decomposition rank enabled by the larger auxiliary modulus. Overall bootstrapping throughput is 21–26% lower than that of the corresponding baselines.

With $K = 25$, our ring provides a certified failure bound comparable to that of the power-of-two baselines. For both FHE(9,4)-S and FHE(9,4)-D, total latency remains within 6% of the corresponding baseline, while throughput is 16.5–21.1% lower.

In contrast, FHE(8,5) uses an approximately 5% smaller ring degree than FHE(16) and reduces the number of post-bootstrapping multiplication primes from thirteen to eleven, while retaining six 61-bit auxiliary primes and dnum $= 5$. With $K = 16$, total latency is slightly lower (17.29 versus 17.49 s), but throughput is approximately 4% higher because of the smaller slot count. Increasing the range to $K = 25$ raises total latency by approximately 5.9% to 18.30 s, with throughput approximately 10% higher than that of FHE(16).

5.2 Bootstrapping for Gentry–Lee

We compare our batch bootstrapping method (Section 4) with a baseline that applies our mixed-radix bootstrapping method (Section 3.3) independently to each of the n_1 ciphertexts. Table 7 also includes Crockett et al. [CGK+26], whose comparison with our construction is discussed in Section 5.3. For all three methods, we measure the time to process the same batch of n_1 ciphertexts in separate single-thread and 16-threads runs.

For a fair comparison, we reimplement Crockett et al.'s method [CGK+26] in our Lattigo-based code base and use identical parameters for both batch methods. Both apply the boundary and modulus-raising optimizations of Section 4.3, allowing batch-case evaluation to use fewer multiplication primes than the per-ciphertext baseline; our method additionally uses branch fusion and hoisting. All three methods perform EvalMod over the same base ring $\mathcal{R}$ with $(\tilde{h}, K) = (32, 16)$.

Comparisons with baseline in Section 3.3. For bootstrapping in the Gentry–Lee scheme, we adapt the baseline's modulus chain to the two-level CtS and two-level StC

Table 6: Bootstrapping performance comparison with standard CKKS. Each parameter block lists the power-of-two baseline, followed by our ring with the same EvalMod configuration ($K = 16$) and with the range selected according to Theorem 3 ($K = 25$). The depth split $a + b$ denotes the levels used for polynomial approximation and double-angle iterations, respectively, totaling eight levels in every case. Reported failure probabilities are theoretical upper bounds based on Section 3.2 with $\tilde{h} = 32$.

Parameter Set	EvalMod			#slots	Latency (s)				Throughput	Prec. (bits)
	K	depth	fail. prob.	$n = \frac{N}{2}$	CtS	EvalMod	StC	Total	(ms/slot)	worst / avg.
FHE(15)-S	16	$5+3$	$2^{-139.7}$	16384	3.105	1.234	0.647	5.112	0.3120	12.4 / 15.2
FHE(9,4)-S	16	$5+3$	$2^{-43.4}$	20736	2.351	1.636	0.829	5.090	0.2455	12.3 / 14.9
FHE(9,4)-S	25	$6+2$	$2^{-140.5}$	20736	2.352	1.946	0.831	5.403	0.2606	12.0 / 14.8
FHE(15)-D	16	$5+3$	$2^{-139.7}$	16384	3.802	1.604	0.995	6.556	0.4001	14.6 / 16.7
FHE(9,4)-D	16	$5+3$	$2^{-43.4}$	20736	2.688	1.968	1.181	6.159	0.2970	12.2 / 14.5
FHE(9,4)-D	25	$6+2$	$2^{-140.5}$	20736	2.680	2.364	1.178	6.544	0.3156	11.7 / 13.9
FHE(16)	16	$5+3$	$2^{-138.7}$	32768	7.273	5.802	3.604	17.488	0.5337	21.1 / 23.1
FHE(8,5)	16	$5+3$	$2^{-42.9}$	31104	7.271	5.356	3.859	17.285	0.5557	17.3 / 19.3
FHE(8,5)	25	$6+2$	$2^{-140.1}$	31104	7.226	6.443	3.833	18.301	0.5884	16.2 / 18.2

Table 7: Comparison of our bootstrapping and Crockett et al.'s [CGK+26] for the Gentry–Lee with n_1-batched bootstrapping with baseline in Section 3.3. Times are in seconds; parentheses give BTS latency per ciphertext. MM denotes modular matrix multiplication and associated NTT time, while KS includes remaining key-switching operations. For the baseline, KS reports the full stage latency. Key and Peak denote evaluation-key storage and peak process RSS, respectively, in GB. Peak RSS includes key generation, preprocessing, and evaluation. Prec. reports worst / average precision in bits. Key sizes and precision values are shared across thread counts.

Params	Method	Threads	CtS		EvalMod	StC		BTS	Size (GB)		Prec.
			MM	KS	(s)	MM	KS	(s)	Key	Peak	(bits)
FHE(8,4)-D	Sec. 3.3	1	–	153.28	65.94	–	22.94	247.65 (3.870)	6.27	20.9	8.0 / 8.6
		16	–	11.37	4.64	–	1.79	18.22 (0.285)		21.1	
	[CGK+26]	1	19.49	35.93	49.56	5.29	6.30	116.56 (1.821)	8.16	26.7	9.0 / 9.4
		16	2.31	3.63	4.87	0.64	0.66	12.12 (0.189)		29.7	
	Sec. 4	1	7.97	21.66	49.54	1.24	3.24	83.65 (1.307)	8.19	28.0	9.2 / 9.5
		16	0.89	2.08	4.47	0.15	0.36	7.95 (0.124)		29.5	
FHE(10,3)-D	Sec. 3.3	1	–	782.59	325.90	–	201.71	1342.66 (5.245)	17.26	45.5	10.6 / 11.0
		16	–	63.24	24.18	–	15.52	105.26 (0.411)		46.1	
	[CGK+26]	1	63.65	102.70	312.38	22.21	27.37	528.40 (2.064)	14.23	57.1	10.6 / 11.1
		16	5.04	7.57	23.41	1.81	2.22	40.01 (0.156)		62.6	
	Sec. 4	1	52.91	94.53	316.07	13.60	22.02	499.18 (1.950)	14.27	70.1	10.7 / 11.1
		16	4.82	8.46	23.54	1.19	2.02	40.04 (0.156)		73.6	
FHE(9,4)-S	Sec. 3.3	1	–	305.28	211.56	–	106.87	661.39 (5.167)	8.91	15.7	12.1 / 12.6
		16	–	24.68	16.44	–	8.49	52.23 (0.408)		26.5	
	[CGK+26]	1	57.56	72.63	205.25	18.62	20.76	374.81 (2.928)	12.31	62.5	14.6 / 14.9
		16	4.73	5.71	15.56	1.58	1.72	29.24 (0.228)		70.2	
	Sec. 4	1	40.71	73.07	208.05	8.36	17.34	347.38 (2.714)	12.35	70.7	14.7 / 14.9
		16	3.44	5.91	15.82	0.77	1.44	27.37 (0.214)		74.8	

circuits, while keeping the multiplication modulus after bootstrapping unchanged. By applying ModRaise to modulus pQ as described in Section 4.3, we save one CtS level, allowing us to reassign one CtS prime from Q to P for all three parameter sets. As a result, CtS consumes only one ciphertext level. For FHE(9,4)-S, we further remove two 43-bit

CtS primes and one 39-bit StC prime from the ciphertext modulus chain Q. Thus, $\log PQ$ remains 534 and 714 for FHE(8,4)-D and FHE(10,3)-D, respectively, and decreases from 1001 to 876 for FHE(9,4)-S.

With a single thread, ours is 1.90–2.96× faster than the baseline, primarily through reduced CtS and StC costs. The level budget also matters: on FHE(9,4)-S, our CtS and StC consume 1 and 2 levels, versus 4 and 3 for the baseline; on FHE(8,4)-D, our construction evaluates EvalMod on a smaller number of multiplication primes, reducing its time by about 1.33×. Moving from one to 16 threads yields speedups of 10.52–12.69× for our method and 12.66–13.59× for the baseline. At 16 threads, our method remains 1.91–2.63× faster than the baseline.

