Homomorphic Encryption and Bootstrapping for Real Matrices over Conjugate-Invariant Rings

Ilaria Chillotti [1], Hyojun Kim [1], Yeongmin Lee [1], Yongwoo Lee [1,2], and Damien Ligier [1]

[1] *DESILO Inc., Seoul, Korea,* [2] *Inha University*

November 13, 2026

Abstract. The conjugate-invariant variant of CKKS (CI-CKKS) by Kim and Song encrypts real vectors over the subring of a power-of-two cyclotomic ring fixed by $X \mapsto X^{-1}$, with as many real slots as the rank of the ring. The Gentry–Lee scheme (GL) packs batches of complex matrices into a three-variable ring and multiplies them natively. GL bootstrapping replaces the rotations of the slot transforms with ciphertext–plaintext matrix products. We combine the two lines of work. First, we give a bootstrapping procedure for CI-CKKS: its transforms factor the real decoding matrix into sparse stages, and its modular reduction runs only once, on real slots, and needs no conjugation key. Second, by folding the three GL variables under their inversions, we obtain three subrings that have real slots and are closed under the matrix product; they define CI^1-GL, CI^2-GL, and CI^3-GL. We describe CI^3-GL in detail: it encrypts real matrices, needs no conjugation, and, for the same matrix size and number of matrices, has keys of half the rank of GL keys. Third, we adapt the matrix-product bootstrapping of GL to CI^3-GL. We prototyped these techniques.

Keywords. Homomorphic Encryption · CKKS · GL · Conjugate-Invariant Ring · Matrix Arithmetic · Bootstrapping

1. Introduction

Fully homomorphic encryption (FHE) allows computation on encrypted data without decrypting it. The CKKS scheme [CKKS17] supports approximate arithmetic: a plaintext polynomial encodes a vector of complex numbers, called slots, and ciphertext additions and multiplications act slot-wise. Each multiplication is followed by a rescale that consumes part of the ciphertext modulus. Bootstrapping restores the modulus by combining the four steps: ModRaise, the coefficient-to-slot transform CtS, the modular reduction EvalMod, and the slot-to-coefficient transform StC [CHK+18, CCS19, BMTH21].

Applications such as machine learning compute on real numbers, so the imaginary part of each CKKS slot goes unused. Kim and Song [KS19] build an approximate scheme over the conjugate-invariant (CI) ring $\mathcal{R}^{\mathsf{CI},X}_{\{X\}} = \{f \in \mathbb{Z}[X]/\langle \Phi_n(X)\rangle : f(X) = f(X^{-1})\}$, the ring of integers of the maximal real subfield of the n-th cyclotomic field. Its canonical embedding is real, so a plaintext holds $\phi(n)/2$ real slots, as many as the rank of the ring. Adapting the reduction of Peikert, Regev, and Stephens-Davidowitz [PRS17], they show that RLWE over this ring is no easier than a lattice problem over ideal lattices. They also report twice as many slots as CKKS for real data at the same security level, storage, and

computational cost. However, their paper gives no bootstrapping procedure. We call their scheme CI-CKKS.

Matrix products dominate privacy-preserving machine-learning inference, and vector slots do not represent them natively [ST26]. Gentry and Lee [GL25] proposed a scheme, called GL, whose plaintext ring $\mathbb{Z}[\mathrm{i}][X,Y,W]/\langle X^d-\mathrm{i},\ Y^d-\mathrm{i},\ \Phi_p(W)\rangle$ encodes $\phi(p)$ complex $d\times d$ matrices: X indexes the rows, Y the columns, and W the batch. The matrix product is computed as a trace over an auxiliary variable. On ciphertexts, the product therefore reduces to four products, per batch slot, of $d\times d$ matrices over $\mathbb{Z}_q[\mathrm{i}]$, with no conversion between slots and coefficients. Transposes and rotations of the rows, columns, and batch are ring automorphisms; each is followed by a key switch if it changes the key. A GL ciphertext is a collection of d RLWE ciphertexts over a ring isomorphic to $\mathbb{Z}[\mathrm{i}][X,W]/\langle X^d-\mathrm{i},\Phi_p(W)\rangle\cong\mathbb{Z}[X]/\langle\Phi_{4dp}(X)\rangle$, so CKKS bootstrapping applies to each of them [GL25]. Crockett, Gentry, Kim, Lee, and Lee [CGK+26] bootstrap GL using its own matrix product: CtS and StC become ciphertext–plaintext matrix products with a constant number of key switches per transform.

GL slots are complex, so for real matrices the imaginary parts carry no data, as in CKKS before Kim and Song. We ask whether the conjugate-invariant construction carries over to GL, that is, whether a subring of $\mathbb{Z}[X,Y,W]/\langle\Phi_n(X),\Phi_n(Y),\Phi_p(W)\rangle$ with $n=4d$ (the GL ring $\mathbb{Z}[\mathrm{i}][X,Y,W]/\langle X^d-\mathrm{i},\ Y^d-\mathrm{i},\ \Phi_p(W)\rangle$ is its quotient by Y^d-X^d, not a subring) has real slots and still supports the native matrix product, the transposes, and bootstrapping by matrix products. Two obstacles arise. First, GL has three variables, each with its own inversion, so there are several ways to fold the ring (i.e., to keep only the elements invariant under some of these inversions), and not every folded ring is closed under the matrix product. Second, bootstrapping CI-CKKS natively has only just been addressed: Lattigo [Ins24] supports CI-CKKS but bootstraps it in the standard CKKS ring (it packs two CI ciphertexts into one CKKS ciphertext, bootstraps it, and unpacks the result), and a first native CI-CKKS bootstrapping, based on a real FFT over the conjugate-invariant ring, appeared in concurrent work by Liu, Ma, Wang, and Wang [LMWW26].

For a power of two n and an odd p, each of the variables X, Y, and W within the ring $\mathbb{Z}[X,Y,W]/\langle\Phi_n(X),\Phi_n(Y),\Phi_p(W)\rangle$ has an inversion, such as $X\mapsto X^{-1}$. Every subgroup of the group generated by the three inversions fixes a subring of this ring, and the subring has real slots exactly when the subgroup contains the joint inversion of all three variables. Section 4 shows these subrings in a single diagram. Four of them have real slots and are closed under the matrix product. We build schemes on three of them and call them CI^k-GL, where $k\in\{1,2,3\}$ is the number of independent inversions that generate the subgroup. CI^1-GL folds the three variables jointly, CI^2-GL folds W separately and (X,Y) jointly, and CI^3-GL folds each variable separately. Our main construction, CI^3-GL, is a variant of GL whose slots are all real. Its key ring is the tensor product of the conjugate-invariant ring in X, which carries the rows, and the conjugate-invariant ring in W, which carries the batch, and all its slots are real. Tensoring two copies of the key ring over the conjugate-invariant ring in W gives the plaintext ring, the tensor product of the conjugate-invariant rings in X, Y, and W, whose slots hold $\phi(p)/2$ real $d\times d$ matrices. Swapping X and Y is a ring automorphism that transposes all these matrices at once, followed by a key switch. As in GL, a ciphertext is a collection of d RLWE ciphertexts, all under one secret in the key ring, so the secret key stays in this small ring. The security of the ciphertexts reduces to decisional RLWE over the key ring, with a circular-security assumption for the

key-switching keys. Since this ring is a tensor product of three conjugate-invariant rings, one per axis, its algorithms act axis by axis, using the tools of CI-CKKS. We therefore first bootstrap CI-CKKS and then build CI^3-GL and its bootstrapping on top of it.

1.1. Our Contributions

Our main contribution is the conjugate-invariant version of GL and its bootstrapping by matrix products (Section 4 and Section 5). Since this bootstrapping acts axis by axis with the tools of CI-CKKS, it needs a native bootstrapping for CI-CKKS, which we developed independently of and concurrently with Liu et al. [LMWW26].

First, in Section 3, we give the four steps of bootstrapping for the CI-CKKS. The transforms use a factorization of the real decoding matrix into $\log_2 N$ sparse stages, where N is the number of slots; this factorization is the real analog of the CKKS FFT; up to a row permutation and a factor of 2, the real decoding matrix is a DCT-III. With a baby-step giant-step split, a transform has depth $\log_2 N$ and uses $3\log_2 N - 3$ rotations with $2\log_2 N - 2$ rotation keys; at $N = 2^{15}$, this amounts to 42 rotations and 28 keys at depth 15. CtS^* and StC^*, which are variants of CtS and StC for CI-CKKS, share the same keys since StC^* rotates by the same amounts as CtS^*. After CtS, one ciphertext holds all N entries as real slots, so a single real EvalMod reduces them, with no split into real and imaginary parts and no conjugation key. The price is a larger overflow after ModRaise: for a ternary secret with $\mathfrak{h}$ nonzero entries, its variance is about $(1 + 2\mathfrak{h})/12$ per entry and about $(1 + 4\mathfrak{h})/12$ for the constant coordinate, compared with $(1 + \mathfrak{h})/12$ for CKKS with the same $\mathfrak{h}$; in addition, for $N \geq 8$, each stage except the last has condition number 2 instead of 1.

Second, in Section 4, we classify the folds of the GL variables, identify the four folds with real slots and a closed matrix product, and define CI^1-GL, CI^2-GL, CI^3-GL on three of them, and give their ranks. For the same matrix size d and the same number $\phi(p)$ of matrices (batch index $3p$), CI^1-GL and CI^3-GL have keys of rank $d\phi(p)$ and plaintexts of rank $d^2\phi(p)$, compared with $2d\phi(p)$ and $2d^2\phi(p)$ for GL.

In the same section, we give a detailed description of CI^3-GL, the scheme that folds each axis separately: its rings, its integer coefficient tensors, its encoding, its operations, and its key switches. A plaintext holds $(p-1)/2$ real $d \times d$ matrices, with $d = N$. The matrix product maps the real matrices $M_A^{(\kappa)}$, $M_B^{(\kappa)}$ of each batch slot κ to $M_A^{(\kappa)} M_B^{(\kappa)\mathsf{T}}$ and, modulo a prime $q \equiv 1 \pmod{n}$ with $q \equiv \pm 1 \pmod{p}$, costs $\phi(p)/2$ products of $d \times d$ matrices over $\mathbb{Z}_q$ per RNS limb. Its ring products, including those in key switching, are computed by an efficient NTT for the CI^3-GL ring, which corresponds to the number-theoretic variant of the discrete cosine transformation. Its per-slot error variance is also larger than that of GL (Section 4.5).

Third, Section 5 turns to bootstrapping. All three schemes can be bootstrapped using rotations only. For CI^3-GL, we detail a bootstrapping procedure based on matrix products that follows Crockett et al. [CGK+26]. Per ciphertext and per direction (CtS or StC), it costs 2 big key switches and $2\lceil d/\mathfrak{k} \rceil$ auxiliary key switches, where $\mathfrak{k}$ is the number of batch slots, not counting the shortcuts of Crockett et al. [CGK+26]. This gives $2 + 4$ at $(d, p) = (256, 257)$, which holds 128 matrices, and $2 + 2$ at $d = 256$ with batch index $771 = 3 \cdot 257$ (a cost count; correctness is proved for prime p), which holds the 256 matrices of GL at $p = 257$. In the latter setting, its auxiliary key has the size of 256 small keys of CI^3-GL, each half the size of a GL small key, compared with 256 small GL keys for the auxiliary key of GL bootstrapping.

Finally, we implemented all these techniques as prototypes (Section 6.1). Section 6.2 summarizes what is known about the hardness of RLWE over the key rings; we do not give a new reduction.

1.2. Related Work

CKKS bootstrapping follows the blueprint of Cheon et al. [CHK+18]; Chen, Chillotti, and Song [CCS19] evaluate the DFT-like transforms with level collapsing and approximate the modular reduction with Chebyshev polynomials. Bossuat et al. [BMTH21] remove the need for sparse secrets, optimize key switching with double hoisting, and parameterize the procedure by its failure probability. Lattigo [Ins24] supports CI-CKKS but bootstraps it in the standard ring: it packs two CI ciphertexts into one CKKS ciphertext of twice the ring degree, bootstraps that ciphertext, and unpacks the result into two CI ciphertexts. Concurrently with this work, Liu, Ma, Wang, and Wang [LMWW26] give a native bootstrapping for CI-CKKS based on a real FFT over the conjugate-invariant ring, together with a functional bootstrapping; we have not yet compared their transforms with ours in detail. SCORE [Seu25] modifies StC for real vectors through the skew-symmetry of their plaintext polynomials, with speedups of up to $2\times$. Unlike CI-CKKS, it keeps the CKKS ring. PaCo [CS25] bootstraps through a partial CtS and blind rotations; we compare the condition numbers of our real stages with the complex Cooley–Tukey stages of its factorization.

MHEAAN [CKY18] extends HEAAN with a tensor structure of slots that supports rotations along several dimensions; the key ring of CI^1-GL is the fixed ring of its conjugation. MatHEAAN [JLK+22] extends CKKS with matrix representations for recurrent networks. Zheng, Li and Wang [ZLW25] give an encoding in which the product of two encrypted $d \times d$ matrices needs $\tilde{O}(1)$ basic operations when d is at most the cube root of the ring dimension. İzbudak et al. [İKOS26] bilinearly embed matrix algebras into cyclotomic rings, using the maximal real subfield and self-dual normal bases.

Bae et al. [BCH+24] reduce plaintext–ciphertext matrix products to products of plaintext matrices through multi-secret RLWE, and fuse them with batch bootstrapping. Cheon, Kang, and Lee [CKL25] extend this reduction to batches of small matrices encoded over algebraic integers.

GL [GL25] encodes matrices in slots and multiplies them natively, and Crockett et al. [CGK+26] bootstrap it with matrix products (Section 2.5). Shokri and Tsoutsos [ST26] give a low-depth complex bootstrapping procedure for GL with a two-stage StC, a native matrix product along X, and a diagonal baby-step giant-step transform along W. All of these works on GL use complex slots.

2. Background

In this section, we recall the notation and the schemes that the remaining sections build on: CKKS, GL, CI-CKKS, and GL bootstrapping.

2.1. Notation and Conventions

Every algorithm in this paper works on integer vectors and tensors. A tensor is a three-index array $T \in \mathcal{K}^{I_1 \times I_2 \times I_3}$ where I_1, I_2 and I_3 are non-negative integers with entries $T[i,k,u]$ for $0 \le i < I_1$, $0 \le k < I_2$, and $0 \le u < I_3$ in a ring $\mathcal{K}$: $\mathbb{Z}$ (or $\mathbb{Z}[\mathrm{i}]$) for plaintexts and secret keys, $\mathbb{Z}_{Q_\ell}$ for ciphertexts at level ℓ, and $\mathbb{R}$ or $\mathbb{C}$ for slots. Here $Q_\ell = q_0 q_1 \cdots q_\ell$

is the modulus at level ℓ. Indices start at 0. Ring elements are set in lowercase, matrices and 3-tensors in plain uppercase, and vectors carry an arrow and are column vectors (so $D_n\vec{v}$ is a matrix–vector product); $T[:,k,:]$ is the slice of T at axis-2 index k (likewise for the other axes).

The axis-k product $T \times_k M$ applies a matrix M to the k-th index of T and leaves the other two unchanged; the subscript k always refers to an axis of T, not of M. For $M_1 \in \mathcal{K}^{J\times I_1}$, $(T \times_1 M_1)[j,k,u] = \sum_{i<I_1} M_1[j,i]\, T[i,k,u]$, and likewise for $\times_2$ and $\times_3$, with given matrices $M_2 \in \mathcal{K}^{J\times I_2}$ and $M_3 \in \mathcal{K}^{J\times I_3}$, respectively. For $I_3 = 1$, $T \times_1 M_1 = M_1 T$ and $T \times_2 M_2 = TM_2^{\mathsf{T}}$. Products along different axes commute, and $(T \times_k M) \times_k M' = T \times_k (M'M)$. The transpose T^{T} swaps axes 1 and 2, $\bar{T}$ is the entrywise complex conjugate, and $M^{\mathsf{H}} := \bar{M}^{\mathsf{T}}$ is the conjugate transpose of a matrix.

We write Ω for the signed permutation of the real factorization of Section 3.3, $\delta_{i,j}$ for the Kronecker delta, $\vec{1}$ for the all-ones vector, Id_n for the identity matrix of size n (omitting n when it is clear from context), and cond for the condition number. The operator $\lfloor\cdot\rceil$ rounds to the nearest integer, and $[x]_q$ is the representative of x mod q in $[-q/2, q/2)$; both apply entrywise, and to real and imaginary parts separately. Finally, Φ_m is the m-th cyclotomic polynomial, ϕ is Euler's totient function, $\mathcal{R}_m := \mathbb{Z}[X]/\langle\Phi_m(X)\rangle$, and $\mathcal{R}_{m,q} := \mathcal{R}_m/q\mathcal{R}_m$.

2.2. CKKS

CKKS [CKKS17] is the starting point of every scheme in this paper. Let n be a power of two, so that $\Phi_n(X) = X^{\phi(n)} + 1$ with $\phi(n) = n/2$, and let $N := \phi(n)/2$ and $\mathcal{R} := \mathcal{R}_n$. Let $\zeta := e^{2\pi\mathrm{i}/n}$. The slots of $m \in \mathcal{R}$ are the N complex values $m(\zeta^{5^j})$, $0 \le j < N$; the other primitive roots ζ^{-5^j} give their conjugates. Encoding with a scaling factor Δ maps a complex vector to the polynomial with these values, multiplies it by Δ, and rounds the coefficients; decoding evaluates the polynomial and divides by Δ.

A ciphertext at level ℓ is a pair $(a,b) \in \mathcal{R}_{Q_\ell}^2$ with $as + b = m + e \pmod{Q_\ell}$, where $s \in \mathcal{R}$ is the secret key and $e \in \mathcal{R}$ a small error. Addition is componentwise. A product of ciphertexts decrypts under $(1, s, s^2)$ and is relinearized by a key switch from s^2 to s. The rescale $\mathsf{Rescale}$ divides by the last prime q_ℓ and drops one level. The automorphism $\theta_{5^k} : X \mapsto X^{5^k}$ rotates the slots by k, and $X \mapsto X^{-1}$ conjugates them; both are followed by a key switch back to s, with one rotation key per rotation amount k and one conjugation key.

Bootstrapping [CHK+18, CCS19, BMTH21] starts from a ciphertext at level 0. $\mathsf{ModRaise}$ reads it modulo a larger Q, so that it decrypts to $m + e + q_0 e_{\mathsf{ModRaise}}$ for a small integer polynomial e_{ModRaise}. CtS multiplies the slots by the inverse of the decoding matrix, so that they hold the coefficients; since there are $2N$ real coefficients but only N complex slots, the coefficients occupy both real and imaginary parts, which are then separated using a conjugation. $\mathsf{EvalMod}$ evaluates an approximation of $x \mapsto [x]_{q_0}$ on the slots, e.g., a scaled sine, and StC multiplies by the decoding matrix. The transforms factor the decoding matrix into sparse FFT-like stages and evaluate each stage with a few rotations.

2.3. GL

GL [GL25] packs $\phi(p)$ complex $d \times d$ matrices into one plaintext. We present it using the axis products of Section 2.1, so that GL and the schemes of Section 4 can be compared formula by formula.

The matrix size d is a power of two and p is an odd prime, so $\phi(p) = p - 1$; GL allows any odd p, Crockett et al. [CGK+26] allow any power of an odd prime, and Section 4 also uses composite p. With $\mathsf{i}^2 = -1$, the plaintext ring is $\mathcal{R}^{\mathsf{GL}} := \mathbb{Z}[\mathsf{i}][X, Y, W]/\langle X^d - \mathsf{i},\ Y^d - \mathsf{i},\ \Phi_p(W)\rangle$. A secret key has no Y. Setting $\mathsf{i} = X^d$ gives $\mathbb{Z}[\mathsf{i}][X]/\langle X^d - \mathsf{i}\rangle \cong \mathcal{R}_{4d}$, and since $\gcd(4d, p) = 1$, the key ring $\tilde{\mathcal{R}}^{\mathsf{GL}} := \mathbb{Z}[\mathsf{i}][X, W]/\langle X^d - \mathsf{i},\ \Phi_p(W)\rangle$ is isomorphic to $\mathbb{Z}[X]/\langle \Phi_{4dp}(X)\rangle$, of degree $2d\phi(p)$. We identify a key with the corresponding element of $\tilde{\mathcal{R}}^{\mathsf{GL}} \subset \mathcal{R}^{\mathsf{GL}}$. The index $4dp$ is not a power of two, and $d = \phi(4d)/2$ is the N of Section 2.2 at $n = 4d$.

The slots are evaluations at roots of unity. Let $\zeta_{4d} := e^{2\pi\mathsf{i}/(4d)}$ and $\eta := e^{2\pi\mathsf{i}/p}$, and let γ be a primitive root mod p, so that $\gamma^{(p-1)/2} \equiv -1 \pmod{p}$. The κ-th matrix of $m \in \mathcal{R}^{\mathsf{GL}}$, $0 \le \kappa < p - 1$, has the entries $m(\zeta_{4d}^{5^j}, \zeta_{4d}^{5^k}, \eta^{\gamma^\kappa})$, $0 \le j, k < d$.

GL algorithms do not act on polynomials but on their arrays of coefficients, which we call tensors. Every $m \in \mathcal{R}^{\mathsf{GL}}$ is uniquely $m = \sum_{a,k<d} \sum_{u<p-1} m_{a,k,u} X^a Y^k W^u$ with $m_{a,k,u} \in \mathbb{Z}[\mathsf{i}]$; its tensor is $\mathsf{PT} \in \mathfrak{R}^{\mathsf{GL}} := \mathbb{Z}[\mathsf{i}]^{d\times d\times(p-1)}$, with $\mathsf{PT}[a, k, u] = m_{a,k,u}$, whose axes 1, 2, 3 index the powers of X, Y, W. Since the entries are the coefficients themselves, we identify m with PT: the product $\mathsf{PT} \cdot \mathsf{PT}'$ of two tensors is the tensor of the product of their polynomials, given explicitly in the table below. With entrywise addition, $\mathfrak{R}^{\mathsf{GL}}$ is a commutative ring isomorphic to $\mathcal{R}^{\mathsf{GL}}$; $\cdot$ is always the ring product. Modulo q, we write $\mathcal{R}_q^{\mathsf{GL}}$ and $\mathfrak{R}_q^{\mathsf{GL}} := \mathbb{Z}_q[\mathsf{i}]^{d\times d\times(p-1)}$. The tensor S of a key s has $S[a, k, u] = 0$ for $k \neq 0$, since s has no Y.

Definition 1 (GL ciphertext, [GL25])**.** A GL ciphertext at level ℓ under a secret key $s \in \tilde{\mathcal{R}}^{\mathsf{GL}}$ is a pair $(a, b) \in (\mathcal{R}_{Q_\ell}^{\mathsf{GL}})^2$ with $as + b = m + e \pmod{Q_\ell}$ for a message m and a small error e. It is stored as (A, B), and $A \cdot S + B = \mathsf{PT} + E$ for the tensors S, PT, E of s, m, e.

Lemma 1 ([GL25])**.** *Slice k of $A \cdot S + B$ is $A[:, k, :] \cdot S[:, 0, :] + B[:, k, :]$, a product in $\tilde{\mathcal{R}}^{\mathsf{GL}}$. Hence a GL ciphertext is a collection of d RLWE ciphertexts $(A[:, k, :], B[:, k, :])$ over $\tilde{\mathcal{R}}^{\mathsf{GL}}$ under one key, one per power Y^k.*

A message is a tensor $M \in \mathbb{C}^{d\times d\times(p-1)}$ with matrices $M^{(\kappa)} := M[:, :, \kappa]$. Decoding evaluates all three variables at once, with one axis product per variable, using $D_X^{\mathsf{GL}} \in \mathbb{C}^{d\times d}$, $D_X^{\mathsf{GL}}[j, a] = \zeta_{4d}^{5^j a}$, and $D_W^{\mathsf{GL}} \in \mathbb{C}^{(p-1)\times(p-1)}$, $D_W^{\mathsf{GL}}[\kappa, u] = \eta^{\gamma^\kappa u}$: $\mathsf{Decode}^{\mathsf{GL}}(\mathsf{PT}, \Delta) := \Delta^{-1}\,\mathsf{PT} \times_1 D_X^{\mathsf{GL}} \times_2 D_X^{\mathsf{GL}} \times_3 D_W^{\mathsf{GL}}$. Encoding inverts decoding and rounds: $\mathsf{Encode}^{\mathsf{GL}}(M, \Delta) := \lfloor \Delta\, M \times_1 D_X^{\mathsf{GL}^{-1}} \times_2 D_X^{\mathsf{GL}^{-1}} \times_3 D_W^{\mathsf{GL}^{-1}} \rceil$, with $D_X^{\mathsf{GL}^{-1}} = D_X^{\mathsf{GL}^{\mathsf{H}}}/d$ and $D_W^{\mathsf{GL}^{-1}}[u, \kappa] = (\eta^{-\gamma^\kappa u} - \eta^{\gamma^\kappa})/p$.

Every GL operation is an axis product/addition, a contraction with a fixed integer tensor, or a transpose; the constants express the reductions $X^d = \mathsf{i}$, $Y^d = \mathsf{i}$ and $\Phi_p(W) = 0$ on coefficients. Table 1 summarizes the operations and the constants they use. $\Pi_\nu \in \mathbb{Z}[\mathsf{i}]^{d\times d}$ is the map $X^a \mapsto X^{5^\nu a}$, $\Pi_\nu[c, a] = \mathsf{i}^{\lfloor 5^\nu a/d\rfloor}\delta_{c,5^\nu a \bmod d}$, and $\Pi_\nu^W \in \mathbb{Z}^{(p-1)\times(p-1)}$ the map $W^u \mapsto W^{\gamma^\nu u}$, with column u equal to $\vec{\omega}_W(\gamma^\nu u)$. $\Theta_{-1} \in \mathbb{Z}[\mathsf{i}]^{d\times d}$ is the map $X^a \mapsto X^{-a}$: $\Theta_{-1}[0, 0] = 1$, $\Theta_{-1}[d - a, a] = -\mathsf{i}$ for $0 < a < d$, and 0 elsewhere. Moreover, $\Pi_{(p-1)/2}^W$ is the map $W \mapsto W^{-1}$.

In coefficients, the matrix product uses $B^\flat := \bar{B} \times_1 \Theta_{-1} \times_3 \Pi_{(p-1)/2}^W$. In Table 1, M_A is the message of A at $\Delta = 1$, $M_A^{(\kappa)}$ its matrices, and M the message of the result. All formulas are $\mathbb{Z}$-linear or $\mathbb{Z}$-bilinear, so they hold modulo Q_ℓ.

operation	**tensor**	**message**
ring product $\cdot$	$(A \cdot B)[a'', k'', u''] = \sum_{\substack{a,a',k,k'<d \\ u,u'<p-1}} (\gamma_{\bar{a}}\gamma_{\bar{k}}\gamma_{\bar{u}}\alpha\beta)$ with $\gamma_{\bar{a}} = \Gamma^X[a, a', a'']$, $\gamma_{\bar{k}} = \Gamma^X[k, k', k'']$, $\gamma_{\bar{u}} = \Gamma^W[u, u', u'']$, $\alpha = A[a, k, u]$, $\beta = B[a', k', u']$	entrywise product
matrix product $\circledast$	$(A \circledast B)[a, k, w] = d \sum_{\substack{v<d, \\ u,u'}} \Gamma^W[u, u', w] A[a, v, u] B^\flat[k, v, u']$	$M^{(\kappa)} = M_A^{(\kappa)} M_B^{(\kappa)\mathsf{H}}$
conjugation	$\bar{A} \times_1 \Theta_{-1} \times_2 \Theta_{-1} \times_3 \Pi^W_{(p-1)/2}$	$\bar{M}_A$
transpose	A^T	M_A^T
row, column rotation by ν	$A \times_1 \Pi_\nu$, $A \times_2 \Pi_\nu$	row, column index shifted by ν mod d
batch rotation by ν	$A \times_3 \Pi^W_\nu$	κ shifted by ν mod $p-1$

Table 1. GL operations on coefficient tensors A, B, and their constants. The structure tensor $\Gamma^X \in \mathbb{Z}[\mathsf{i}]^{d\times d\times d}$ holds the coefficients of $X^a X^b$ after reduction, $\Gamma^X[a, b, c] = \mathsf{i}^{\lfloor (a+b)/d \rfloor} \delta_{c,(a+b) \bmod d}$; it is also used for Y. $\vec{\omega}_W(t) \in \mathbb{Z}^{p-1}$ is the coefficient vector of W^t: $\omega_W(t)_u = \delta_{u,t \bmod p}$ if $t \bmod p < p-1$, and $\vec{\omega}_W(t) = -\vec{1}$ if $t \equiv p-1 \pmod p$. The structure tensor of W is $\Gamma^W \in \mathbb{Z}^{(p-1)\times(p-1)\times(p-1)}$, $\Gamma^W[u, v, :] = \vec{\omega}_W(u+v)$.

The matrix product is a trace in an auxiliary variable Z, which sums over the shared inner index. For f, h with tensors A, B, $f \circledast h := \mathsf{Trace}_Z\big(f(X, Z, W)\, \bar{h}(Y^{-1}, Z^{-1}, W^{-1})\big)$ with $\mathsf{Trace}_Z := d\,[Z^0]$, the trace of Crockett et al. [CGK+26], where $[Z^0]$ takes the coefficient of Z^0 (with $Z^d = \mathsf{i}$), and the bar also applies $\mathsf{i} \mapsto -\mathsf{i}$. With $M_A := A \times_1 D_X^{\mathsf{GL}} \times_2 D_X^{\mathsf{GL}} \times_3 D_W^{\mathsf{GL}}$ the message of A at $\Delta = 1$ and $M_A^{(\kappa)}$ its matrices, the κ-th message of $A \circledast B$ has the matrices $M_A^{(\kappa)} M_B^{(\kappa)\mathsf{H}}$ (Table 1).

A homomorphic operation applies the map of Table 1 to both components of a ciphertext; this changes the key under which the result decrypts, and a key switch brings it back to s. We call a key *small* if it has no Y, and *big* otherwise.