Bootstrapping precision. We measure the precision of bootstrapping by comparing the input message $\boldsymbol{m}$ with the decoded output $\tilde{\boldsymbol{m}}$ of each base-ring $\mathcal{R}^{(\mathrm{i})}$ ciphertext. Precision is defined as $-\log_2(\|\boldsymbol{m}-\tilde{\boldsymbol{m}}\|_\infty/\|\boldsymbol{m}\|_\infty)$ in bits, with input messages normalized to $\|\boldsymbol{m}\|_\infty = 1$. For each batch, we report the minimum and mean precision across its constituent ciphertexts, capturing the worst-case and average accuracy, respectively.

In both the Gentry–Lee bootstrapping of Crockett et al. [CGK+26] and our construction, the raised message and overflow amplify CtS matrix encoding errors. To improve precision, we choose the largest prime p among the moduli comprising P when performing ModRaise to pQ. The first product, evaluated modulo pQ and rescaled by p, then supports a larger encoding scale than the second product modulo Q in our parameter sets. We exploit this difference by balancing the two products:

$$\boldsymbol{D}_1^{-1}\boldsymbol{A}(\boldsymbol{D}_2^{-1})^T = (t^{-1}\boldsymbol{D}_1^{-1})\boldsymbol{A}(t\boldsymbol{D}_2^{-1})^T, \qquad t > 0.$$

In Algorithm 3, we implement this balancing by scaling the matrix corresponding to $d_1^{(-1)}$ by t^{-1} and the branch matrices corresponding to $d_2'^{(*)}$ and $d_2'^{(T)}$ by t before encoding. This reduces the second product's encoding-error contribution at the expense of amplifying earlier errors, while keeping the encoding scale factors fixed. We apply the same balancing to both constructions, using $t = 2^k$ with $k = 5, 3, 3$ for FHE(8,4)-D, FHE(10,3)-D, and FHE(9,4)-S, respectively. Both achieve precision comparable to or higher than the baseline on all parameter sets in Table 7, even using smaller primes in CtS and StC.

Key size and peak memory. For a sparse secret s', our construction uses four switching keys over the base ring $\mathcal{R}^{(\mathrm{i})}$: $\mathsf{swk}_{s\to s'}$, $\mathsf{swk}_{\bar{s}(U^{-1},W^{-1})\to s}$, $\mathsf{swk}_{s(U^{-1},W)\to s}$, and $\mathsf{swk}_{s^2\to s}$; two over $\mathcal{S}^{(\mathrm{i})}$: $\mathsf{swk}_{\bar{s}'(V^{-1},W^{-1})\to s}$ and $\mathsf{swk}_{\bar{s}(V^{-1},W^{-1})\to s}$; and two over $\mathcal{S}^{(\omega)}$: $\mathsf{swk}_{\bar{s}(U^{-1},Y^{-1})\to s}$ and $\mathsf{swk}_{s(U^{-1},Y)\to s}$. Our evaluation-key sizes are 8.19, 14.27, and 12.35 GB on FHE(8,4)-D, FHE(10,3)-D, and FHE(9,4)-S, respectively. Relative to the baseline in Section 3.3, these correspond to a 1.92 GB increase, a 2.99 GB decrease, and a 3.44 GB increase, respectively. Peak memory with single thread (16 threads) is 28.0 (29.5), 70.1 (73.6), and 70.7 (74.8) GB, exceeding the baseline by 7.1 (8.4), 24.6 (27.5), and 55.0 (48.3) GB, respectively. The construction of Crockett et al. [CGK+26] shows similar trends in key size and peak memory; detailed figures for all three methods are provided in Table 7.

5.3 Comparison with Crockett et al.

Using the common implementation described in Section 5.2, we compare the two constructions under identical parameters, focusing on packing granularity and W-axis matrix-multiplication costs. For FHE(8,4)-D, the finer granularity of our auxiliary-ring packing reduces padding for small batches. For FHE(9,4)-S, our lower matrix-multiplication cost improves performance even when both constructions require the same total padding.

5.3.1 Finer Packing Granularity

For the W-axis transform, one $\mathcal{S}^{(\omega)}$ ciphertext packs $n_2/2$ base-ring ciphertexts, whereas an auxiliary-ring ciphertext in Crockett et al. packs n_2. For FHE(8,4)-D, where $n_1 = 64$ and $n_2 = 162$, both constructions use one auxiliary ciphertext, but ours has capacity 81 rather than 162, requiring only 17 padded entries instead of 98. This finer granularity reduces auxiliary-ring computation for small batches. Together with the elimination of the correction term, it yields a full-bootstrapping speedup of 1.39× with one thread and 1.52× with 16 threads (Table 7).

5.3.2 Reduced W-Axis Matrix-Multiplication Cost

For FHE(9,4)-S, where $n_1 = 128$ and $n_2 = 162$, both constructions have total capacity 162 and require 34 padded entries. Crockett et al. uses one auxiliary ciphertext, whereas ours uses two $\mathcal{S}^{(\omega)}$ ciphertexts containing 81 and 47 inputs. The second block still requires full 81×81 matrix products, including its 34 zero-padded columns.

Let $C = n_1 n_2^3$ denote the classical MM cost of Crockett et al.'s main W-axis product. Its rank-$n_2/2$ correction adds a cost of $C/2$. In our construction, each product over $\mathcal{S}^{(\omega)}$ computes $2n_1$ matrix multiplications of size $n_2/2$, costing $2n_1(n_2/2)^3 = C/4$. Each of the two $\mathcal{S}^{(\omega)}$ ciphertexts requires two such products in CtS and one in StC after branch fusion. The resulting costs are

$$C_{W,\text{prior}}^{\mathsf{CtS}} = C_{W,\text{prior}}^{\mathsf{StC}} = \tfrac{3}{2}C, \qquad C_{W,\text{ours}}^{\mathsf{CtS}} = C, \qquad C_{W,\text{ours}}^{\mathsf{StC}} = \tfrac{1}{2}C.$$

Our W-axis MM cost is therefore two-thirds of Crockett et al.'s in CtS and one-third in StC. The measured MM times in Table 7 additionally include the common X-axis products and associated NTTs.

These savings come with a key-switching trade-off. For the W stage, Crockett et al.'s construction uses two key-switching operations over R^{aux} in each of CtS and StC, whereas ours uses six over $\mathcal{S}^{(\omega)}$ in CtS (which can be optimized as described in Section 4.3) and four in StC. Since $\mathcal{S}^{(\omega)}$ has half the dimension of R^{aux}, the dimension-weighted switching count is 1.5 times as large in CtS and unchanged in StC. With one thread, the CtS-MM time decreases from 57.56 to 40.71 s, while the CtS-KS time increases only slightly, from 72.63 to 73.07 s.

Overall, CtS throughput is 12.6% lower with one thread and 10.4% lower with 16 threads; the corresponding reductions for StC are 34.7% and 33.0%. Full bootstrapping is 1.08× and 1.07× faster, respectively, showing that the performance benefit persists with identical total padding.

A Proofs in Section 3

A.1 Proofs for Section 3.1

Proof of Lemma 2. Since the power-of-two case is well-known; we prove only the power-of-three case by induction on β. For $\beta = 1$, $2^{2\cdot 3^0} = 4 \equiv 1 + 3 \pmod{3^2}$.