A key switch $\mathsf{Switch}(\hat{C}, \mathsf{ksk}_{s'\to s})$ turns a tensor $\hat{C}$ that decrypts under s' into a ciphertext under s, with a key-switching key $\mathsf{ksk}_{s'\to s}$ made of dnum RNS digits and a special modulus P, as in GL [GL25] (its Sections 2.3 and 3.2); Lemma 2 states its correctness. Let P be the special modulus, and split Q_ℓ into dnum partitions with RNS constants g_r. $\mathsf{Decompose}^{\mathsf{RNS}}(\hat{C}) := (\hat{C}_r)_{r<\mathsf{dnum}}$ satisfies $\sum_r g_r \hat{C}_r \equiv \hat{C} \pmod{Q_\ell}$, computed entrywise on real and imaginary parts. The key-switching key from s' to s is $\mathsf{ksk}_{s'\to s} := (\tilde{a}_r,\ -\tilde{a}_r s + \tilde{e}_r + P g_r s')_{r<\mathsf{dnum}}$ modulo $Q_\ell P$, with tensors $\tilde{A}_r$, $\tilde{E}_r$. $\mathsf{Switch}(\hat{C}, \mathsf{ksk}_{s'\to s})$ sums $\hat{C}_r \cdot \mathsf{ksk}_r$ over r into (A'', B'') and returns $(A^{\mathsf{ks}}, B^{\mathsf{ks}}) := (\lfloor P^{-1} A'' \rceil, \lfloor P^{-1} B'' \rceil)$.

Lemma 2 (key switching, [GL25]). *$A'' \cdot S + B'' \equiv P\,\hat{C} \cdot S' + \sum_r \hat{C}_r \cdot \tilde{E}_r \pmod{Q_\ell P}$, and $A^{\mathsf{ks}} \cdot S + B^{\mathsf{ks}} \equiv \hat{C} \cdot S' + E^{\mathsf{ks}} \pmod{Q_\ell}$ with $E^{\mathsf{ks}} := P^{-1} \sum_r \hat{C}_r \cdot \tilde{E}_r + \epsilon_A \cdot S + \epsilon_B$, where ϵ_A, ϵ_B are the rounding errors (entries, or their real and imaginary parts, in $[-1/2, 1/2]$).*

For a small source key, $\mathsf{Switch}_{\mathsf{small}}$ acts on each axis-2 slice separately; for a big one, $\mathsf{Switch}_{\mathsf{big}}$ mixes slices.

Rotations and the conjugation lead to small keys; the transpose $\mathsf{Transpose}$, the conjugate transpose $\mathsf{ConjTranspose}$, and the ct–ct product need big keys [GL25].

2.4. Conjugate-Invariant CKKS

CI-CKKS [KS19] removes the unused imaginary parts by restricting CKKS to the elements fixed by $X \mapsto X^{-1}$.

Definition 2 (conjugate-invariant ring, [KS19])**.** For $n \geq 8$ a power of two and $N = \phi(n)/2$ as in Section 2.2, $\mathcal{R}_{\{X\}}^{\mathsf{CI},X} := \{f \in \mathcal{R} : f(X) = f(X^{-1})\}$, with the $\mathbb{Z}$-basis $\vec{c}_n^{\mathsf{CI}} := (c_u)_{u<N}$, $c_0 = 1$, $c_u = X^u + X^{-u} = X^u - X^{2N-u}$.

An element $v \in \mathcal{R}_{\{X\}}^{\mathsf{CI},X}$ is stored as its coordinate vector $\vec{v} \in \mathfrak{R}_{\{X\}}^{\mathsf{CI},X} := \mathbb{Z}^N$, with $\mathsf{Expand}(\vec{v}) := \langle \vec{v}, \vec{c}_n^{\mathsf{CI}} \rangle = v$ and its inverse Compact. With entrywise addition and the product $\vec{v} \cdot \vec{w} := \mathsf{Compact}(\mathsf{Expand}(\vec{v})\,\mathsf{Expand}(\vec{w}))$, $\mathfrak{R}_{\{X\}}^{\mathsf{CI},X}$ is a ring isomorphic to $\mathcal{R}_{\{X\}}^{\mathsf{CI},X}$; as in Section 2.3, $\cdot$ is the ring product. The products of basis elements are $c_u c_w = (X^{u+w} + X^{-u-w}) + (X^{u-w} + X^{w-u})$. Hence every RLWE algorithm over $\mathcal{R}$ (addition, ciphertext multiplication, which multiplies the messages slot-wise, key switch, rescale) works unchanged on this subring. The rotations $X \mapsto X^{5^k}$ act on coordinates as signed permutations (Lemma 5), while $X \mapsto X^{-1}$ is the identity, so there is no conjugation. Only encoding, decoding, CtS, and StC change.

The slots are the values of an element at N real points. With $\zeta = e^{2\pi \mathrm{i}/n}$, the slot points are $\xi_j := \zeta^{5^j} + \zeta^{-5^j}$, $0 \leq j < N$, and slot j of v is $v(\zeta^{5^j})$, a real number. The decoding matrix $D_n \in \mathbb{R}^{N \times N}$ has entries $D_n[j,u] = c_u(\zeta^{5^j})$, i.e., $D_n[j,0] = 1$ and $D_n[j,u] = 2\cos(2\pi 5^j u/n)$ for $u \geq 1$.

Lemma 3 (decoding matrix, [KS19])**.** *The map $\tau(\vec{v}) := D_n\vec{v}$ is a ring homomorphism $\mathfrak{R}_{\{X\}}^{\mathsf{CI},X} \to \mathbb{R}^N$ with slot-wise products, and $D_n^{\mathsf{T}} D_n = \mathsf{diag}(N, 2N, \ldots, 2N)$. Hence $D_n^{-1} = \Lambda_n D_n^{\mathsf{T}}$ with $\Lambda_n := (D_n^{\mathsf{T}} D_n)^{-1}$, and $\mathsf{cond}(D_n) = \sqrt{2}$.*

Encoding and decoding are $\mathsf{Encode}(\vec{z}, \Delta) := \lfloor \Delta\, \Lambda_n D_n^{\mathsf{T}} \vec{z} \rceil$ and $\mathsf{Decode}(\vec{m}, \Delta) := \Delta^{-1} D_n \vec{m}$, for $\vec{z} \in \mathbb{R}^N$ and $\vec{m} \in \mathfrak{R}_{\{X\}}^{\mathsf{CI},X}$ with centered entries. A ciphertext is an RLWE pair over $\mathcal{R}_{\{X\}}^{\mathsf{CI},X}$, stored as two coordinate vectors (Section 3.1).

2.5. GL Bootstrapping

Crockett et al. [CGK+26] follow the CKKS steps but evaluate CtS and StC with the GL matrix product. We recall it in our notation, with $d = \phi(p)$ for simplicity.

Bootstrapping runs on the d RLWE ciphertexts $(A[:,k,:], B[:,k,:])$ of a GL ciphertext. ModRaise lifts each of them, so that slice k decrypts to $\mathsf{PT}[:,k,:] + E[:,k,:] + q_0 E_{\mathsf{ModRaise}}[:,k,:]$. The slots of slice k form the matrix $M_k \in \mathbb{C}^{d \times (p-1)}$, with rows along X and columns along W, given by $M_k = D_X^{\mathsf{GL}}(\mathsf{PT} + E + q_0 E_{\mathsf{ModRaise}})[:,k,:] D_W^{\mathsf{GL}\,\mathsf{T}}$. CtS must therefore compute $D_X^{\mathsf{GL}\,-1} M_k D_W^{\mathsf{GL}\,-\mathsf{T}}$ for every k, i.e., multiply along X and along W, whereas $\circledast$ multiplies by a plaintext on the right, along Y. Crockett et al. [CGK+26] resolve this mismatch with transposes.

Along X, a conjugate transpose, a ct–pt product with an encoding of $D_X^{\mathsf{GL}\,-1}$, and a second conjugate transpose together apply $D_X^{\mathsf{GL}\,-1}$. Along W, up to $\phi(p)$ RLWE ciphertexts are stacked along an auxiliary variable V, and a swap $W \leftrightarrow V$ with an auxiliary key $\mathsf{ksk}^{\mathsf{aux}}$, a trace over a further variable W', and a swap back together apply $D_W^{\mathsf{GL}\,-1}$. A transform costs four key switches and two homomorphic traces, and the auxiliary key has the size of $\phi(p)$ small GL keys. Details and further optimizations are in Crockett et al. [CGK+26].

3. Bootstrapping for CI-CKKS

Bootstrapping refreshes a ciphertext whose modulus is exhausted, so that computation can continue. In this section, we follow the classical CKKS bootstrapping structure [CHK+18],

that is, the composition $\mathsf{Boot}^{\mathsf{CI\text{-}CKKS}} := \mathsf{StC}^* \circ \mathsf{EvalMod} \circ \mathsf{CtS}^* \circ \mathsf{ModRaise}$. ModRaise and EvalMod are the CKKS steps applied to $\mathcal{R}^{\mathsf{CI},X}_{\{X\}}$, while the coefficient-to-slot and slot-to-coefficient transforms, which we write CtS^* and StC^* to distinguish them from those of CKKS, are new: they use real diagonals and only the rotations $X \mapsto X^{5^k}$, so no step needs a conjugation key. ModRaise moves the ciphertext to a large modulus at the price of an integer overflow; CtS^* moves the entries of the plaintext vector into the slots; EvalMod removes the overflow slot-wise; and StC^* moves the result back into the vector.

3.1. Ciphertexts as Vectors

All steps of bootstrapping act on the N integer coordinates of a ciphertext in the basis $\vec{c}^{\mathsf{CI}}_n$, so we write ciphertexts as vectors, with the notation of Section 2.4 and indices $0 \le u < N$. Modulo q, we write $\mathcal{R}^{\mathsf{CI},X}_{q,\{X\}} := \mathcal{R}^{\mathsf{CI},X}_{\{X\}}/q\mathcal{R}^{\mathsf{CI},X}_{\{X\}}$ for polynomials and $\mathfrak{R}^{\mathsf{CI},X}_{q,\{X\}} := \mathbb{Z}_q^N$ for vectors, with the same product taken mod q.

Definition 3 (CI-CKKS ciphertext)**.** A ciphertext at level ℓ is a pair $(a,b) \in (\mathcal{R}^{\mathsf{CI},X}_{Q_\ell,\{X\}})^2$ that decrypts under the secret key $s \in \mathcal{R}^{\mathsf{CI},X}_{\{X\}}$ as $as + b = m + e \pmod{Q_\ell}$, with m the encoded message and e a small error. Encryption samples a uniform a and a small e and outputs $(a, -as + m + e)$. Since Compact is a ring isomorphism (Section 2.4), the ciphertext is stored as $\mathsf{ct} = (\vec{a}, \vec{b}) \in (\mathfrak{R}^{\mathsf{CI},X}_{Q_\ell,\{X\}})^2$, $\vec{a} := \mathsf{Compact}(a)$, $\vec{b} := \mathsf{Compact}(b)$: two vectors of N residues, where N is also the LWE dimension. With $\vec{s}$, $\vec{m}$, $\vec{e}$ the vectors of s, m, e, decryption reads

$$\vec{a} \cdot \vec{s} + \vec{b} = \vec{m} + \vec{e} \pmod{Q_\ell}.$$

The slots of ct are those of its decryption, $D_n(\vec{m} + \vec{e})/\Delta$, with D_n the decoding matrix and Δ the scaling factor of Section 2.4. Bootstrapping starts at level 0, with modulus q_0; we take all moduli to be odd.

3.2. ModRaise

ModRaise reads a level-0 ciphertext modulo a larger $Q > q_0$ without decrypting. Each entry of $\vec{a}$ and $\vec{b}$ is replaced by its centered representative $[\cdot]_{q_0} \in [-q_0/2, q_0/2)$, and this signed integer is then read as a residue mod Q.

ModRaise is the CKKS step [CHK+18]: for odd q_0, lifting the CI coordinates of $\vec{a}$ and $\vec{b}$ gives the same polynomials as lifting the coefficients of $\mathsf{Expand}(\vec{a})$ and $\mathsf{Expand}(\vec{b})$. Hence the raised ciphertext decrypts to $\vec{m} + \vec{e} + q_0\vec{e}_{\mathsf{ModRaise}}$ modulo Q, and the overflow $\vec{e}_{\mathsf{ModRaise}}$ is again an element of $\mathfrak{R}^{\mathsf{CI},X}_{\{X\}}$.

What does change is the size of $\vec{e}_{\mathsf{ModRaise}}$, which determines the input interval of EvalMod.

Lemma 4 (overflow variance, heuristic)**.** *Model the entries of $\vec{a}\,'$, $\vec{b}\,'$ as independent and uniform on $[-q_0/2, q_0/2)$, and let $\vec{s}$ be ternary (entries in $\{0, \pm 1\}$) with $\mathfrak{h}$ nonzero entries. Then the entries $e_{\mathsf{ModRaise},u}$, $u < N$, of the overflow $\vec{e}_{\mathsf{ModRaise}} \in \mathfrak{R}^{\mathsf{CI},X}_{\{X\}} = \mathbb{Z}^N$ (its coordinates in the basis $\vec{c}^{\mathsf{CI}}_n$) satisfy $\mathsf{Var}(e_{\mathsf{ModRaise},u}) \approx (1 + 2\mathfrak{h})/12$ for $u \ge 1$ and $\mathsf{Var}(e_{\mathsf{ModRaise},0}) \approx (1 + 4\mathfrak{h})/12$, compared with $(1 + \mathfrak{h})/12$ for CKKS with a secret of the same weight $\mathfrak{h}$ [BMTH21].*

3.3. The Transforms CtS and StC

CtS^* multiplies the slot vector by ΩD_n^{-1}, so that the slots hold the entries of $\vec{m} + \vec{e} + q_0\vec{e}_{\mathsf{ModRaise}}$, signed and permuted by the signed permutation Ω defined below; StC^* multiplies it by $D_n\Omega^{\mathsf{T}}$. Homomorphically, a matrix acts on slots through plaintext products and rotations, which we recall first.

On vectors, the automorphism $\theta_{5^k} : X \mapsto X^{5^k}$ is $\vec{v} \mapsto \mathsf{Compact}(\theta_{5^k}(\mathsf{Expand}(\vec{v})))$; we write rot_k for the cyclic shift $\mathsf{rot}_k(\vec{x})[j] := x_{j+k \bmod N}$ of a vector in $\mathbb{R}^N$. The homomorphic rotation applies θ_{5^k} to both components and switches the rotated mask back to $\vec{s}$ with the rotation key $\mathsf{rk}_k := \mathsf{ksk}_{\theta_{5^k}(s)\to s}$:

$$\mathsf{Rotate}(\mathsf{ct}, \mathsf{rk}_k) := (0, \theta_{5^k}(\vec{b})) + \mathsf{Switch}(\theta_{5^k}(\vec{a}), \mathsf{rk}_k),$$

where Switch is the RLWE key switch over $\mathcal{R}^{\mathsf{CI},X}_{\{X\}}$ (the CKKS key switch [CKKS17]).