Suppose $2^{2\cdot 3^{\beta-1}} \equiv 1+3^\beta \pmod{3^{\beta+1}}$ for some $\beta \geq 1$, and write $2^{2\cdot 3^{\beta-1}} = 1+3^\beta+3^{\beta+1}k$ for some $k \in \mathbb{Z}$. Then

$$\begin{aligned} 2^{2\cdot 3^\beta} &= \left(1 + 3^\beta + 3^{\beta+1}k\right)^3 \\ &= 1 + 3(3^\beta + 3^{\beta+1}k) + 3(3^\beta + 3^{\beta+1}k)^2 + (3^\beta + 3^{\beta+1}k)^3 \\ &\equiv 1 + 3^{\beta+1} \pmod{3^{\beta+2}}, \end{aligned}$$

which completes the induction. □

Proof of Lemma 3. By the Chinese Remainder Theorem,

$$\mathbb{Z}_{3N}^{\times} \cong \mathbb{Z}_{2^\alpha}^{\times} \times \mathbb{Z}_{3^{\beta+1}}^{\times}.$$

For $\alpha \geq 3$, it is well known that

$$\mathbb{Z}_{2^\alpha}^{\times} \cong \langle 5 \rangle \times \langle -1 \rangle \cong \mathcal{C}_{2^{\alpha-2}} \times \mathcal{C}_2.$$

For $\beta \geq 1$, Lemma 2 shows that 2 generates $\mathbb{Z}_{3^{\beta+1}}^{\times}$, so $4 = 2^2$ generates its unique odd-order subgroup of order 3^β. For $\beta = 0$, we simply have $4 \equiv 1 \pmod 3$. Hence

$$\mathbb{Z}_{3^{\beta+1}}^{\times} \cong \langle 4 \rangle \times \langle -1 \rangle \cong \mathcal{C}_{3^\beta} \times \mathcal{C}_2.$$

Under this identification, define

$$g_1 = (5, 1), \qquad g_2 = (-1, 4), \qquad g_3 = (-1, -1).$$

Then, we can check that the multiplicative orders of g_1, g_2, and g_3 are $2^{\alpha-2}, 2 \cdot 3^\beta$, and 2, respectively. Let $H := \langle g_2, g_3 \rangle$. Since

$$g_2^{3^\beta} = (-1, 1), \qquad g_2^{3^\beta} g_3 = (1, -1), \qquad g_2^{3^\beta+1} = (1, 4),$$

the subgroup H contains $\{\pm 1\} \times \langle 4, -1 \rangle = \{\pm 1\} \times \mathbb{Z}_{3^{\beta+1}}^{\times}$. Conversely, every element of H has first CRT component equal to ± 1, since both g_2 and g_3 do. Hence

$$H = \{\pm 1\} \times \mathbb{Z}_{3^{\beta+1}}^{\times}, \qquad |H| = 4 \cdot 3^\beta.$$

Now $\langle g_1 \rangle = \langle (5, 1) \rangle = \langle 5 \rangle \times \{1\}$. If $(5^t, 1) \in H$, then its first CRT component lies in $\langle 5 \rangle \cap \{\pm 1\}$. Since $\mathbb{Z}_{2^\alpha}^{\times} = \langle 5 \rangle \times \langle -1 \rangle$ is a direct product, this intersection is trivial. Therefore

$$\langle g_1 \rangle \cap H = \{1\}.$$

Consequently,

$$|\langle g_1, g_2, g_3 \rangle| = |\langle g_1 \rangle| \, |H| = 2^{\alpha-2} \cdot 4 \cdot 3^\beta = \varphi(3N) = |G|.$$

This proves that g_1, g_2, and g_3 generate $\mathbb{Z}_{3N}^{\times}$ with the stated orders. □

A.2 Proofs for Section 3.2

Proof of Lemma 4. The unreduced secret satisfies $\mathbb{E}[s_j] = 0$ and $\mathbb{E}[s_j s_\ell] = \frac{h}{3N} \delta_{j,\ell}$ for $j, \ell \in [3N]$, by the uniform support and independent symmetric signs. For $i \in [N]$, let $\rho_i(t)$ be the coefficient of X^i in $X^t \bmod \Phi_{3N}(X)$. The relations $X^{2n} = X^n - 1$ and $X^{3n} = -1$ make ρ_i $3N$-periodic. For $u \in [n]$, its nonzero entries lie in the residue class of u modulo n, with

$$(\rho_u(u + jn))_{j \in [6]} = (1, 0, -1, -1, 0, 1),$$
$$(\rho_{u+n}(u + jn))_{j \in [6]} = (0, 1, 1, 0, -1, -1).$$

Since $d_i = \sum_{k \in [N]} \sum_{j \in [3N]} c_k s_j \rho_i(j + k)$, independence and centering give $\mathbb{E}[d_i] = 0$ and

$$\begin{aligned} \mathrm{Var}(d_i) &= \frac{h\sigma^2}{3N} \sum_{k \in [N]} \sum_{j \in [3N]} \rho_i(j + k)^2 \\ &= \frac{h\sigma^2}{3} \sum_{t \in [3N]} \rho_i(t)^2 = \tfrac{4}{3} h\sigma^2. \end{aligned}$$

The second equality uses the $3N$-periodicity of ρ_i, and the last uses the four nonzero entries in each displayed pattern. □

Proof of Corollary 1. Reduction modulo Φ_{3N} preserves evaluation at ξ, so $s(\xi) = \widetilde{s}(\xi)$ and $d(\xi) = c(\xi)s(\xi)$. Both $c(\xi)$ and $s(\xi)$ have mean zero; for a centered complex variable Z, we use $\mathrm{Var}(Z) = \mathbb{E}|Z|^2$. Since $|\xi| = 1$, centering and independence of the coefficients of c give $\mathrm{Var}(c(\xi)) = N\sigma^2$. Similarly, the independent symmetric signs of the unreduced secret make its cross terms vanish, yielding

$$\mathrm{Var}(s(\xi)) = \sum_{j\in[3N]} \mathrm{Var}(s_j)|\xi^j|^2 = 3N \cdot \frac{h}{3N} = h.$$

By independence and centering of $c(\xi)$ and $s(\xi)$, we obtain

$$\frac{1}{N}\,\mathrm{Var}(d(\xi)) = \frac{1}{N}\,\mathrm{Var}(c(\xi))\,\mathrm{Var}(s(\xi)) = h\sigma^2,$$

which proves (4). □

Proof of Theorem 3

Let ρ_i be as in the proof of Lemma 4 and identify a secret with its unreduced coefficient vector $(s_\ell)_{\ell\in[3N]}$, which has h nonzero entries in $\{\pm 1\}$. The entries of the multiplication matrix of Theorem 3 are $m_{ik}(s) := (\boldsymbol{M}_s)_{i,k} = \sum_{\ell\in[3N]} s_\ell\, \rho_i(\ell + k)$.

Lemma 9 (Row structure)**.** *Fix $i \in [N]$ and a secret s.*

1. *$\rho_i(\ell + k) \neq 0$ only if $\ell + k \equiv i \pmod n$. Hence $m_{ik}(s)$ is a signed sum of the coefficients s_ℓ with $\ell \equiv i - k \pmod n$, at most four of which have $\rho_i(\ell + k) \neq 0$; in particular $|m_{ik}(s)| \leq 4$.*

2. *Each coefficient s_ℓ enters at most two entries of row i, so $\sum_{k\in[N]} |m_{ik}(s)| \leq 2h$.*

3. *For $r \in [n]$ let $C_r = \{r + \mu n : \mu \in [6]\}$ be a residue class of positions. The class C_r feeds exactly the two entries m_{ik} with $k \equiv i - r \pmod n$, say $k_0 \in [n]$ and $k_0 + n$, and distinct classes feed disjoint pairs of entries. With $v = (1, 0, -1, -1, 0, 1)$ and indices modulo 6, there are $\mu_0 \in [6]$ and $\sigma \in \{\pm 1\}$ such that the signs seen by $s_{r+\mu n}$ in the entries (i, k_0) and $(i, k_0 + n)$ are $\sigma v_{\mu+\mu_0}$ and $\sigma v_{\mu+\mu_0+1}$.*