Lemma 5 (rotations). *θ_{5^k} maps $\mathcal{R}^{\mathsf{CI},X}_{\{X\}}$ to itself and acts on coordinates by a signed permutation, and the slots of $\theta_{5^k}(v)$ are rot_k of the slots of v. If ct encrypts $\vec{m}$ under $\vec{s}$, then $\mathsf{Rotate}(\mathsf{ct}, \mathsf{rk}_k)$ encrypts $\theta_{5^k}(\vec{m})$ under $\vec{s}$, up to the key-switching error.*

For a plaintext $\vec{\mathsf{pt}} \in \mathfrak{R}^{\mathsf{CI},X}_{\{X\}}$, $\vec{\mathsf{pt}} \cdot \mathsf{ct} := (\vec{\mathsf{pt}} \cdot \vec{a}, \vec{\mathsf{pt}} \cdot \vec{b})$ multiplies the slots slot-wise, since D_n is a ring homomorphism. Diagonals are encoded at a working scale Δ_w as $\mathsf{Encode}(\cdot, \Delta_w)$, and Rescale is the RNS rescale of CKKS, which drops the last prime and one level.

A dense D_n would cost $N-1$ rotations per transform ($O(\sqrt{N})$ with baby-step giant-step) and N plaintext products; a factorization into sparse stages makes the transforms cheap. A stage $S \in \mathbb{R}^{N\times N}$ is described by its offset set $O \subseteq \mathbb{Z}_N$ and its diagonals $\vec{d}_o \in \mathbb{R}^N$, $d_o[j] := S[j, j+o \bmod N]$, so that $S\vec{x} = \sum_{o\in O} \vec{d}_o \odot \mathsf{rot}_o(\vec{x})$, where $\odot$ denotes the slot-wise product.

We use the following factorization, which we call realfft3. Let $N \geq 8$. There exist a signed permutation $\Omega \in \{0, \pm 1\}^{N\times N}$ and stages $S_0, \ldots, S_{\log_2 N-1} \in \mathbb{R}^{N\times N}$ such that

$$D_n = S_{\log_2 N-1} \cdots S_1 S_0 \, \Omega,$$

where slot j of $\Omega\vec{v}$ is $\mathsf{sign}(j)\, v_{\mathsf{idx}(j)}$, $\mathsf{sign}(j) \in \{\pm 1\}$, $0 \leq \mathsf{idx}(j) < N$. The offset sets are $O_0 = \{0, N/2\}$, $O_i = \{0, \pm 2^{i-1}, N/2, N/2 \pm 2^{i-1}\}$ for $1 \leq i \leq \log_2 N - 2$, and $O_{\log_2 N-1} = \{0, \pm N/4, N/2\}$. Each S_i has condition number $\kappa(S_i) := \|S_i\|_2 \|S_i^{-1}\|_2 = 2$ (the ratio of its largest to its smallest singular value), except the last one, for which $\kappa = 1$; it bounds how much a stage can amplify the relative error of its input. Section A gives the construction and compares the condition numbers with those of complex stages.

The inverse transform requires no new stages.

Lemma 6 (inverse transform). *Under the factorization* realfft3 *(Section 3.3),*

$$\Omega\, D_n^{-1} = \Lambda_\Omega\, S_0^{\mathsf{T}} S_1^{\mathsf{T}} \cdots S_{\log_2 N-1}^{\mathsf{T}},$$

with $\Lambda_\Omega := \Omega \Lambda_n \Omega^{\mathsf{T}}$ diagonal, with entry $1/N$ at j if $\mathsf{idx}(j) = 0$ and $1/(2N)$ otherwise. The transpose S_i^{T} has diagonal $\mathsf{rot}_o(\vec{d}^{(i)}_{-o})$ at offset o, and its offset set is again O_i.

CtS* outputs the entries in the layout Ω, which is the input layout of StC*, so Ω cancels in StC* $\circ$ EvalMod $\circ$ CtS*, as the bit reversal does for the complex transforms [CCS19]. Up to a row permutation and a factor of 2, D_n is a DCT-III [ANR74] (Section A).

We write $\mathsf{Stage}(\mathsf{ct}, O, \{\vec{d}_o\}_{o\in O}, \{\mathsf{rk}_k\}) \to \mathsf{ct}'$ for the evaluation of one stage: given ct at level $\ell \geq 1$ with slots $\vec{x}$, it returns ct′ at level $\ell - 1$ with slots $\sum_{o\in O} \vec{d}_o \odot \mathsf{rot}_o(\vec{x})$, using the baby-step giant-step method of Halevi and Shoup [HS18], as in CKKS bootstrapping [CCS19, BMTH21]. With our offsets, the giant steps are $g \in \mathcal{G} := \{0, N/2\}$ and the baby steps $b \in \mathcal{B} \subseteq (-N/4, N/4]$, so a stage needs one key switch per nonzero baby step and at most one giant-step key switch.

Lemma 7 (one stage)**.** *Let* ct *at level* $\ell \geq 1$ *have the slots* $\vec{x}$*, and let* $\Delta_w = q_\ell$*. Then* Stage *returns a ciphertext at level* $\ell - 1$*, at the scale of* ct*, with the slots* $\sum_{o \in O} \vec{d}_o \odot \mathsf{rot}_o(\vec{x})$*, up to the encoding, key-switching, and rescaling errors. It uses* $|\mathcal{B} \setminus \{0\}| + |\mathcal{G} \setminus \{0\}|$ *rotations.*

Algorithm 1 and Algorithm 2 give CtS^* and StC^* as sequences of these stages.

$\mathsf{CtS}^*(\mathsf{ct}, \{\mathsf{rk}_k\}_{k \in J_{\mathsf{rot}}}) \to \mathsf{ct}'$

Input:
- ct at level $\ell \geq \log_2 N$ with slots $D_n \vec{x}/\Delta$, $\vec{x} \in \mathbb{R}^N$ (e.g., $\vec{x} = \vec{m} + \vec{e} + q_0 \vec{e}_{\mathsf{ModRaise}}$ after ModRaise)
- rotation keys rk_k for $k \in J_{\mathsf{rot}} := \{N/2,\, N/4\} \cup \{\pm 2^{i-1}\}_{1 \leq i \leq \log_2 N - 2}$

Output: $\mathsf{ct}' \in (\mathfrak{R}^{\mathsf{CI},X}_{Q_{\ell - \log_2 N}, \{X\}})^2$ with slots $\Omega \vec{x}/\Delta$, i.e., slot j holds $\mathsf{sign}(j)\, x_{\mathsf{idx}(j)}/\Delta$

Context: diagonal of S_i^{T} at offset o: $\mathsf{rot}_o(\vec{d}^{(i)}_{-o})$; Λ_Ω as above

1 **for** $i = \log_2 N - 1, \ldots, 1, 0$ **do**
2 **for** $o \in O_i$ **do**
3 $\vec{d}'_o \leftarrow \mathsf{rot}_o(\vec{d}^{(i)}_{-o})$
4 **if** $i = 0$ **then** *fold* Λ_Ω *into the last stage*
5 $\vec{d}'_o \leftarrow \Lambda_\Omega \vec{d}'_o$
6 $\mathsf{ct} \leftarrow \mathsf{Stage}(\mathsf{ct}, O_i, \{\vec{d}'_o\}_{o \in O_i}, \{\mathsf{rk}_k\})$
7 $\mathsf{ct}' \leftarrow \mathsf{ct}$
8 **return** ct′

Algorithm 1. CI-CKKS coefficient-to-slot transform

$\mathsf{StC}^*(\mathsf{ct}, \{\mathsf{rk}_k\}_{k \in J_{\mathsf{rot}}}) \to \mathsf{ct}'$

Input:
- ct at level $\ell \geq \log_2 N$ with slots $\Omega \vec{y}/\Delta$, $\vec{y} \in \mathbb{R}^N$
- rotation keys rk_k, $k \in J_{\mathsf{rot}}$ (the same keys as Algorithm 1)

Output: $\mathsf{ct}' \in (\mathfrak{R}^{\mathsf{CI},X}_{Q_{\ell - \log_2 N}, \{X\}})^2$ with slots $D_n \vec{y}/\Delta$

Context: stages S_i of the factorization realfft3 (Section 3.3), offset sets O_i, diagonals $\vec{d}^{(i)}_o$

1 **for** $i = 0, 1, \ldots, \log_2 N - 1$ **do**
2 $\mathsf{ct} \leftarrow \mathsf{Stage}(\mathsf{ct}, O_i, \{\vec{d}^{(i)}_o\}_{o \in O_i}, \{\mathsf{rk}_k\})$
3 $\mathsf{ct}' \leftarrow \mathsf{ct}$
4 **return** ct′

Algorithm 2. CI-CKKS slot-to-coefficient transform

Proposition 1 (cost of the transforms)**.** *Under the factorization* realfft3 *(Section 3.3), each of* CtS^* *and* StC^* *has depth* $\log_2 N$ *and uses* $3 \log_2 N - 3$ *rotations (*$5 \log_2 N - 6$ *without BSGS), with BSGS, with the* $2 \log_2 N - 2$ *rotation keys* rk_k*,* $k \in J_{\mathsf{rot}}$*, shared by both transforms; consecutive stages can be merged to trade depth for rotations, as in [CCS19].*

We describe only the radix-2 version.

3.4. EvalMod and the Full Bootstrapping

After CtS^*, one ciphertext holds all N signed, permuted entries of $\vec{m} + \vec{e} + q_0 \vec{e}_{\mathsf{ModRaise}}$ as real slots. EvalMod evaluates a polynomial approximation of $x \mapsto [x]_{q_0}$ on every slot (on slots scaled by Δ); its input interval is set by a bound B_{ModRaise} on $\|\vec{e}_{\mathsf{ModRaise}}\|_\infty$. One real EvalMod thus reduces all N entries, with no split into real and imaginary parts and no conjugation. For the approximation itself, any CKKS method applies unchanged (e.g., [BMTH21]); only the interval, which is set by B_{ModRaise}, depends on the scheme. Its products are relinearized with the relinearization key $\mathsf{evk} := \mathsf{ksk}_{s^2 \to s}$.

Theorem 1 (correctness of $\mathsf{Boot}^{\mathsf{CI\text{-}CKKS}}$)**.** *Assume the factorization* realfft3 *(Section 3.3), odd moduli, $\Delta_w = q_\ell$ in every stage (Lemma 7), and $L \geq 2\log_2 N + \ell_{\mathsf{EvalMod}}$. Let* ct *encrypt $\vec{m}$ at level 0 with $\|\vec{m} + \vec{e}\|_\infty < q_0/2$, let $\|\vec{e}_{\mathsf{ModRaise}}\|_\infty \leq B_{\mathsf{ModRaise}}$ after* ModRaise*, and let* EvalMod *be accurate on the input interval set by B_{ModRaise}. Then* $\mathsf{Boot}^{\mathsf{CI\text{-}CKKS}}$ *returns a ciphertext at level $L - 2\log_2 N - \ell_{\mathsf{EvalMod}}$, where L is the level after* ModRaise *and ℓ_{EvalMod} is the number of levels consumed by* EvalMod*, that encrypts $\vec{m}$ up to the errors of the transforms, of* EvalMod*, and of key switching. No step uses a conjugation key.*

Algorithm 3 gives $\mathsf{Boot}^{\mathsf{CI\text{-}CKKS}}$.

$\mathsf{Boot}^{\mathsf{CI\text{-}CKKS}}(\mathsf{ct}, \mathsf{evk}, \{\mathsf{rk}_k\}_{k\in J_{\mathsf{rot}}}) \to \mathsf{ct}'$

Input:
- $\mathsf{ct} \in (\mathfrak{R}^{\mathsf{CI},X}_{q_0,\{X\}})^2$ under $\vec{s}$, encrypting $\vec{m}$
- relinearization key evk (products inside EvalMod)
- rotation keys rk_k, $k \in J_{\mathsf{rot}}$; no conjugation key

Output: ct' at level $L - 2\log_2 N - \ell_{\mathsf{EvalMod}}$ (L: level after ModRaise), encrypting $\vec{m}$ up to a small error

1 $\mathsf{ct} \leftarrow \mathsf{ModRaise}(\mathsf{ct})$ — *modulus $q_0 \to Q$, overflow $q_0\vec{e}_{\mathsf{ModRaise}}$*
2 $\mathsf{ct} \leftarrow \mathsf{CtS}^*(\mathsf{ct}, \{\mathsf{rk}_k\})$
3 $\mathsf{ct} \leftarrow \mathsf{EvalMod}(\mathsf{ct}, \mathsf{evk})$ — *one real run for all N slots*
4 $\mathsf{ct}' \leftarrow \mathsf{StC}^*(\mathsf{ct}, \{\mathsf{rk}_k\})$
5 **return** ct'

Algorithm 3. CI-CKKS bootstrapping

By Proposition 1, the two transforms of $\mathsf{Boot}^{\mathsf{CI\text{-}CKKS}}$ take $2\log_2 N$ levels in the radix-2 version (fewer if stages are merged), and $\mathsf{Boot}^{\mathsf{CI\text{-}CKKS}}$ uses $2(3\log_2 N - 3)$ rotations, the relinearizations of one EvalMod, $2\log_2 N - 2$ rotation keys, and one relinearization key.

4. Conjugate-Invariant GL

GL packs $\phi(p)$ complex matrices into one plaintext, so half of its plaintext and key space carries imaginary parts that a real computation does not use. In this section, we apply the conjugate-invariant fold of Section 2.4 to the three variables of GL. Section 4.1 shows that four folded rings have both real slots and a closed matrix product, three of which define schemes, and Section 4.2 to Section 4.5 describe the one we develop, CI^3-GL.

4.1. Fold Groups and Three Variants

The folds are best described within a single ambient ring. Let $n \geq 8$ be a power of two, $N := \phi(n)/2 = n/4$ as in Section 2.4, and let p be odd and squarefree. The ambient ring is

$$\mathcal{R}_{n,n,p} := \mathbb{Z}[X, Y, W] / \langle \Phi_n(X), \Phi_n(Y), \Phi_p(W) \rangle,$$

of rank $\phi(n)^2\phi(p)$ over $\mathbb{Z}$. As in GL, X indexes rows, Y columns, and W the batch. Let $\sigma_X : X \mapsto X^{-1}$, and define σ_Y and σ_W likewise. They generate a group $\{1, \sigma_X\} \times \{1, \sigma_Y\} \times \{1, \sigma_W\} \cong (\mathbb{Z}/2)^3$ of automorphisms.

Definition 4 (fold group, folded ring)**.** A fold group is a subgroup H_F of $\langle \sigma_X, \sigma_Y, \sigma_W \rangle$. Its folded ring $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{F\}}$ is the subring of $\mathcal{R}_{n,n,p}$ fixed by every element of H_F, where F lists generators of H_F by the variables they invert together (e.g., $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{W,XY\}}$ is the folded ring of $H_F = \langle \sigma_W, \sigma_X\sigma_Y \rangle$: W stands for σ_W and XY for $\sigma_X\sigma_Y$). The slots of $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{F\}}$ are its values at (x, y, w), with x, y primitive n-th and w primitive p-th roots of unity.

Proposition 2 (rank and slots of a folded ring). *(i) $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{F\}}$ is a free $\mathbb{Z}$-module of rank $\phi(n)^2\phi(p)/|H_F|$. (ii) The slots of $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{F\}}$ are all real if and only if $\sigma_X\sigma_Y\sigma_W \in H_F$. (iii) Exactly five fold groups contain $\sigma_X\sigma_Y\sigma_W$.*

The matrix product of two elements of a folded ring, which contracts a third variable Z, is not always in that ring.

Proposition 3 (closure under the matrix product). *Let $n \geq 4$ be a power of two and $p \geq 3$, let Z be a third variable with $\Phi_n(Z) = 0$, and let $[Z^0]$ take the coefficient of Z^0 in the basis $\{Z^c\}_{0\leq c<2N}$. For $f, h \in \mathcal{R}_{n,n,p}$, let $f \circledast h := \mathsf{Trace}_Z\big(f(X,Z,W)\,h(Y,Z,W)\big)$ with $\mathsf{Trace}_Z := N[Z^0]$; this extends $\circledast$ of Definition 8 (where $d = N$) from $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,Y,W\}}$ to $\mathcal{R}_{n,n,p}$, with the same operand order, and for $f, h \in \mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,Y,W\}}$ with tensors A, B the slot matrices of $f \circledast h$ are $M_A^{(\kappa)} M_B^{(\kappa)\mathsf{T}}$ (Theorem 2). (i) $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{XYW\}}$, $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{W,XY\}}$ and $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,Y,W\}}$ are closed under $\circledast$. (ii) $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,YW\}}$ is closed under $\circledast$, and $f \circledast h \in \mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,Y,W\}}$ for all $f, h \in \mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,YW\}}$. (iii) $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{Y,XW\}}$ is not closed under $\circledast$: 1 and $h := XW + X^{-1}W^{-1}$ lie in it, and $1 \circledast h = N(YW + Y^{-1}W^{-1})$ does not. Its products lie in $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{XYW\}}$, and not all of them in the folded ring of any larger fold group. (i)–(iii) hold with Trace_Z replaced by $c[Z^0]$ for any nonzero integer c, with N replaced by c in (iii). With the other operand order, $\mathsf{Trace}_Z\big(f(X,Z,W)\,h(Z,Y,W)\big)$ (slot matrices $M_A^{(\kappa)} M_B^{(\kappa)}$ on $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,Y,W\}}$), all five rings of (i)–(iii) are closed.*

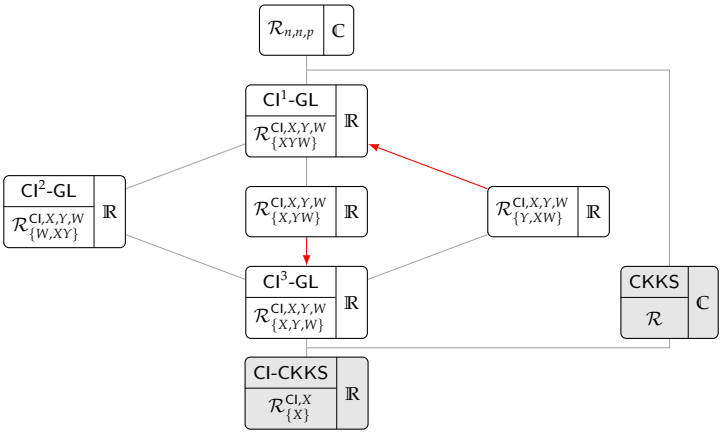


Figure 1. Inclusions among the subrings of $\mathcal{R}_{n,n,p}$. A white box offers homomorphic matrix product, and a gray box does not. A line joins a ring (below) to a ring that contains it (above). A red arrow shows the ring where results of matrix products land when it is either a bigger or smaller one. Each box contains either $\mathbb{R}$ or $\mathbb{C}$ to indicate the type of the slots.

From now on, we name each ring by its scheme: $\mathcal{R}^{\mathsf{CI\text{-}CKKS}} := \mathcal{R}^{\mathsf{CI},X}_{\{X\}}$, $\mathcal{R}^{\mathsf{CI^1\text{-}GL}} := \mathcal{R}^{\mathsf{CI},X,Y,W}_{\{XYW\}}$, $\mathcal{R}^{\mathsf{CI^2\text{-}GL}} := \mathcal{R}^{\mathsf{CI},X,Y,W}_{\{W,XY\}}$, and $\mathcal{R}^{\mathsf{CI^3\text{-}GL}} := \mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,Y,W\}}$; $\tilde{\mathcal{R}}$ denotes the key ring of a scheme, e.g., $\tilde{\mathcal{R}}^{\mathsf{CI^3\text{-}GL}} = \tilde{\mathcal{R}}^{\mathsf{CI},X,Y,W}_{\{X,Y,W\}}$, and for CI-CKKS the key ring is $\mathcal{R}^{\mathsf{CI\text{-}CKKS}}$ itself. All these rings are subrings of $\mathcal{R}_{n,n,p} = \mathbb{Z}[X,Y,W]/\langle \Phi_n(X), \Phi_n(Y), \Phi_p(W)\rangle$. The GL ring $\mathcal{R}^{\mathsf{GL}} = \mathbb{Z}[\mathsf{i}][X, Y, W]/\langle X^d - \mathsf{i},\ Y^d - \mathsf{i},\ \Phi_p(W)\rangle$ of Section 2.3 is not one of them: eliminating $\mathsf{i} = X^d$ gives $\mathcal{R}^{\mathsf{GL}} \cong \mathbb{Z}[X,Y,W]/\langle X^{2d} + 1,\ Y^d - X^d,\ \Phi_p(W)\rangle$, which for $n = 4d$ (so $\Phi_n(X) = X^{2d} + 1$) is the quotient of $\mathcal{R}_{n,n,p}$ by $Y^d - X^d$.

Remark 1 ($\mathcal{R}_{n,n,p}$ and two copies of $\mathcal{R}^{\mathsf{GL}}$). For $n = 4d$ and an odd modulus q, reducing modulo $Y^d - X^d$ and modulo $Y^d + X^d$ gives a ring isomorphism $\mathcal{R}_{n,n,p,q} \cong \mathcal{R}^{\mathsf{GL}}_q \times \mathcal{R}^{\mathsf{GL}}_q$,

where the second factor is the GL ring written with $Y^d = -X^d$ (isomorphic to $\mathcal{R}^{\mathsf{GL}}$ by $Y \mapsto X^2Y$). An RLWE ciphertext over $\mathcal{R}_{n,n,p}$ is therefore a pair of RLWE ciphertexts over $\mathcal{R}^{\mathsf{GL}}$ with the same number of coefficients, and additions, multiplications, NTTs, key switching, and the automorphisms $X \mapsto X^a$, $Y \mapsto Y^b$, $W \mapsto W^c$ with $a \equiv b \pmod 4$ act on the two copies separately. The matrix product of $\mathcal{R}_{n,n,p}$ does not: it mixes the two copies. The two GL secret keys obtained from one key are equal modulo 2, not independent; automorphisms with $a \not\equiv b \pmod 4$ swap the two copies; and over $\mathbb{Z}$ the image of $\mathcal{R}_{n,n,p}$ is a proper subring of $\mathcal{R}^{\mathsf{GL}} \times \mathcal{R}^{\mathsf{GL}}$, of index $2^{2d^2\phi(p)}$.

Figure 1 shows the inclusions between these subrings, with the five real-slot rings of Proposition 2, four of them closed under the matrix product by Proposition 3. Three of them define the schemes CI^k-GL, where k is the number of independent inversions that generate the fold group (Section 1). In every scheme, d denotes the matrix size: $d = 2N$ for CI^1-GL and $d = N$ for CI^2-GL and CI^3-GL, hence equal d means different n.

1. CI^1-GL uses $H_F = \{1, \sigma_X\sigma_Y\sigma_W\}$, that is, a single joint fold. Its key ring is the fixed ring $\tilde{\mathcal{R}}^{\mathsf{CI}^1\text{-GL}}$ of the conjugation $(X, W) \mapsto (X^{-1}, W^{-1})$ of MHEAAN [CKY18], of rank $N(p-1)$ for p prime. It holds $(p-1)/2$ real $d \times d$ matrices with $d = \phi(n) = 2N$. Its d RLWE ciphertexts are not the axis-2 slices but the components in a tuple basis.
2. CI^2-GL uses $H_F = \{1, \sigma_W, \sigma_X\sigma_Y, \sigma_X\sigma_Y\sigma_W\}$: a separate fold of W and a joint fold of (X, Y). Its key ring is that of CI^3-GL, of rank $N(p-1)/2$, and for prime p it holds $p-1$ real $N \times N$ matrices; a ciphertext consists of $2d$ RLWE ciphertexts, with $d = N$.
3. CI^3-GL uses the whole group: three separate folds. Its key ring has rank $N(p-1)/2$, it holds $(p-1)/2$ real $d \times d$ matrices with $d = N$, and its d RLWE ciphertexts are the axis-2 slices, as in GL.

The details of CI^1-GL and CI^2-GL are left to a later version; we focus on CI^3-GL in the rest of the paper. Of the other two real-slot rings, $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,YW\}}$ is closed under $\circledast$, with products in $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{X,Y,W\}}$, and $\mathcal{R}^{\mathsf{CI},X,Y,W}_{\{Y,XW\}}$ is not (Proposition 3); we do not develop schemes on them.

4.2. CI^3-GL: *Rings, Slots, and Encoding*

CI^3-GL needs a real batch axis in addition to the real row and column axes. In Section 4 and *Bootstrapping* CI^3-GL, p is an odd prime unless stated otherwise; composite squarefree p is treated in Section B.

Definition 5 (CI^3-GL rings)**.** Let $\lambda_u := W^u + W^{-u}$ and $\mathcal{R}^{\mathsf{CI},W}_{\{W\}} := \{f \in \mathbb{Z}[W]/\langle\Phi_p(W)\rangle \mid f(W) = f(W^{-1})\}$, the batch analog of $\mathcal{R}^{\mathsf{CI},X}_{\{X\}}$. Write $\mathcal{R}^{\mathsf{CI},X}_{\{X\}}$ and $\mathcal{R}^{\mathsf{CI},Y}_{\{Y\}}$ for $\mathcal{R}^{\mathsf{CI},X}_{\{X\}}$ in the variables X and Y, and $\mathcal{R}^{\mathsf{CI},V}_{\{V\}}$ for $\mathcal{R}^{\mathsf{CI},W}_{\{W\}}$ in the variable V. The plaintext ring and the key ring are

$$\mathcal{R}^{\mathsf{CI}^3\text{-GL}} := \mathcal{R}^{\mathsf{CI},X}_{\{X\}} \otimes \mathcal{R}^{\mathsf{CI},Y}_{\{Y\}} \otimes \mathcal{R}^{\mathsf{CI},W}_{\{W\}}, \qquad \tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}} := \mathcal{R}^{\mathsf{CI},X}_{\{X\}} \otimes \mathcal{R}^{\mathsf{CI},W}_{\{W\}},$$

and the matrix size is $d = N$.

Lemma 8 (bases and ranks)**.** *For p an odd prime, $\{\lambda_u\}_{1 \le u \le (p-1)/2}$ is a $\mathbb{Z}$-basis of $\mathcal{R}^{\mathsf{CI},W}_{\{W\}}$, which is the subring of $\mathbb{Z}[W]/\langle\Phi_p(W)\rangle$ fixed by σ_W, and $1 = -\sum_{1 \le u \le (p-1)/2} \lambda_u$. Hence $\mathcal{R}^{\mathsf{CI}^3\text{-GL}}$ is the folded ring of the whole group of Section 4.1, with $\mathbb{Z}$-basis $\{c_a(X)c_k(Y)\lambda_u(W)\}$, and $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ is the subring of $\tilde{\mathcal{R}}_{n,n,p} := \mathbb{Z}[X, W]/\langle\Phi_n(X), \Phi_p(W)\rangle$ fixed by σ_X and σ_W separately, with $\mathbb{Z}$-basis $\{c_a(X)\lambda_u(W)\}$ and rank $r_{\mathsf{CI}^3\text{-GL}} := N(p-1)/2$.*

The key ring $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ differs from the key ring $\tilde{\mathcal{R}}^{\mathsf{CI}^1\text{-GL}}$ of CI^1-GL, which is fixed only by the joint inversion of X and W and has twice the rank. Every element of $\mathcal{R}^{\mathsf{CI}^3\text{-GL}}$ is invariant under each of σ_X, σ_Y, and σ_W, so there is no conjugation to implement.

With $\zeta := e^{2\pi \mathrm{i}/n}$, $\eta := e^{2\pi \mathrm{i}/p}$, and γ a primitive root mod p (Section 2.3), the slots of $m \in \mathcal{R}^{\mathsf{CI}^3\text{-GL}}$ are $m(\zeta^{5^j}, \zeta^{5^k}, \eta^{\gamma^\kappa})$, $0 \le j,k < N$, $0 \le \kappa < \mathfrak{k}$, with $\mathfrak{k} := \phi(p)/2$ the number of batch slots ($\mathfrak{k} = (p-1)/2$ for p prime). They are real (Proposition 2), and the batch points take one point from each pair $\{\eta^x, \eta^{-x}\}$ (Section B). Batch slot κ holds the real $d \times d$ matrix $M^{(\kappa)}$ with entries $m(\zeta^{5^j}, \zeta^{5^k}, \eta^{\gamma^\kappa})$.

As in Section 2.3, algorithms act on integer tensors. An element of $\mathcal{R}^{\mathsf{CI}^3\text{-GL}}$ has a unique tensor $C \in \mathfrak{R}^{\mathsf{CI}^3\text{-GL}} := \mathbb{Z}^{N\times N\times \mathfrak{k}}$, with $C[a,k,u]$ the coefficient of $c_a(X)\, c_k(Y)\, \lambda_u(W)$ and c_a the a-th entry of $\vec{c}_n^{\mathsf{CI}}$. Axis 3 is labeled by $1 \le u \le \mathfrak{k}$, with label u stored at position $u-1$; an index u with $1 \le u \le \mathfrak{k}$ (as in $C[a,k,u]$, $D_W[\kappa,u]$) is a label, an index $u < \mathfrak{k}$ is a position, holding λ_{u+1}. The map $\mathsf{Expand} : \mathfrak{R}^{\mathsf{CI}^3\text{-GL}} \to \mathcal{R}^{\mathsf{CI}^3\text{-GL}}$ sends a tensor C to $\mathsf{Expand}(C) := \sum_{0\le a,k<N} \sum_{1\le u\le \mathfrak{k}} C[a,k,u]\, c_a(X)\, c_k(Y)\, \lambda_u(W)$, where $c_0 = 1$ and $c_a = X^a + X^{-a}$ for $a \ge 1$ (the CI basis of Section 2.4, the same in Y), and $\lambda_u = W^u + W^{-u}$: it returns the element whose coordinates in this basis are the entries of C. It is a bijection, and we write $\mathsf{Compact}$ for its inverse. With entrywise addition and the product $C \cdot C' := \mathsf{Compact}(\mathsf{Expand}(C)\,\mathsf{Expand}(C'))$, $\mathfrak{R}^{\mathsf{CI}^3\text{-GL}}$ is a commutative ring and Expand is a ring isomorphism. Modulo q, we write $\mathcal{R}^{\mathsf{CI}^3\text{-GL}}{}_q$ and $\mathfrak{R}^{\mathsf{CI}^3\text{-GL}}{}_q := \mathbb{Z}_q^{N\times N\times \mathfrak{k}}$. The elements of $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ are the tensors with $C[a,k,u] = 0$ for $k \ne 0$. For a tensor T, $T[:,k,:]$ is its axis-2 slice at k.