Proof. The patterns displayed in the proof of Lemma 4 show that $\rho_i(t) \neq 0$ forces $t \equiv i \pmod n$, that $(\rho_i(i + \mu n))_{\mu\in[6]}$ equals v for $i < n$ and $(1, 1, 0, -1, -1, 0) = (-v_{\mu+2})_{\mu\in[6]}$ for $i \geq n$ (the second pattern of that proof is indexed from $i - n$), and that ρ_i is $6n$-periodic. For fixed i and ℓ the congruence $\ell + k \equiv i \pmod n$ has exactly two solutions $k \in [N]$, which gives (2) and the first claim of (3); for fixed i and k it selects one residue class of ℓ, whose six positions see the six entries of the pattern, four of which are nonzero, which gives (1). For (3), set $\eta = (r + k_0 - i)/n$. The two entries use the pattern at indices $\mu + \eta$ and $\mu + \eta + 1$. Thus take $(\sigma, \mu_0) = (1, \eta)$ for $i < n$ and $(-1, \eta + 2)$ for $i \geq n$, with indices modulo 6. □

Lemma 10 (Uniform decomposition)**.** *Let u be uniform on $(-1/2, 1/2)$ and m a nonzero integer. Then mu has the law of $u' + \delta$, where u' is uniform on $(-1/2, 1/2)$, δ is uniform on $\{-(|m| - 1)/2, -(|m| - 1)/2 + 1, \ldots, (|m| - 1)/2\}$, and u' and δ are independent.*

Proof. mu is uniform on $(-|m|/2, |m|/2)$, the union of the $|m|$ unit intervals centered at the points of the support of δ. Choosing one of these intervals uniformly and then a uniform point inside it yields the same law. □

Proof of Theorem 3. Fix s and i, write $u_0 = b/q_0$, $u_1 = a/q_0$, and $m_k = m_{ik}(s)$. Then $S_i = u_{0,i} + \sum_{k\in[N]} m_k u_{1,k}$, with all uniforms independent. Applying Lemma 10 to each term with $m_k \neq 0$, and using that $-u'$ has the law of u', gives

$$S_i \overset{d}{=} X + D, \qquad X = u_{0,i} + \sum_{k:\, m_k\neq 0} u'_k, \qquad D = \sum_{k:\, m_k\neq 0} \delta_k,$$

where X is a centered Irwin–Hall sum of order $1 + L_i(s)$, D is independent of X, and $|D| \le \sum_k (|m_k| - 1)_+/2 = c_i(s)/2$. The range condition fails at i when $|S_i| > K$. Conditioning on D and using that $|X + d| > t$ implies $|X| > t - |d|$,

$$\begin{aligned}\Pr\big[|S_i| > K \;\big|\; s\big] &= \sum_d \Pr[D = d]\, \Pr\big[|X + d| > K\big] \\ &\le \Pr\big[|X| > K - \tfrac{c_i(s)}{2}\big] = \bar{F}_{1+L_i(s)}\big(K - \tfrac{c_i(s)}{2}\big).\end{aligned}$$

When $c_i(s) = 0$ we have $D = 0$ and the inequality is an equality. Averaging over s, with $\pi_{L,c}$ independent of i by (10) below, gives $\Pr[|S_i| > K] \le \sum_{L,c} \pi_{L,c}\, \bar{F}_{1+L}(K - c/2)$, and a union bound over $i \in [N]$ gives (6). We also note that, by Lemma 9(2), $|S_i| \le \frac{1}{2} + \frac{1}{2}\sum_k |m_k| \le h + \frac{1}{2}$ for every secret, so the range condition cannot fail when $K \ge h + \frac{1}{2}$. □

The law of (L_i, c_i). By Lemma 9(3), row i decomposes over the n residue classes, each contributing to its own pair of entries. For one class, a set of nonzero positions $O \subseteq [6]$ with signs $\epsilon \in \{\pm 1\}^O$ yields the two entries $m^{(0)} = \sigma \sum_{\mu\in O} \epsilon_\mu v_{\mu+\mu_0}$ and $m^{(1)} = \sigma \sum_{\mu\in O} \epsilon_\mu v_{\mu+\mu_0+1}$; let $L(O,\epsilon) \in \{0,1,2\}$ count the nonzero ones and $c(O,\epsilon) = (|m^{(0)}| - 1)_+ + (|m^{(1)}| - 1)_+$. The class polynomial

$$P(x, y, z) = \sum_{O\subseteq[6]} \sum_{\epsilon\in\{\pm1\}^O} z^{|O|}\, x^{L(O,\epsilon)}\, y^{c(O,\epsilon)}$$

is invariant under a cyclic shift of μ_0 and under the sign σ, so it is the same for every row and class. Explicitly,

$$\begin{aligned}P = 1 &+ z\,(8x + 4x^2) \\ &+ z^2\,(6 + 16x + 4xy + 16x^2 + 16x^2y + 2x^2y^2) \\ &+ z^3\,(16 + 32x + 32xy + 24x^2 + 24x^2y + 24x^2y^2 + 8x^2y^3) \\ &+ z^4\,(12 + 48x + 20xy + 16xy^2 + 40x^2 + 48x^2y + 28x^2y^2 + 16x^2y^3 + 12x^2y^4) \\ &+ z^5\,(40x + 8xy^2 + 36x^2 + 48x^2y + 24x^2y^2 + 24x^2y^3 + 4x^2y^4 + 8x^2y^5) \\ &+ z^6\,(10 + 24xy + 4xy^3 + 16x^2y^2 + 8x^2y^4 + 2x^2y^6).\end{aligned}$$

The linear term records that a single nonzero coefficient feeds one entry from four of the six positions and two entries from the other two, which is the source of $\mathbb{E}\big[\sum_k m_{ik}(s)^2\big] = 4h/3$ in Lemma 4 (note $L_i \le \sum_k m_{ik}^2$, with equality exactly when $c_i = 0$); the quadratic term records that two coefficients of a class may cancel (6) or merge into one entry ($16x$), produce one entry ± 2 ($4xy + 16x^2y$), or produce two entries ± 2 ($2x^2y^2$). Sampling s chooses uniformly h of the $3N$ positions with independent signs, that is, a family $(O_r, \epsilon_r)_{r\in[n]}$ with $\sum_r |O_r| = h$, uniformly among all such families. By Lemma 9(3), distinct classes feed disjoint pairs of entries, so $L_i = \sum_r L(O_r, \epsilon_r)$ and $c_i = \sum_r c(O_r, \epsilon_r)$. Therefore

$$\pi_{L,c} = \frac{[z^h x^L y^c]\; P(x, y, z)^n}{\binom{3N}{h}\, 2^h}, \tag{10}$$

which does not depend on i. The numerator counts the families with $(L_i, c_i) = (L, c)$ and the denominator counts all of them.

A.3 Proofs for Section 3.3

Throughout, $\boldsymbol{S}_k$ is the reordered DFT core of size k at modulus $6k$, as defined in Section 3.3, and g_1, g_2 are reduced modulo $6k$. The proofs of the three lemmas follow the same pattern: a congruence for a power of g_1 or g_2, supplied by Lemma 2, is combined with the recursion $\mathsf{rev}_k(j + tk/d) = d\,\mathsf{rev}_{k/d}(j) + t$ for $t \in [d]$ and $j < k/d$, where $d = d_k$.