Decoding uses one matrix per axis. Along X and Y, it is D_n of Section 2.4, with $D_n^{-1} = \Lambda_n D_n^{\mathsf{T}}$. Along W, it is $D_W \in \mathbb{R}^{(p-1)/2\times(p-1)/2}$, $D_W[\kappa,u] := \lambda_u(\eta^{\gamma^\kappa}) = 2\cos(2\pi u\gamma^\kappa/p)$, $1 \le u \le (p-1)/2$. Unlike the columns of D_n, its columns are not orthogonal, so its inverse involves the Gram matrix.

Lemma 9 (batch decoding matrix)**.** $D_W^{\mathsf{T}} D_W = p\,\mathsf{Id} - 2\,\vec{1}\vec{1}^{\mathsf{T}}$ *and* $D_W^{-1}[u,\kappa] = \big(2\cos(2\pi u\gamma^\kappa/p) - 2\big)/p$. *For $p \ge 5$, the singular values of D_W are 1 and $\sqrt{p}$, so* $\mathsf{cond}(D_W) = \sqrt{p}$.

Definition 6 (encoding and decoding)**.** For $M \in \mathbb{R}^{N\times N\times \mathfrak{k}}$, whose κ-th matrix is $M^{(\kappa)} = M[:,:,\kappa]$, and a scaling factor Δ,

$$\mathsf{Encode}^{\mathsf{CI}^3\text{-GL}}(M,\Delta) := \lfloor \Delta\, M \times_1 D_n^{-1} \times_2 D_n^{-1} \times_3 D_W^{-1} \rceil,$$
$$\mathsf{Decode}^{\mathsf{CI}^3\text{-GL}}(\mathsf{PT},\Delta) := \Delta^{-1}\,\mathsf{PT} \times_1 D_n \times_2 D_n \times_3 D_W,$$

with entrywise rounding.

Lemma 10. $\mathsf{Decode}^{\mathsf{CI}^3\text{-GL}}(\mathsf{PT},\Delta)[j,k,\kappa] = \Delta^{-1}\mathsf{Expand}(\mathsf{PT})(\zeta^{5^j}, \zeta^{5^k}, \eta^{\gamma^\kappa})$, *and* $\mathsf{Decode}^{\mathsf{CI}^3\text{-GL}}(\mathsf{Encode}^{\mathsf{CI}^3\text{-GL}}(M,\Delta),\Delta) = M + \Delta^{-1}R \times_1 D_n \times_2 D_n \times_3 D_W$ *where R is the rounding error, with entries in $[-1/2, 1/2]$.*

Everything else that GL does on coefficients (rounding, rescaling, and RNS decomposition) acts entrywise on these integer tensors, with no imaginary parts.

4.3. Ciphertexts and Keys

As in GL, a CI^3-GL ciphertext is a collection of d RLWE ciphertexts under one key.

Definition 7 (CI^3-GL ciphertext)**.** A ciphertext at level ℓ is a pair $(a, b) \in (\mathcal{R}_{Q_\ell}^{\mathsf{CI}^3\text{-GL}})^2$ with

$$as + b = m + e \pmod{Q_\ell},$$

for the secret key $s \in \tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$, the message m, and a small error e. It is stored as $(A, B) := (\mathsf{Compact}(a), \mathsf{Compact}(b))$, and, with S, PT, E the tensors of s, m, e, we have $A \cdot S + B = \mathsf{PT} + E \pmod{Q_\ell}$.

Lemma 11 (slices)**.** *Slice k of $A \cdot S + B$ is $A[:,k,:] \cdot S[:,0,:] + B[:,k,:]$, a product in $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$. The slices $(A[:,k,:], B[:,k,:])$, $0 \le k < d$, are therefore d RLWE ciphertexts over $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ under s, and the k-th one encrypts $\mathsf{PT}[:,k,:]$, the coefficient of $c_k(Y)$ in m. More generally, a product with an element of $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ acts slice by slice.*

Key generation, encryption, and decryption are those of GL [GL25], with the key ring $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$, $\tilde{\mathcal{R}}_q^{\mathsf{CI}^3\text{-GL}} := \tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}/q\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ and the distributions $\mathcal{D}_s$, $\mathcal{D}_e$, $\mathcal{D}_v$ taken on $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$. Errors and randomness of a ciphertext are sampled per slice.

1. $\mathsf{KeyGen}^{\mathsf{CI}^3\text{-GL}}(Q_L, P)$ samples $s \leftarrow \mathcal{D}_s$, a uniform $a_{\mathsf{pk}} \in \tilde{\mathcal{R}}_{Q_L}^{\mathsf{CI}^3\text{-GL}}$ and $e_{\mathsf{pk}} \leftarrow \mathcal{D}_e$, and sets $\mathsf{pk} := (a_{\mathsf{pk}}, -a_{\mathsf{pk}}s + e_{\mathsf{pk}} \bmod Q_L)$. A key-switching key from s' to s is $\mathsf{ksk}_{s'\to s} := (\mathsf{ksk}_r)_{0\le r<\mathsf{dnum}}$, $\mathsf{ksk}_r := (\tilde{a}_r,\ -\tilde{a}_r s + \tilde{e}_r + Pg_r s' \bmod Q_L P)$, with dnum, g_r and P as in Section 2.3. For a big source key s' (one with Y), $\tilde{a}_r$ is uniform in $\mathcal{R}^{\mathsf{CI}^3\text{-GL}}{}_{Q_L P}$ and $\tilde{e}_r = \sum_{k<N} \tilde{e}_{r,k} c_k(Y)$ with $\tilde{e}_{r,k} \leftarrow \mathcal{D}_e$. For a small source key (in $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$), $\tilde{a}_r$ and $\tilde{e}_r$ lie in $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$. The product keys are $\mathsf{evk}_\circledast := (\mathsf{ksk}_{s^\mathsf{T}\to s}, \mathsf{ksk}_{ss^\mathsf{T}\to s})$, both big, with $s^\mathsf{T} := s(Y, W)$.
2. $\mathsf{Encrypt}_{\mathsf{sk}}^{\mathsf{CI}^3\text{-GL}}(m, s, \ell)$ samples a uniform $a \in \mathcal{R}^{\mathsf{CI}^3\text{-GL}}{}_{Q_\ell}$ and $e = \sum_{k<N} e_k c_k(Y)$, $e_k \leftarrow \mathcal{D}_e$, and returns $(a, -as + m + e \bmod Q_\ell)$.
3. $\mathsf{Encrypt}_{\mathsf{pk}}^{\mathsf{CI}^3\text{-GL}}(m, \mathsf{pk}, \ell)$ samples $v = \sum_{k<N} v_k c_k(Y)$, $e_\mathsf{a} = \sum_{k<N} e_{\mathsf{a},k} c_k(Y)$ and $e_\mathsf{b} = \sum_{k<N} e_{\mathsf{b},k} c_k(Y)$, with $v_k \leftarrow \mathcal{D}_v$ and $e_{\mathsf{a},k}, e_{\mathsf{b},k} \leftarrow \mathcal{D}_e$ independent in $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$, and returns $(va_{\mathsf{pk}} + e_\mathsf{a},\ vb_{\mathsf{pk}} + m + e_\mathsf{b}) \bmod Q_\ell$.
4. $\mathsf{Decrypt}^{\mathsf{CI}^3\text{-GL}}((a, b), s)$ returns $[\mathsf{Compact}(as + b \bmod Q_\ell)]_{Q_\ell}$, which equals $\mathsf{PT} + E$ when the entries of $\mathsf{PT} + E$ lie in $[-Q_\ell/2, Q_\ell/2)$.

Each of these consists of d RLWE encryptions or decryptions over $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ under s, one per slice (Proposition 4).

Proposition 4 (correctness of encryption and decryption)**.** $\mathsf{Encrypt}_{\mathsf{sk}}^{\mathsf{CI}^3\text{-GL}}(m, s, \ell)$ *returns* (a, b) *with* $as + b = m + e \bmod Q_\ell$*, and* $\mathsf{Encrypt}_{\mathsf{pk}}^{\mathsf{CI}^3\text{-GL}}(m, \mathsf{pk}, \ell)$ *returns* (a, b) *with* $as + b = m + ve_{\mathsf{pk}} + e_\mathsf{a} s + e_\mathsf{b} \bmod Q_\ell$*.* $\mathsf{Decrypt}^{\mathsf{CI}^3\text{-GL}}$ *returns* $\mathsf{PT} + E$ *when the entries of* $\mathsf{PT} + E$ *lie in* $[-Q_\ell/2, Q_\ell/2)$*. Each algorithm consists of d RLWE encryptions or decryptions over $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ under s, one per slice.*

The key switch $\mathsf{Switch}^{\mathsf{CI}^3\text{-GL}}(\hat{C}, \mathsf{ksk}_{s'\to s})$ is the Switch of GL (Section 2.3), applied to integer entries.

Lemma 12 (key switches)**.** $\mathsf{Switch}^{\mathsf{CI}^3\text{-GL}}(\hat{C}, \mathsf{ksk}_{s'\to s})$ *returns* $(A^{\mathsf{ks}}, B^{\mathsf{ks}})$ *with* $A^{\mathsf{ks}} \cdot S + B^{\mathsf{ks}} = \hat{C} \cdot S' + E^{\mathsf{ks}} \pmod{Q_\ell}$*,* E^{ks} *as in Lemma 2.*

1. The small switch $\mathsf{Switch}_{\mathsf{small}}^{\mathsf{CI}^3\text{-GL}}$ has a source key in $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$ (its tensor is supported on

axis-2 index 0). It acts on each slice separately: d RLWE key switches with one small key.

2. The big switch $\mathsf{Switch}_{\mathsf{big}}^{\mathsf{CI^3\text{-}GL}}$ has a source key with Y, such as s^{T} or ss^{T}. It mixes slices. Its key-switching key consists of dnum $\mathsf{CI^3\text{-}GL}$ ciphertexts, that is, $d \cdot \mathsf{dnum}$ RLWE ciphertexts over $\tilde{\mathcal{R}}^{\mathsf{CI^3\text{-}GL}}$. A small key-switching key consists of dnum RLWE ciphertexts, so one big key-switching key has the size of d small ones.

4.4. Homomorphic Operations

$\mathsf{CI^3\text{-}GL}$ has the operations of GL (Section 2.3), with real constants and without conjugation. All tensor maps below (ring and matrix products, automorphisms, transpose) are $\mathbb{Z}$-linear or bilinear in the tensors, so they hold entrywise mod Q_ℓ. The structure tensors and rotation matrices they use are given in Section B.

$\mathsf{Rescale}$ is the RNS rescale of CKKS applied to each of the d RLWE ciphertexts, all dropping the same prime q_ℓ.

Rows, columns, and batch are rotated by automorphisms. The row rotation by ν applies $X \mapsto X^{5^\nu}$, i.e., $A \times_1 \Psi_\nu$ on tensors with a signed permutation Ψ_ν; it moves the key to $s(X^{5^\nu}, W)$ and needs $\mathsf{Switch}_{\mathsf{small}}^{\mathsf{CI^3\text{-}GL}}$. The batch rotation by ν applies $W \mapsto W^{\gamma^\nu}$, i.e., $A \times_3 \Psi_\nu^W$ with a permutation Ψ_ν^W, and needs a small switch from $s(X, W^{\gamma^\nu})$. The column rotation, $A \times_2 \Psi_\nu$, keeps s and needs no key switch.

Proposition 5 (rotations)**.** *On slots, the row, column, and batch rotations by ν cyclically shift j, k, or κ by ν, modulo N, N, and $(p-1)/2$, respectively. After a small switch (row, batch) or none (column), the result encrypts the rotated plaintext under s.*

Swapping X and Y is an automorphism of $\mathcal{R}^{\mathsf{CI^3\text{-}GL}}$; on tensors, it is $T \mapsto T^{\mathsf{T}}$ (axes 1 and 2 swapped). Applied to a ciphertext, the swap changes the key from s to $s^{\mathsf{T}} := s(Y, W)$, a big key, so the homomorphic transpose needs one big key switch back to s:

$$\mathsf{Transpose}^{\mathsf{CI^3\text{-}GL}}((A,B), \mathsf{ksk}_{s^{\mathsf{T}}\to s}) := (0, B^{\mathsf{T}}) + \mathsf{Switch}_{\mathsf{big}}^{\mathsf{CI^3\text{-}GL}}(A^{\mathsf{T}}, \mathsf{ksk}_{s^{\mathsf{T}}\to s}).$$

The slots are real, so the conjugate transpose of GL coincides with this transpose, and its key is the first key of $\mathsf{evk}_{\circledast}$.

Proposition 6 (transpose)**.** *If (A, B) encrypts PT under S with error E, then* $\mathsf{Transpose}^{\mathsf{CI^3\text{-}GL}}$ *returns an encryption of PT^{T} under S with error $E^{\mathsf{T}} + E^{\mathsf{ks}}$; its slot matrices are $M^{(\kappa)\mathsf{T}}$.*

The matrix product of GL contracts a third variable via a trace.

Definition 8 (matrix product)**.** As in Section 2.3, we use the trace $\mathsf{Trace}_Z = d[Z^0]$ of Crockett et al. [CGK+26] on a third copy of $\mathcal{R}_{\{X\}}^{\mathsf{CI},X}$ in a variable Z: Trace_Z is the field trace in Z, the sum of the N real evaluations at ξ_l, $0 \le l < N$, extended linearly over the other variables. For $f, h \in \mathcal{R}^{\mathsf{CI^3\text{-}GL}}$,

$$f \circledast h := \mathsf{Trace}_Z\big(f(X,Z,W)\, h(Y,Z,W)\big),$$

which equals GL's $\mathsf{Trace}_Z\big(f(X,Z,W)\, \bar{h}(Y^{-1},Z^{-1},W^{-1})\big)$ because h has real coefficients and is invariant under each inversion. Unlike in GL, no operand is flipped.

Theorem 2 (matrix product)**.** *Let $f, h \in \mathcal{R}^{\mathsf{CI^3\text{-}GL}}$ have tensors A, B and slot matrices $M_A^{(\kappa)}$,*

$M_B^{(\kappa)}$ *(at* $\Delta = 1$*). Then* $f \circledast h \in \mathcal{R}^{\mathsf{CI^3\text{-}GL}}$*, its slot matrices are* $M_A^{(\kappa)} M_B^{(\kappa)\mathsf{T}}$*, and its tensor is*

$$(A \circledast B)[a,k,w] = N \sum_{0 \le v < N} (2 - \delta_{v,0}) \sum_{u,u'} \Upsilon^W[u,u',w]\, A[a,v,u] B[k,v,u'],$$

where $\Upsilon^W[u,u',w]$ *is the coefficient of* λ_w *in* $\lambda_u \lambda_{u'}$*.*

Ring products (the slot-wise product $\cdot$, and the products $\hat{C}_r \cdot \mathsf{ksk}_r$ inside key switching) are computed with the number-theoretic transform (NTT) of the ambient ring $\mathcal{R}_{n,n,p}$: the inputs are first rewritten from the CI bases to monomials ($c_a = X^a - X^{2N-a}$, $\lambda_u = W^u + W^{p-u}$), in linear time, and the modulus satisfies $q \equiv 1 \pmod{n}$ and $q \equiv 1 \pmod{p}$. The product $\circledast$ is not a ring product: it contracts the variable Z, so it does not use this NTT; its cost is given next.

Proposition 7 (cost of $\circledast$)**.** *Over* $\mathbb{Z}$*,* $\circledast$ *takes one GEMM (general matrix–matrix multiplication, here a product of two* $d \times d$ *matrices) per pair* (u,u') *of batch coordinates. Over* $\mathbb{Z}_q$ *with* $q \equiv 1 \pmod{n}$ *and* $q \equiv \pm 1 \pmod{p}$*, a transform of the batch axis mod q splits* $\circledast$ *into one contraction per batch slot, and one* $\circledast$ *costs* $\phi(p)/2$ *GEMMs in* $\mathbb{Z}_q^{d \times d}$ *per RNS limb.*

The key factors out of the trace, so a product with a plaintext needs no key switch. For a ciphertext (A,B) and a plaintext $\mathsf{PT} \in \mathbb{Z}^{N \times N \times \mathfrak{k}}$,

$$\mathsf{MatMultPt}^{\mathsf{CI^3\text{-}GL}}((A,B),\mathsf{PT}) := \mathsf{Rescale}(A \circledast \mathsf{PT},\ B \circledast \mathsf{PT}).$$

Corollary 1 (ct–pt product)**.** *Let* (A,B) *be at level* ℓ *and scale* Δ*, and let* PT *be at scale* Δ'*. Then* $(A \circledast \mathsf{PT}) \cdot S + (B \circledast \mathsf{PT}) = (A \cdot S + B) \circledast \mathsf{PT}$ *exactly, and* $\mathsf{MatMultPt}^{\mathsf{CI^3\text{-}GL}}((A,B),\mathsf{PT})$ *is a ciphertext under S at level* $\ell - 1$ *and scale* $\Delta\Delta'/q_\ell$ *with the* κ*-th matrix of slots* $M^{(\kappa)} M_{\mathsf{PT}}^{(\kappa)\mathsf{T}}$*, up to the rescaling error.*

A product of two ciphertexts produces four tensors under four keys; the two tensors under big keys are switched back to s (Algorithm 4).

Proposition 8 (ct–ct product)**.** *For ciphertexts* (A,B) *and* (A',B') *of* PT *and* PT' *under S, with errors E, E′,*

$$\begin{aligned} B \circledast B' + (A \circledast B') \cdot S + (B \circledast A') \cdot S^{\mathsf{T}} + (A \circledast A') \cdot (S \cdot S^{\mathsf{T}}) \\ \equiv (\mathsf{PT} + E) \circledast (\mathsf{PT}' + E') \pmod{Q_\ell}, \end{aligned}$$

and $\mathsf{MatMult}^{\mathsf{CI^3\text{-}GL}}$ *returns an encryption of* $(\mathsf{PT} + E) \circledast (\mathsf{PT}' + E')$ *under S at level* $\ell - 1$*, up to the errors of the two big switches and of the rescale.*

The order of the second and third components matters: GL [GL25] (its Section 3.5) lists $b_u \circledast a_v$ and $a_u \circledast b_v$ in the opposite order and switches $a_u \circledast b_v$ with the key of $\bar{s}(Y^{-1}, W^{-1})$, and with that order the product does not decrypt correctly.

$\mathsf{MatMult}^{\mathsf{CI^3\text{-}GL}}(\mathsf{ct}, \mathsf{ct}', \mathsf{evk}_{\circledast}) \to \mathsf{ct}''$

Input:
- $\mathsf{CI^3\text{-}GL}$ ciphertexts $\mathsf{ct} = (A, B)$, $\mathsf{ct}' = (A', B')$ in $(\mathfrak{R}^{\mathsf{CI^3\text{-}GL}}{}_{Q_\ell})^2$, of PT and PT' under S
- $\mathsf{evk}_{\circledast} = (\mathsf{ksk}_{s^\mathsf{T}\to s}, \mathsf{ksk}_{ss^\mathsf{T}\to s})$, both big

Output: $\mathsf{CI^3\text{-}GL}$ ciphertext ct'' at level $\ell - 1$ under S of $\mathsf{PT} \circledast \mathsf{PT}'$ up to a small error; slots $M^{(\kappa)}_{\mathsf{PT}} M^{(\kappa)\mathsf{T}}_{\mathsf{PT}'}$

1 $T_0 \leftarrow B \circledast B'$, $T_1 \leftarrow A \circledast B'$, $T_2 \leftarrow B \circledast A'$, $T_3 \leftarrow A \circledast A'$, all in $\mathfrak{R}^{\mathsf{CI^3\text{-}GL}}{}_{Q_\ell}$ *under* $1, S, S^\mathsf{T}, S \cdot S^\mathsf{T}$
2 $(A_2, B_2) \leftarrow \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}(T_2, \mathsf{ksk}_{s^\mathsf{T}\to s})$
3 $(A_3, B_3) \leftarrow \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}(T_3, \mathsf{ksk}_{ss^\mathsf{T}\to s})$
4 $\mathsf{ct}'' \leftarrow \mathsf{Rescale}\big(T_1 + A_2 + A_3,\ T_0 + B_2 + B_3\big)$
5 **return** ct''

Algorithm 4. $\mathsf{CI^3\text{-}GL}$ ciphertext–ciphertext matrix product

The product uses four $\circledast$ products, that is, $4 \cdot \phi(p)/2$ GEMMs per limb, and two big switches.

The ring product $\cdot$ multiplies slots entrywise. On ciphertexts it is the RLWE product followed by relinearization with $\mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{small}}$ and $\mathsf{ksk}_{s^2\to s}$, and on tensors it is

$$(A \cdot B)[a'', k'', u''] = \sum_{a,a',k,k',u,u'} \mathrm{Y}^X[a, a', a'']\mathrm{Y}^X[k, k', k'']\mathrm{Y}^W[u, u', u'']A[a, k, u]B[a', k', u'],$$

where $\mathrm{Y}^X[a, a', a'']$ is the coefficient of $c_{a''}$ in $c_a c_{a'}$ (the same tensor serves axis 2) and $\mathrm{Y}^W[u, u', u'']$ that of $\lambda_{u''}$ in $\lambda_u \lambda_{u'}$; Section B gives their entries.

4.5. Comparison with GL

We compare the two schemes at the same matrix size d and the same number of matrices. $\mathsf{CI^3\text{-}GL}$ folds its batch axis, so it holds $\phi(p)/2$ matrices at batch index p.

Proposition 9 (sizes at the same d and number of matrices)**.** *Let $p \neq 3$ be an odd prime and $n = 4d$. The batch index $3p$ is odd and squarefree, $\phi(3p)/2 = \phi(p)$, and $\mathbb{Z}^*_{3p}/\{\pm 1\}$ is cyclic; at batch index $3p$, $\mathsf{CI^3\text{-}GL}$ has the sizes and counts of the right column of Table 2.*

Table 2 collects the differences in this setting; the GL column is taken from Section 2.3 and [GL25].

item	GL [GL25]	CI^3-GL (this work)
matrices per plaintext	$\phi(p)$ complex $d \times d$	$\phi(p)$ real $d \times d$
key ring, rank	$\tilde{\mathcal{R}}^{\mathsf{GL}}$, $2d\phi(p)$	$\tilde{\mathcal{R}}^{\mathsf{CI^3\text{-}GL}}$, $d\phi(p)$
plaintext	$2d^2\phi(p)$	$d^2\phi(p)$
ciphertext	d RLWE ciphertexts: $4d^2\phi(p)$	d RLWE ciphertexts: $2d^2\phi(p)$
small key	$4\,\mathsf{dnum}\,d\phi(p)$	$2\,\mathsf{dnum}\,d\phi(p)$
product and transpose keys	2 big keys for the product: $\mathsf{ksk}_{\bar{s}(Y^{-1},W^{-1})\to s}$, $\mathsf{ksk}_{s\,\bar{s}(Y^{-1},W^{-1})\to s}$; a transpose uses a third big key $\mathsf{ksk}_{\bar{s}(Y,W)\to s}$, or a conjugation key (small) and ConjTranspose	2 big keys: $\mathsf{ksk}_{s^\mathsf{T}\to s}$, $\mathsf{ksk}_{ss^\mathsf{T}\to s}$; no conjugation key needed
conjugation key	1 small key	none
GEMMs per $\circledast$, per limb	$2\phi(p)$ in $\mathbb{Z}_q^{d\times d}$, $q \equiv 1 \pmod{4p}$	$\phi(p)$ in $\mathbb{Z}_q^{d\times d}$, $q \equiv \pm 1 \pmod{3p}$
ct–ct matrix product	4 products, 2 big switches	4 products, 2 big switches

Table 2. GL vs. CI^3-GL at the same d and $\phi(p)$ matrices per plaintext (CI^3-GL at batch index $3p$); sizes in residues per RNS limb

Remark 2 (noise). At equal key-ring rank, the per-slot error variance of CI^3-GL is about 8 times that of GL.

5. Bootstrapping CI^3-GL

Bootstrapping a CI^3-GL ciphertext refreshes its d RLWE ciphertexts over $\tilde{\mathcal{R}}^{\mathsf{CI^3\text{-}GL}}$, just as CI-CKKS bootstrapping (Section 3) refreshes a single one. It is the composition of Section 3: ModRaise, a coefficient-to-slot transform CtS, EvalMod on the real slots of each RLWE ciphertext, and a slot-to-coefficient transform StC. The transforms CtS and StC can be evaluated by two different approaches:

- rotation bootstrapping, the approach of CKKS bootstrapping [CHK+18] and of Section 3: the transforms are linear maps on the slots of each RLWE ciphertext, evaluated with rotations and plaintext products;
- matrix-product bootstrapping, the approach of GL bootstrapping [CGK+26]: the transforms are ciphertext–plaintext matrix products between transposes, with no rotation key.

Section 5.1 applies rotation bootstrapping to all three variants. Section 5.2 gives matrix-product bootstrapping for CI^3-GL: first ModRaise, then the transforms, with their cost and correctness; EvalMod is that of Section 3.

5.1. Bootstrapping by Rotations

All three variants of Section 4.1 can be bootstrapped with rotations only, as CI-CKKS is, by reusing the transforms of Section 3.3 along X and W.

For CI^3-GL, ModRaise gives each of the d RLWE ciphertexts (the slices) its own overflow (Proposition 10), so rotation bootstrapping bootstraps each one separately (Remark 3).

Remark 3 (rotation bootstrapping of CI^3-GL). Slice k has the real slots $D_n C[:,k,:]D_W^\mathsf{T}$, so $\mathsf{CtS}^{\mathsf{rot}}$ inverts one axis at a time with $\mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{small}}$ only: the $\log_2 N$ stages of Section 3.3 along X, with the row keys $\mathsf{ksk}_{s(X^{5^k},W)\to s}$, then one batch stage along W, with the batch keys $\mathsf{ksk}_{s(X,W^{\gamma^\nu})\to s}$; $\mathsf{StC}^{\mathsf{rot}}$ runs the inverse stages in reverse order. Storing the coordinate

of λ_u in batch slot κ with $u \equiv \pm\gamma^{-\kappa} \pmod p$ makes, for p prime, the batch stage a scalar circulant on the batch slots $\vec{x} \in \mathbb{R}^{\mathfrak{k}}$ of a row: $\sum_{\nu<\mathfrak{k}} \frac{2\cos(2\pi\gamma^\nu/p)-2}{p} \mathsf{rot}_\nu(\vec{x})$ in $\mathsf{CtS}^{\mathsf{rot}}$ and $\sum_{\nu<\mathfrak{k}} 2\cos(2\pi\gamma^{-\nu}/p)\,\mathsf{rot}_\nu(\vec{x})$ in $\mathsf{StC}^{\mathsf{rot}}$, where $\mathsf{rot}_\nu(\vec{x})[\kappa] := x_{\kappa+\nu \bmod \mathfrak{k}}$, the action on slots of the batch rotation $W \mapsto W^{\gamma^\nu}$. With BSGS along W, $n_1 n_2 = \mathfrak{k}$, the batch stage costs $n_1 + n_2 - 2$ key switches and batch keys, so per RLWE ciphertext each of $\mathsf{CtS}^{\mathsf{rot}}$ and $\mathsf{StC}^{\mathsf{rot}}$ costs $3\log_2 N - 3 + n_1 + n_2 - 2$ key switches over $\log_2 N + 1$ levels (Proposition 1), and fewer levels if consecutive X stages are merged. For example, at $N = 256$ and $p = 257$ ($\mathfrak{k} = 128$), this is $21 + 22 = 43$ key switches over 9 levels, or 40 over 4 levels with merged X stages.

CI^1-GL needs one more step: its joint fold $(X, W) \mapsto (X^{-1}, W^{-1})$ does not split into an X transform and a W transform. A layout in two halves, coordinates symmetric and antisymmetric under $W \mapsto W^{-1}$, makes it separable. CtS applies a butterfly by $W \mapsto W^{-1}$, one key switch per RLWE ciphertext and direction with key $\mathsf{ksk}_{s(X,W^{-1})\to s}$, then stages along X, then a batch kernel on each half.

A CI^2-GL ciphertext consists of $2d$ RLWE ciphertexts under the key of CI^3-GL, and can be split, without any key, into two CI^3-GL ciphertexts, to be merged after bootstrapping. The slice-wise rotation route runs rotation bootstrapping on each slice; the split rotation route splits before ModRaise, runs rotation bootstrapping on both halves, and merges; the split matrix-product route splits before ModRaise and runs matrix-product bootstrapping on both halves, whose V-tuples (Section 5.2) may mix slices of both. Splitting before ModRaise matters: on an unsplit CI^2-GL ciphertext, the slots after the X stage of matrix-product bootstrapping are not integers, so EvalMod cannot run.

The rest of this section gives matrix-product bootstrapping for CI^3-GL, which needs no rotation key and uses 2 levels, 2 big and $2\lceil d/\mathfrak{k}\rceil$ auxiliary key switches per ciphertext and direction (Proposition 11), whereas rotation bootstrapping pays its count per RLWE ciphertext.

5.2. Bootstrapping by Matrix Products

Matrix-product bootstrapping starts with ModRaise.

ModRaise runs on each RLWE ciphertext, with the lift of Section 3.2. On a ciphertext (A, B) at modulus q_0, it lifts the entries of A and B to $[-q_0/2, q_0/2)$ and reads them modulo the top modulus Q.

Proposition 10 (ModRaise for CI^3-GL)**.** *Let q_0 be odd, and let (A, B) decrypt to $\mathsf{PT} + E$ modulo q_0. The raised ciphertext decrypts to $\mathsf{PT} + E + q_0 E_{\mathsf{ModRaise}} \pmod Q$ for an integer tensor E_{ModRaise}, whose slice k is the overflow of RLWE ciphertext k, an element of $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-GL}}$. The λ-coordinates of $q_0 E_{\mathsf{ModRaise}}$ are multiples of q_0; these are what* EvalMod *removes.*

Matrix-product bootstrapping replaces the rotations with ct–pt matrix products between transposes. Along X, it works on the ciphertext; along W, it works on V-tuples, where the batch axis becomes a matrix axis.

The product $\circledast$ multiplies slot matrices on the right by a transposed plaintext matrix (Theorem 2). To multiply on the left, $\mathsf{CtS}^{\mathsf{mm}}$ transposes, multiplies, and transposes back. The plaintext is $\mathsf{PT}_X \in \mathbb{Z}^{N\times N\times \mathfrak{k}}$ with slots D_n^{-1} in every batch slot κ, encoded at a working scale Δ_w. Slot matrices $M^{(\kappa)}$ become $D_n^{-1} M^{(\kappa)}$ (Theorem 3). This costs two big key switches $\mathsf{Switch}^{\mathsf{CI}^3\text{-GL}}_{\mathsf{big}}$ with $\mathsf{ksk}_{s^\mathsf{T}\to s}$ and one level.