Proof of Lemma 5. Let $k = 2^u 3^v$ with $u > 1$, so that $6k = 2^{u+1}3^{v+1}$ and $d_k = 2$. By Lemma 2 applied with $\alpha = u$, $g_1^{2^{u-2}} \equiv 5^{2^{u-2}} \equiv 1 + 2^u \pmod{2^{u+1}}$, and $3k = 2^u 3^{v+1} \equiv 2^u \pmod{2^{u+1}}$ because 3^{v+1} is odd. Since $g_1 \equiv 1 \pmod{3^{v+1}}$, the Chinese Remainder Theorem gives

$$g_1^{2^{u-2}} \equiv 1 + 3k \pmod{6k}.$$

Write a row index of $\boldsymbol{S}_{k/2}$ as $i = (2 \cdot 3^v)i_1 + i_2$ with $i_1 \in [2^{u-2}]$ and $i_2 \in [2 \cdot 3^v]$; the row $i + k/2$ of $\boldsymbol{S}_k$ then carries the exponent $g_1^{i_1 + 2^{u-2}} g_2^{i_2}$. For $0 \le i, j < k/2$, the recursion of rev_k gives

$$\begin{aligned}
(\boldsymbol{S}_k)_{i,j} &= \omega_{6k}^{g_1^{i_1} g_2^{i_2} \cdot 2\mathsf{rev}_{k/2}(j)} = (\boldsymbol{S}_{k/2})_{i,j}, \\
(\boldsymbol{S}_k)_{i+k/2,j} &= \omega_{6k}^{g_1^{i_1} g_2^{i_2}\, g_1^{2^{u-2}} \cdot 2\mathsf{rev}_{k/2}(j)} = \omega_{3k}^{g_1^{i_1} g_2^{i_2} (1+3k)\mathsf{rev}_{k/2}(j)} = (\boldsymbol{S}_{k/2})_{i,j}, \\
(\boldsymbol{S}_k)_{i,j+k/2} &= \omega_{6k}^{g_1^{i_1} g_2^{i_2} (2\mathsf{rev}_{k/2}(j)+1)} = (\boldsymbol{S}_{k/2})_{i,j} \cdot \omega_{6k}^{g_1^{i_1} g_2^{i_2}}, \\
(\boldsymbol{S}_k)_{i+k/2,j+k/2} &= (\boldsymbol{S}_{k/2})_{i,j} \cdot \omega_{6k}^{(1+3k) g_1^{i_1} g_2^{i_2}} = -(\boldsymbol{S}_{k/2})_{i,j} \cdot \omega_{6k}^{g_1^{i_1} g_2^{i_2}},
\end{aligned}$$

where the last step uses that $g_1^{i_1} g_2^{i_2}$ is odd, so that $\omega_{6k}^{3k\, g_1^{i_1} g_2^{i_2}} = -1$. Reading off the four blocks gives $\boldsymbol{S}_k = \boldsymbol{F}_k^{(2)}(\boldsymbol{I}_2 \otimes \boldsymbol{S}_{k/2})$ with $(\boldsymbol{W}_{k/2})_{i,i} = \omega_{6k}^{g_1^{i_1} g_2^{i_2}}$. □

Proof of Lemma 6. Let $\ell = 3^v$, so that $\boldsymbol{S}_{2\ell}$ lives at modulus $12\ell = 4 \cdot 3^{v+1}$, its rows are indexed by $i \in [2\ell]$ with exponent g_2^i, and $d_{2\ell} = 2$. By Lemma 2 applied with $\beta = v + 1$, $4^{3^v} = 2^{2 \cdot 3^v} \equiv 1 \pmod{3^{v+1}}$, hence $g_2^\ell \equiv 1 \pmod{3^{v+1}}$; moreover $g_2^\ell \equiv (-1)^{3^v} = -1 \equiv 3 \pmod 4$. Since also $1 + 6\ell = 1 + 2 \cdot 3^{v+1} \equiv 3 \pmod 4$ and $\equiv 1 \pmod{3^{v+1}}$, the Chinese Remainder Theorem gives

$$g_2^\ell \equiv 1 + 6\ell \pmod{12\ell}.$$

For $0 \le i, j < \ell$, the recursion of $\mathsf{rev}_{2\ell}$ gives

$$\begin{aligned}
(\boldsymbol{S}_{2\ell})_{i,j} &= \omega_{12\ell}^{g_2^i \cdot 2\mathsf{rev}_\ell(j)} = (\boldsymbol{S}_\ell)_{i,j}, \qquad (\boldsymbol{S}_{2\ell})_{i+\ell,j} = \omega_{6\ell}^{g_2^i (1+6\ell)\mathsf{rev}_\ell(j)} = (\boldsymbol{S}_\ell)_{i,j}, \\
(\boldsymbol{S}_{2\ell})_{i,j+\ell} &= (\boldsymbol{S}_\ell)_{i,j} \cdot \omega_{12\ell}^{g_2^i}, \qquad (\boldsymbol{S}_{2\ell})_{i+\ell,j+\ell} = (\boldsymbol{S}_\ell)_{i,j} \cdot \omega_{12\ell}^{(1+6\ell) g_2^i} = -(\boldsymbol{S}_\ell)_{i,j} \cdot \omega_{12\ell}^{g_2^i},
\end{aligned}$$

using that g_2^i is odd in the last step. This is $\boldsymbol{S}_{2\ell} = \boldsymbol{F}_{2\ell}^{(\mathrm{tr})}(\boldsymbol{I}_2 \otimes \boldsymbol{S}_\ell)$ with $\boldsymbol{T}_\ell = \mathsf{diag}(\omega_{12\ell}^{g_2^i})_{i \in [\ell]}$. □

Proof of Lemma 7. Let $k = 3^v$ with $v > 0$, so that $6k = 2 \cdot 3^{v+1}$, the rows of $\boldsymbol{S}_k$ are indexed by $i \in [k]$ with exponent g_2^i, and $d_k = 3$. By Lemma 2 applied with $\beta = v$, $g_2^{k/3} \equiv 2^{2 \cdot 3^{v-1}} \equiv 1 + 3^v \pmod{3^{v+1}}$, and $g_2^{k/3}$ is odd; since $1 + 4k = 1 + 4 \cdot 3^v \equiv 1 + 3^v \pmod{3^{v+1}}$ is odd as well, we obtain

$$g_2^{k/3} \equiv 1 + 4k \pmod{6k}, \qquad g_2^{2k/3} \equiv (1 + 4k)^2 \equiv 1 + 2k \pmod{6k},$$

the latter because $16k^2 \equiv 0$ and $8k \equiv 2k \pmod{6k}$. Note that $g_2 \equiv 1 \pmod 3$, so $\omega_{6k}^{4k\, g_2^i} = \omega_3^{2g_2^i} = \omega^2$ and $\omega_{6k}^{2k\, g_2^i} = \omega$. For $0 \le i, j < k/3$ and $t \in \{1, 2\}$, the recursion

$\mathsf{rev}_k(j + tk/3) = 3\mathsf{rev}_{k/3}(j) + t$ gives

$$
\begin{aligned}
(\boldsymbol{S}_k)_{i+tk/3,\,j} &= \omega_{2k}^{g_2^i g_2^{tk/3}\mathsf{rev}_{k/3}(j)} = (\boldsymbol{S}_{k/3})_{i,j}, \qquad \text{since } g_2^{tk/3} \equiv 1 \pmod{2k},\\
(\boldsymbol{S}_k)_{i,\,j+tk/3} &= (\boldsymbol{S}_{k/3})_{i,j} \cdot \omega_{6k}^{t\,g_2^i},\\
(\boldsymbol{S}_k)_{i+k/3,\,j+tk/3} &= (\boldsymbol{S}_{k/3})_{i,j} \cdot \omega_{6k}^{t\,g_2^i(1+4k)} = (\boldsymbol{S}_{k/3})_{i,j} \cdot \omega_{6k}^{t\,g_2^i} \cdot \omega^{2t},\\
(\boldsymbol{S}_k)_{i+2k/3,\,j+tk/3} &= (\boldsymbol{S}_{k/3})_{i,j} \cdot \omega_{6k}^{t\,g_2^i(1+2k)} = (\boldsymbol{S}_{k/3})_{i,j} \cdot \omega_{6k}^{t\,g_2^i} \cdot \omega^{t}.
\end{aligned}
$$

With $\boldsymbol{A}_{k/3} = \mathsf{diag}(\omega_{6k}^{g_2^i})_{i\in[k/3]}$ and $\omega^4 = \omega$, the nine blocks are exactly those of $\boldsymbol{F}_k^{(3)}$, which proves $\boldsymbol{S}_k = \boldsymbol{F}_k^{(3)}(\boldsymbol{I}_3 \otimes \boldsymbol{S}_{k/3})$. □

Proof of Proposition 1. For $k = 2^u 3^v > 1$, let $\boldsymbol{F}_k$ denote the butterfly matrix of the lemma that applies to $\boldsymbol{S}_k$: $\boldsymbol{F}_k^{(2)}$ if $u > 1$ (Lemma 5), $\boldsymbol{F}_k^{(\mathrm{tr})}$ if $u = 1$ (Lemma 6), and $\boldsymbol{F}_k^{(3)}$ if $u = 0$ (Lemma 7). In all three cases,

$$\boldsymbol{S}_k = \boldsymbol{F}_k\,(\boldsymbol{I}_{d_k} \otimes \boldsymbol{S}_{k/d_k}). \tag{11}$$

For every $a \geq 1$, the mixed-product property $(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{AC}) \otimes (\boldsymbol{BD})$ and $\boldsymbol{I}_a \otimes \boldsymbol{I}_d = \boldsymbol{I}_{ad}$ turn (11) into

$$\boldsymbol{I}_a \otimes \boldsymbol{S}_k = (\boldsymbol{I}_a \otimes \boldsymbol{F}_k)\,(\boldsymbol{I}_{ad_k} \otimes \boldsymbol{S}_{k/d_k}).$$

Starting from $\boldsymbol{S} = \boldsymbol{I}_1 \otimes \boldsymbol{S}_n$, we apply this identity with $a = n/k$ for $k = n, n/2, \ldots, 4 \cdot 3^\beta$ (the $\alpha - 2$ radix-2 steps), then $k = 2 \cdot 3^\beta$ (the transition), and finally $k = 3^\beta, 3^{\beta-1}, \ldots, 3$ (the β radix-3 steps). This gives

$$\boldsymbol{S} = (\boldsymbol{I}_{n/n} \otimes \boldsymbol{F}_n)\,(\boldsymbol{I}_{n/(n/2)} \otimes \boldsymbol{F}_{n/2}) \cdots (\boldsymbol{I}_{n/3} \otimes \boldsymbol{F}_3)\,(\boldsymbol{I}_n \otimes \boldsymbol{S}_1).$$

Since $\boldsymbol{S}_1 = [1]$, the last factor is $\boldsymbol{I}_n$, and with $\widetilde{\boldsymbol{F}}_k = \boldsymbol{I}_{n/k} \otimes \boldsymbol{F}_k$ this is the first identity of (8). For $\beta = 0$, there are no radix-3 steps, and the transition step reaches $\boldsymbol{S}_1$ directly.

For the inverse, let

$$\boldsymbol{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \boldsymbol{\Omega}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega^2 & \omega \\ 1 & \omega & \omega^2 \end{bmatrix},$$

so that $\boldsymbol{H}_2\boldsymbol{H}_2^* = 2\boldsymbol{I}_2$ and, since $1 + \omega + \omega^2 = 0$, $\boldsymbol{\Omega}_3\boldsymbol{\Omega}_3^* = 3\boldsymbol{I}_3$. The butterflies factor as

$$
\begin{aligned}
\boldsymbol{F}_k^{(2)} &= (\boldsymbol{H}_2 \otimes \boldsymbol{I}_{k/2}) \cdot \mathsf{diag}(\boldsymbol{I}_{k/2}, \boldsymbol{W}_{k/2}),\\
\boldsymbol{F}_{2\ell}^{(\mathrm{tr})} &= (\boldsymbol{H}_2 \otimes \boldsymbol{I}_\ell) \cdot \mathsf{diag}(\boldsymbol{I}_\ell, \boldsymbol{T}_\ell),\\
\boldsymbol{F}_k^{(3)} &= (\boldsymbol{\Omega}_3 \otimes \boldsymbol{I}_{k/3}) \cdot \mathsf{diag}(\boldsymbol{I}_{k/3}, \boldsymbol{A}_{k/3}, \boldsymbol{A}_{k/3}^2),
\end{aligned}
$$

where $\mathsf{diag}(\cdot)$ denotes a block-diagonal matrix. The diagonal matrices $\boldsymbol{W}_{k/2}$, $\boldsymbol{T}_\ell$, and $\boldsymbol{A}_{k/3}$ have entries of modulus one, so the block-diagonal factors are unitary. Hence $\boldsymbol{F}\boldsymbol{F}^* = d\,\boldsymbol{I}$ for each butterfly of radix d, with $d = 2$ for the radix-2 and transition factors and $d = 3$ for the radix-3 factors. The same holds for the lifts $\widetilde{\boldsymbol{F}} = \boldsymbol{I} \otimes \boldsymbol{F}$, so $\widetilde{\boldsymbol{F}}^{-1} = d^{-1}\widetilde{\boldsymbol{F}}^*$, whose nonzero pattern is the transpose of that of $\widetilde{\boldsymbol{F}}$. Inverting the first identity of (8) reverses the order of the factors, and the scalars multiply to $2^{-(\alpha-2)} \cdot 2^{-1} \cdot 3^{-\beta} = 2^{-(\alpha-1)}3^{-\beta} = n^{-1}$. This gives the second identity of (8). □

Proof of Lemma 8. Cyclic offsets add under matrix multiplication: $\Delta(\boldsymbol{AB}) \subseteq \Delta(\boldsymbol{A}) + \Delta(\boldsymbol{B})$, since each nonzero contribution passes through an intermediate index. Offsets are taken modulo the corresponding axis size.

For $\boldsymbol{R}$, let s be the smallest row stride. The consecutive radix-2 factors have strides $s, 2s, \ldots, 2^{k_1-1}s$, each contributing zero or either signed stride. Hence

$$\Delta_{\mathrm{row}}(\boldsymbol{R}) \subseteq s\{-(2^{k_1}-1), \ldots, 2^{k_1}-1\} \pmod{n_1},$$

which is a cyclic arithmetic progression of length $2^{k_1+1}-1$ containing the row support.

If $\boldsymbol{C}$ excludes the transition factor, let s be its smallest column stride. Its radix-3 factors have strides $s, 3s, \ldots, 3^{k_2-1}s$ and contribute digits in $\{-2,-1,0,1,2\}$. Since $2\sum_{j=0}^{k_2-1} 3^j = 3^{k_2}-1$, their support lies in a cyclic arithmetic progression of length $2\cdot 3^{k_2}-1$. If $\boldsymbol{C}$ includes the transition factor, consecutiveness gives $s = 3^{\beta-k_2}$, and every factor offset is a multiple of s, including the transition offset $n_2/2$. Therefore

$$\Delta_{\mathrm{col}}(\boldsymbol{C}) \subseteq s\{0, \ldots, 2\cdot 3^{k_2}-1\} \pmod{n_2},$$

because $n_2/s = 2\cdot 3^{k_2}$. This also covers the transition-only case $k_2 = 0$.

If a displayed progression repeats an offset, replace it by one full cycle of its stride. For an axis of size d and stride s, this cycle has $d/\gcd(d,s)$ distinct offsets and is shorter than the original progression. Thus the containing progressions can always be chosen with distinct offsets and the stated length bounds.

Finally, the butterfly identities in Proposition 1 imply that each group inverse is a nonzero scalar multiple of its adjoint. Taking the adjoint negates the offsets, preserving the progression lengths and cardinality bounds. □

B Proofs in Section 4

B.1 Proof of Theorem 4

Proof. For $i \in [n_1]$ and $j \in [n_2/2]$, we have $(\boldsymbol{E}_1)_{i,j} = m(\zeta_i, \eta_j)$ and $(\boldsymbol{E}_2)_{i,j} = m(\zeta_i, \eta_{j+n_2/2})$. Since $\eta_{j+n_2/2} = \overline{\eta_j}$ and $\zeta_{i+n_1} = \overline{\zeta_i}$, it follows that

$$m(\zeta_i, \eta_{j+n_2/2}) = m(\overline{\zeta_{i+n_1}}, \overline{\eta_j}) = \overline{m(\zeta_{i+n_1}, \eta_j)}.$$

By definition of $\psi^{(\omega)}$, we have $\psi^{(\omega)}(\widetilde{m})_{j,i} = m(\zeta_i, \eta_j) = (\boldsymbol{E}_1)_{i,j}$ and

$$\psi^{(\omega)}(\widetilde{m})_{j,i+n_1} = m(\zeta_{i+n_1}, \eta_j) = \overline{m(\zeta_i, \eta_{j+n_2/2})} = \overline{(\boldsymbol{E}_2)_{i,j}}.$$

Therefore $\psi^{(\omega)}(\widetilde{m}) = \left[\boldsymbol{E}_1^T \mid \overline{\boldsymbol{E}_2}^T\right]$. □

B.2 Proof of Propositions 2 and 3

Lemma 11. *Let $a, b \in \mathbb{Z}[\mathrm{i}][V]/(V^{n_1}-\mathrm{i})$. Then $a = b$ if and only if $a(\zeta_j) = b(\zeta_j)$ for all $j \in [n_1]$. Analogously, for $\alpha, \beta \in \mathbb{Z}[\omega][Y]/(Y^{n_2/2}-\omega)$, we have $\alpha = \beta$ if and only if $\alpha(\eta_\ell) = \beta(\eta_\ell)$ for all $\ell \in [n_2/2]$.*

Proof. The equivalences follow from the fact that $\{\zeta_j\}_{j\in[n_1]}$ and $\{\eta_\ell\}_{\ell\in[n_2/2]}$ are precisely the sets of all the roots of $V^{n_1}-\mathrm{i}=0$ and $Y^{n_2/2}-\omega=0$, respectively. □

Proof of Proposition 2. From Theorem 2, we have

$$\Psi_{3,k}^{(\mathrm{i})}(m') = \frac{1}{n_1}\,\Psi_{3,k}^{(\mathrm{i})}(d_1)\left(\Psi_{3,k}^{(\mathrm{i})}(m^*)\right)^*$$
$$= \frac{1}{n_1}\,\Psi_{3,k}^{(\mathrm{i})}(d_1)\,\Psi_{3,k}^{(\mathrm{i})}(m) = \boldsymbol{D}_1 \cdot \left(m(\zeta_i, \zeta_j, \eta_k)\right)_{i,j\in[n_1]}$$

for any index $k \in [n_2]$, which implies

$$m'(\zeta_i, \zeta_j, \eta_k) = \sum_{\ell} (\boldsymbol{D}_1)_{i,\ell} \cdot m(\zeta_\ell, \zeta_j, \eta_k)$$

for any indices i, j, k. It follows that

$$\begin{aligned}\sum_j m'_j(\zeta_i, \eta_k)\zeta_\nu^j = m'(\zeta_i, \zeta_\nu, \eta_k) &= \sum_{\ell} (\boldsymbol{D}_1)_{i,\ell} \cdot m(\zeta_\ell, \zeta_\nu, \eta_k) \\ &= \sum_{\ell} (\boldsymbol{D}_1)_{i,\ell} \sum_j m_j(\zeta_\ell, \eta_k)\zeta_\nu^j \\ &= \sum_j \left(\sum_{\ell} (\boldsymbol{D}_1)_{i,\ell} \cdot m_j(\zeta_\ell, \eta_k) \right) \zeta_\nu^j\end{aligned}$$

for all $\nu \in [n_1]$. By Lemma 11, we have

$$\sum_j m'_j(\zeta_i, \eta_k) V^j = \sum_j \left(\sum_{\ell} (\boldsymbol{D}_1)_{i,\ell} \cdot m_j(\zeta_\ell, \eta_k) \right) V^j.$$

Therefore, we obtain

$$m'_j(\zeta_i, \eta_k) = \sum_{\ell} (\boldsymbol{D}_1)_{i,\ell} \cdot m_j(\zeta_\ell, \eta_k),$$

for any indices i, j, k, which proves our claim. □

Proof of Proposition 3. By Theorem 2, one has

$$m'(\eta_i, \eta_j, \zeta_k) = \sum_{\ell} (\boldsymbol{D}'_2)_{i,\ell} \cdot m(\eta_\ell, \eta_j, \zeta_k).$$

for any indices i, j, k. It follows that

$$\begin{aligned}\sum_j m'_j(\eta_i, \zeta_k) \cdot \eta_\nu^j = m'(\eta_i, \eta_\nu, \zeta_k) &= \sum_{\ell} (\boldsymbol{D}'_2)_{i,\ell} \cdot m(\eta_\ell, \eta_\nu, \zeta_k) \\ &= \sum_{\ell} (\boldsymbol{D}'_2)_{i,\ell} \sum_j m_j(\eta_\ell, \zeta_k)\eta_\nu^j \\ &= \sum_j \left(\sum_{\ell} (\boldsymbol{D}'_2)_{i,\ell} \cdot m_j(\eta_\ell, \zeta_k) \right) \eta_\nu^j\end{aligned}$$

for all $\nu \in [n_2/2]$. By Lemma 11, we have

$$\sum_j m'_j(\eta_i, \zeta_k) Y^j = \sum_j \left(\sum_{\ell} (\boldsymbol{D}'_2)_{i,\ell} \cdot m_j(\eta_\ell, \zeta_k) \right) Y^j.$$

Therefore, we obtain

$$m'_j(\eta_i, \zeta_k) = \sum_{\ell} (\boldsymbol{D}'_2)_{i,\ell} \cdot m_j(\eta_\ell, \zeta_k),$$

for any indices i, j, k, which proves our claim. □

B.3 Proof of Theorem 6

Proof. Write

$$\psi^{(\mathrm{i})}(m) = \begin{bmatrix} \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix}.$$

Then one checks that

$$\psi^{(\mathrm{i})}(m') = \begin{bmatrix} \boldsymbol{E}_2 & \boldsymbol{E}_1 \end{bmatrix}, \qquad \psi^{(\omega)}(\mathsf{pt}'_{\mathrm{stc}}) = \begin{bmatrix} \omega \mathbf{1}_{\frac{n_2}{2} \times n_1} & \mathbf{1}_{\frac{n_2}{2} \times n_1} \end{bmatrix},$$
$$\psi^{(\omega)}(\widetilde{m}) = \begin{bmatrix} \boldsymbol{E}_1^T & \overline{\boldsymbol{E}}_2^T \end{bmatrix}, \qquad \psi^{(\omega)}(\widetilde{m}') = \begin{bmatrix} \boldsymbol{E}_2^T & \overline{\boldsymbol{E}}_1^T \end{bmatrix}.$$

Let m_{out} denote the output obtained by right-multiplication by $\boldsymbol{D}_2^T$. Then

$$\psi^{(\mathrm{i})}(m)\boldsymbol{D}_2^T = \begin{bmatrix} \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_2'^T & \overline{\boldsymbol{D}}_2'^T \\ \omega \boldsymbol{D}_2'^T & \omega^2 \overline{\boldsymbol{D}}_2'^T \end{bmatrix}$$
$$= \begin{bmatrix} \boldsymbol{E}_1 \boldsymbol{D}_2'^T & \omega^2 \boldsymbol{E}_2 \overline{\boldsymbol{D}}_2'^T \end{bmatrix} + \begin{bmatrix} \omega \boldsymbol{E}_2 \boldsymbol{D}_2'^T & \boldsymbol{E}_1 \overline{\boldsymbol{D}}_2'^T \end{bmatrix} = \psi^{(\mathrm{i})}(m_{\mathrm{out}}).$$

Now let $\widetilde{m}_{\mathrm{out}}(W, U) = m_{\mathrm{out}}(U, W) \in \mathcal{R}^{(\omega)}$. Then

$$\psi^{(\omega)}(\widetilde{m}_{\mathrm{out}}) = \boldsymbol{D}'_2 \begin{bmatrix} \boldsymbol{E}_1^T & \omega \overline{\boldsymbol{E}}_2^T \end{bmatrix} + \boldsymbol{D}'_2 \begin{bmatrix} \omega \boldsymbol{E}_2^T & \overline{\boldsymbol{E}}_1^T \end{bmatrix}$$
$$= \boldsymbol{D}'_2 \left(\psi^{(\omega)}(\mathsf{pt}_{\mathrm{stc}}) \odot \psi^{(\omega)}(\widetilde{m}) \right) + \boldsymbol{D}'_2 \left(\psi^{(\omega)}(\mathsf{pt}'_{\mathrm{stc}}) \odot \psi^{(\omega)}(\widetilde{m}') \right)$$
$$= \boldsymbol{D}'_2 \, \psi^{(\omega)}(\mathsf{pt}_{\mathrm{stc}} \cdot \widetilde{m}) + \boldsymbol{D}'_2 \, \psi^{(\omega)}(\mathsf{pt}'_{\mathrm{stc}} \cdot \widetilde{m}'),$$

which proves the claim. Here, $\odot$ denotes component-wise multiplication.

For its inverse,

$$\psi^{(\mathrm{i})}(m)(\boldsymbol{D}_2^T)^{-1}$$
$$= \frac{2}{n_2(\omega^2 - \omega)} \left(\begin{bmatrix} \omega^2 \boldsymbol{E}_1 \overline{\boldsymbol{D}}'_2 & \boldsymbol{E}_2 \boldsymbol{D}'_2 \end{bmatrix} + \begin{bmatrix} -\omega \boldsymbol{E}_2 \boldsymbol{D}'_2 & -\boldsymbol{E}_1 \overline{\boldsymbol{D}}'_2 \end{bmatrix} \right).$$

Let m_{out} denote the corresponding output. Since

$$\psi^{(\omega)}(\mathsf{pt}'_{\mathrm{cts}}) = \frac{2}{n_2(\omega^2 - \omega)} \begin{bmatrix} \omega \mathbf{1}_{\frac{n_2}{2} \times n_1} & -\mathbf{1}_{\frac{n_2}{2} \times n_1} \end{bmatrix},$$

we obtain

$$\psi^{(\omega)}(\widetilde{m}_{\mathrm{out}}) = (\boldsymbol{D}'_2)^* \begin{bmatrix} \frac{2\omega^2}{n_2(\omega^2-\omega)} \boldsymbol{E}_1^T & -\frac{2}{n_2(\omega^2-\omega)} \overline{\boldsymbol{E}}_2^T \end{bmatrix}$$
$$- (\boldsymbol{D}'_2)^T \begin{bmatrix} \frac{2\omega}{n_2(\omega^2-\omega)} \boldsymbol{E}_2^T & -\frac{2}{n_2(\omega^2-\omega)} \overline{\boldsymbol{E}}_1^T \end{bmatrix}$$
$$= (\boldsymbol{D}'_2)^* \, \psi^{(\omega)}(\mathsf{pt}_{\mathrm{cts}} \cdot \widetilde{m}) - (\boldsymbol{D}'_2)^T \, \psi^{(\omega)}(\mathsf{pt}'_{\mathrm{cts}} \cdot \widetilde{m}'),$$

which proves the claim.

□

References

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[BBB+22] Ahmad Al Badawi, Jack Bates, Flávio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, R. V. Saraswathy, Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. OpenFHE: Open-source fully homomorphic encryption library. In *WAHC*, 2022.

[BCH+24] Youngjin Bae, Jung Hee Cheon, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé. Plaintext-ciphertext matrix multiplication and FHE bootstrapping: Fast and fused. In *Advances in Cryptology – CRYPTO*, 2024.

[bla25] *OpenBLAS: An optimized BLAS library*, 2025. Version 0.3.30, available at `https://www.openblas.net/`.

[BMTPH21] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In *Advances in Cryptology – EUROCRYPT*, 2021.

[BTPH22] Jean-Philippe Bossuat, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation. In *ACNS*, 2022.

[CCP+24] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the IND-CPA^D security of exact FHE schemes. In *ACM SIGSAC CCS*, 2024.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*, 2019.

[CGK+26] Eric Crockett, Craig Gentry, Hyojun Kim, Yeongmin Lee, and Yongwoo Lee. Efficient bootstrapping in fully homomorphic encryption for matrix arithmetic. In *Advances in Cryptology – CRYPTO*, 2026.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT*, 2017.

[CKL26] Jung Hee Cheon, Minsik Kang, and Junho Lee. Fast batch matrix multiplication in ciphertexts. In *Advances in Cryptology – CRYPTO*, 2026.

[Cry22] CryptoLab Inc. HEaaN: Homomorphic encryption library, 2022.

[DEdP25] Léo Ducas, Lynn Engelberts, and Paola de Perthuis. Predicting module-lattice reduction. In *Advnaces in Cryptology – ASIACRYPT*. Springer, 2025.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[GL26] Craig Gentry and Yongwoo Lee. Fully homomorphic encryption for matrix arithmetic. In *Advances in Cryptology – CRYPTO*, 2026.

[HHC19] Kyoohyung Han, Minki Hhan, and Jung Hee Cheon. Improved homomorphic discrete fourier transforms and FHE bootstrapping. *IEEE Access*, 2019.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In *Advances in Cryptology – CRYPTO*, Lecture Notes in Computer Science, 2014.

[HS18] Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in HElib. In *Advances in Cryptology – CRYPTO*, 2018.

[HS21] Shai Halevi and Victor Shoup. Bootstrapping for helib. *Journal of Cryptology*, 2021.

[JKLS18] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *ACM SIGSAC CCS*, 2018.

[JLK+22] Jaehee Jang, Younho Lee, Andrey Kim, Byunggook Na, Donggeon Yhee, Byunghan Lee, Jung Hee Cheon, and Sungroh Yoon. Privacy-preserving deep sequential model with matrix homomorphic encryption. In *ACM AsiaCCS*, 2022.

[KP23] Jonghyun Kim and Jong Hwan Park. NTRU+: Compact construction of NTRU using simple encoding method. *IEEE Transactions on Information Forensics and Security*, 2023.

[Lat24] Lattigo v6. Online: `https://github.com/tuneinsight/lattigo`, August 2024. EPFL-LDS, Tune Insight SA.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology – EUROCRYPT*, 2010.

[LS19] Vadim Lyubashevsky and Gregor Seiler. NTTRU: Truly fast NTRU using NTT. *IACR TCHES*, 2019.

[MYJK25] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. THOR: Secure transformer inference with homomorphic encryption. In *ACM SIGSAC CCS*, 2025.

[Par25] Jai Hyun Park. Ciphertext-ciphertext matrix multiplication: Fast for large matrices. In *Advances in Cryptology – EUROCRYPT*, pages 153–180, 2025.

[PPP+26] Jaiyoung Park, Sejin Park, Jai Hyun Park, Jung Ho Ahn, Jung Hee Cheon, Guillaume Hanrot, Jung Woo Kim, Minje Park, and Damien Stehlé. Scaling up fhe-based privacy-preserving ml: Higher throughput, longer inputs for llama-3-8b, 2026.

[Rad68] Charles M. Rader. Discrete fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, 1968.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 2009.

[RKP+25] Donghwan Rho, Taeseong Kim, Minje Park, Jung Woo Kim, Hyunsik Chae, Ernest K. Ryu, and Jung Hee Cheon. Encryption-friendly LLM architecture. In *The Thirteenth International Conference on Learning Representations*, 2025.

[SEA26] Microsoft SEAL (release 4.3). `https://github.com/Microsoft/SEAL`, April 2026. Microsoft Research, Redmond, WA.

[The25] The FLINT team. *FLINT: Fast Library for Number Theory*, 2025. Version 3.3.1, `https://flintlib.org`.

[ZsHCJ24] Lin Zhu, Qiang sheng Hua, Yi Chen, and Hai Jin. Secure outsourced matrix multiplication with fully homomorphic encryption. In *Computer Security – ESORICS*, 2024.