The batch stage must apply D_W^{-1} along axis 3 of each RLWE ciphertext.

Definition 9 (V-tuple, swap, auxiliary key)**.** A V-tuple stacks $\mathfrak{k}$ RLWE ciphertexts under s along a new axis indexed by $v < \mathfrak{k}$; entry v is the coefficient of $\lambda_{v+1}(V)$ in a new variable V (label $v+1$ at position v, as on axis 3). It is a pair of tensors in $\mathbb{Z}_Q^{N\times\mathfrak{k}\times\mathfrak{k}}$ with axes (X, W, V), over $\tilde{\mathcal{R}}^{\mathsf{CI}^3\text{-}\mathsf{GL}} \otimes \mathcal{R}_{\{V\}}^{\mathsf{CI},V}$, where $\mathcal{R}_{\{V\}}^{\mathsf{CI},V}$ is the batch ring in V. The V-tuple of (A, B) number ρ, $0 \le \rho < \lceil N/\mathfrak{k} \rceil$, is $(A^{[\rho]}, B^{[\rho]})$ with $A^{[\rho]}[:, :, v] := A[:, \rho\mathfrak{k} + v, :]$ if $\rho\mathfrak{k} + v < N$ and 0 otherwise (likewise for B). A ciphertext has $\lceil N/\mathfrak{k} \rceil = \lceil d/\mathfrak{k} \rceil$ V-tuples.

The swap $W \leftrightarrow V$ plays the role of the transpose. For a tensor T, $T^{\mathsf{T}_{23}}$ swaps axes 2 and 3; on a V-tuple it moves the key to $s(X, V)$. The auxiliary key switch $\mathsf{Switch}_{\mathsf{aux}}^{\mathsf{CI}^3\text{-}\mathsf{GL}}$ is the key switch of Section 4.3 on V-tuples, with

$$\mathsf{ksk}^{\mathsf{aux}} := \mathsf{ksk}_{s(X,V)\to s(X,W)}.$$

The transpose $X \leftrightarrow Y$ does not apply after the swap, since a V-tuple has no Y; this follows the pattern of Algorithm 1 of Crockett et al. [CGK+26].

After the swap, axis 3 holds the batch coordinates, which we read in a second copy W' of the batch variable. The plaintext $\mathsf{PT}_W \in \mathbb{Z}^{\mathfrak{k}\times\mathfrak{k}}$ has as entry $\mathsf{PT}_W[v, u]$ the coefficient of $\lambda_{v+1}(V)\lambda_{u+1}(W')$ in an element with slots D_W^{-1}, encoded at scale Δ_w. The contraction is the trace in W', the field trace of $\mathbb{Q}(\eta + \eta^{-1})$, given here for p prime (for composite squarefree p, -1 becomes $\mathsf{möb}(p)$, with λ_u indexed by representatives of $\mathbb{Z}_p^*/\{\pm 1\}$ (Section B), where möb is the Möbius function, since $\sum_{u\in\mathbb{Z}_p^*} \eta^u = \mathsf{möb}(p)$ for squarefree p); for an element with coordinates $\alpha_u \in \mathbb{Z}$, $u < \mathfrak{k}$, in the basis $\{\lambda_u\}$,

$$\mathsf{Trace}_{W'}\Big(\sum_{u<\mathfrak{k}} \alpha_u \lambda_{u+1}(W')\Big) = -\sum_{u<\mathfrak{k}} \alpha_u.$$

Since s does not involve W', the trace commutes with decryption, and no key switch is needed. In coefficients, it is one product with the trace Gram matrix G_λ, $G_\lambda[u, v] := \mathsf{Trace}_{W'}(\lambda_{u+1}\lambda_{v+1})$.

Lemma 13 (batch trace)**.** *For p prime, the trace formula above holds, $G_\lambda = D_W^{\mathsf{T}} D_W = p\,\mathsf{Id} - 2\vec{1}\vec{1}^{\mathsf{T}}$ (Lemma 9), and $\mathsf{BatchMult}(\check{A}, \mathsf{PT}_W) := \check{A} \times_3 (\mathsf{PT}_W G_\lambda)$ returns the $\lambda_{v+1}(V)$-coefficients of $\mathsf{Trace}_{W'}$ of the product of its inputs.*

$\mathsf{CtS}^{\mathsf{mm}}$ (Algorithm 5) is the row stage followed by the batch stage on each V-tuple.

$\mathsf{CtS}^{\mathsf{mm}}_{\mathsf{ksk}_{s^\mathsf{T}\to s},\,\mathsf{ksk}^{\mathsf{aux}}}((A,B)) \to (A',B')$

Input:
- ciphertext $(A,B) \in (\mathbb{Z}_Q^{N\times N\times \mathfrak{k}})^2$, output of ModRaise, decrypting to $\mathsf{PT} + E + q_0 E_{\mathsf{ModRaise}}$
- $\mathsf{ksk}_{s^\mathsf{T}\to s}$: big key-switching key from $s^\mathsf{T} = s(Y,W)$ back to s, used by each transpose (the first key of $\mathsf{evk}_{\circledast}$)
- $\mathsf{ksk}^{\mathsf{aux}} = \mathsf{ksk}_{s(X,V)\to s(X,W)}$: auxiliary key-switching key, used by each swap $W \leftrightarrow V$ of a V-tuple
- $\mathsf{PT}_X \in \mathbb{Z}^{N\times N\times \mathfrak{k}}$: plaintext with slots D_n^{-1} in every batch slot, at scale Δ_w; applies D_n^{-1} along X
- $\mathsf{PT}_W \in \mathbb{Z}^{\mathfrak{k}\times\mathfrak{k}}$: plaintext with slots D_W^{-1}, at scale Δ_w; applies D_W^{-1} along W

Output: ciphertext (A',B') two levels lower; RLWE ciphertext k has the slots $(\mathsf{PT} + E + q_0 E_{\mathsf{ModRaise}})[:,k,:]/\Delta$

Context: $\lceil N/\mathfrak{k}\rceil$ V-tuples $(A^{[\rho]},B^{[\rho]}) \in (\mathbb{Z}_Q^{N\times\mathfrak{k}\times\mathfrak{k}})^2$; $M^{(\kappa)}$: the slot matrices at the input

$(A',B') \leftarrow (A,B)$
$(A',B') \leftarrow (0,B'^\mathsf{T}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}(A'^\mathsf{T}, \mathsf{ksk}_{s^\mathsf{T}\to s})$ — *slots $M^{(\kappa)\mathsf{T}}$*
$(A',B') \leftarrow \mathsf{MatMultPt}^{\mathsf{CI^3\text{-}GL}}((A',B'), \mathsf{PT}_X)$ — *slots $(D_n^{-1}M^{(\kappa)})^\mathsf{T}$; one level*
$(A',B') \leftarrow (0,B'^\mathsf{T}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}(A'^\mathsf{T}, \mathsf{ksk}_{s^\mathsf{T}\to s})$ — *slots $D_n^{-1}M^{(\kappa)}$*
for $\rho = 0,\dots,\lceil N/\mathfrak{k}\rceil - 1$ **do** — *V-tuples*
  for $v < \mathfrak{k}$ **do**
    if $\rho\mathfrak{k} + v < N$ **then**
      $A^{[\rho]}[:,:,v] \leftarrow A'[:,\rho\mathfrak{k}+v,:]$, $B^{[\rho]}[:,:,v] \leftarrow B'[:,\rho\mathfrak{k}+v,:]$
    else
      $A^{[\rho]}[:,:,v] \leftarrow 0$, $B^{[\rho]}[:,:,v] \leftarrow 0$
  $(A^{[\rho]},B^{[\rho]}) \leftarrow (0,(B^{[\rho]})^{\mathsf{T}_{23}}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{aux}}((A^{[\rho]})^{\mathsf{T}_{23}}, \mathsf{ksk}^{\mathsf{aux}})$ — *swap $W \leftrightarrow V$*
  $(A^{[\rho]},B^{[\rho]}) \leftarrow \mathsf{Rescale}(\mathsf{BatchMult}(A^{[\rho]},\mathsf{PT}_W),\ \mathsf{BatchMult}(B^{[\rho]},\mathsf{PT}_W))$ — *axis 3 read as W'; one level*
  $(A^{[\rho]},B^{[\rho]}) \leftarrow (0,(B^{[\rho]})^{\mathsf{T}_{23}}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{aux}}((A^{[\rho]})^{\mathsf{T}_{23}}, \mathsf{ksk}^{\mathsf{aux}})$ — *swap back*
  for $v < \mathfrak{k}$ with $\rho\mathfrak{k} + v < N$ **do**
    $A'[:,\rho\mathfrak{k}+v,:] \leftarrow A^{[\rho]}[:,:,v]$, $B'[:,\rho\mathfrak{k}+v,:] \leftarrow B^{[\rho]}[:,:,v]$
return (A',B')

Algorithm 5. $\mathsf{CI^3\text{-}GL}$ coefficient-to-slot transform by matrix products (matrix-product bootstrapping). A line $(0,B^\mathsf{T}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}(A^\mathsf{T}, \mathsf{ksk}_{s^\mathsf{T}\to s})$ is a homomorphic transpose: transposing A and B transposes the slot matrices but moves the key to s^T, and the key switch of A brings it back to s (B needs none). The lines with T_{23} and $\mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{aux}}$ do the same on a V-tuple: they swap W and V, and $\mathsf{ksk}^{\mathsf{aux}}$ brings the key from $s(X,V)$ back to $s(X,W)$.

After $\mathsf{CtS}^{\mathsf{mm}}$, every real slot of every RLWE ciphertext holds one λ-coordinate of $\mathsf{PT} + E + q_0 E_{\mathsf{ModRaise}}$, scaled by Δ^{-1}, so EvalMod of Section 3.4 runs on each of the d RLWE ciphertexts with real slots only, with no conjugation and no split into real and imaginary parts.

$\mathsf{StC}^{\mathsf{mm}}$ uses the plaintexts PT'_X and PT'_W with slots D_n and D_W in place of D_n^{-1} and D_W^{-1}, and runs the steps of $\mathsf{CtS}^{\mathsf{mm}}$ in reverse order; Algorithm 6 gives it in full.

$\mathsf{StC}^{\mathsf{mm}}_{\mathsf{ksk}_{s^{\mathsf{T}}\to s},\,\mathsf{ksk}^{\mathsf{aux}}}((A,B)) \to (A',B')$

Input:
- ciphertext (A,B) whose RLWE ciphertext k has the slots $\mathsf{PT}[:,k,:]/\Delta$ for a tensor PT in λ-coordinates (the output layout of $\mathsf{CtS}^{\mathsf{mm}}$)
- $\mathsf{ksk}_{s^{\mathsf{T}}\to s}$: big key-switching key from $s^{\mathsf{T}} = s(Y,W)$ back to s, used by each transpose (the first key of $\mathsf{evk}_{\circledast}$)
- $\mathsf{ksk}^{\mathsf{aux}} = \mathsf{ksk}_{s(X,V)\to s(X,W)}$: auxiliary key-switching key, used by each swap $W \leftrightarrow V$ of a V-tuple
- $\mathsf{PT}'_X \in \mathbb{Z}^{N\times N\times \mathfrak{k}}$: plaintext with slots D_n in every batch slot, at scale Δ_w; applies D_n along X
- $\mathsf{PT}'_W \in \mathbb{Z}^{\mathfrak{k}\times\mathfrak{k}}$: plaintext with slots D_W, at scale Δ_w; applies D_W along W

Output: ciphertext (A',B') two levels lower with the slots $\mathsf{PT}\times_1 D_n \times_2 D_n \times_3 D_W/\Delta$
Context: V-tuples as in $\mathsf{CtS}^{\mathsf{mm}}$

$(A',B') \leftarrow (A,B)$
for $\rho = 0,\dots,\lceil N/\mathfrak{k}\rceil - 1$ **do** *V-tuples*
  stack $(A^{[\rho]},B^{[\rho]})$ from the slices $\rho\mathfrak{k},\dots,\rho\mathfrak{k}+\mathfrak{k}-1$ of (A',B'), zero beyond N
  $(A^{[\rho]},B^{[\rho]}) \leftarrow (0,(B^{[\rho]})^{\mathsf{T}_{23}}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{aux}}((A^{[\rho]})^{\mathsf{T}_{23}},\mathsf{ksk}^{\mathsf{aux}})$ *swap $W\leftrightarrow V$*
  $(A^{[\rho]},B^{[\rho]}) \leftarrow \mathsf{Rescale}(\mathsf{BatchMult}(A^{[\rho]},\mathsf{PT}'_W),\ \mathsf{BatchMult}(B^{[\rho]},\mathsf{PT}'_W))$ *one level*
  $(A^{[\rho]},B^{[\rho]}) \leftarrow (0,(B^{[\rho]})^{\mathsf{T}_{23}}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{aux}}((A^{[\rho]})^{\mathsf{T}_{23}},\mathsf{ksk}^{\mathsf{aux}})$ *swap back*
  unstack into the slices $\rho\mathfrak{k},\dots,\min(\rho\mathfrak{k}+\mathfrak{k},N)-1$ of (A',B')
$(A',B') \leftarrow (0,B'^{\mathsf{T}}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}(A'^{\mathsf{T}},\mathsf{ksk}_{s^{\mathsf{T}}\to s})$
$(A',B') \leftarrow \mathsf{MatMultPt}^{\mathsf{CI^3\text{-}GL}}((A',B'),\mathsf{PT}'_X)$ *one level*
$(A',B') \leftarrow (0,B'^{\mathsf{T}}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}(A'^{\mathsf{T}},\mathsf{ksk}_{s^{\mathsf{T}}\to s})$
return (A',B')

Algorithm 6. CI^3-GL slot-to-coefficient transform by matrix products (matrix-product bootstrapping). A line $(0,B'^{\mathsf{T}}) + \mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}(A'^{\mathsf{T}},\mathsf{ksk}_{s^{\mathsf{T}}\to s})$ is a homomorphic transpose: transposing A' and B' transposes the slot matrices but moves the key to s^{T}, and the key switch of A' brings it back to s (B' needs none). The lines with T_{23} and $\mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{aux}}$ do the same on a V-tuple: they swap W and V, and $\mathsf{ksk}^{\mathsf{aux}}$ brings the key from $s(X,V)$ back to $s(X,W)$.

The slots of an element $f \in \tilde{\mathcal{R}}^{\mathsf{CI^3\text{-}GL}}$ with tensor C are $f(\zeta^{5^j},\eta^{\gamma^\kappa}) = (D_n C D_W^{\mathsf{T}})[j,\kappa]$; we call them the slots of an RLWE ciphertext of f.

Theorem 3 (correctness of $\mathsf{CtS}^{\mathsf{mm}}$ and $\mathsf{StC}^{\mathsf{mm}}$)**.** *Let p be prime, let (A,B) decrypt to T at scale Δ (e.g., $T = \mathsf{PT} + E + q_0 E_{\mathsf{ModRaise}}$ after $\mathsf{ModRaise}$), and let the plaintexts be encoded at scales Δ_w equal to the dropped primes. Up to the encoding, key-switching, and rescaling errors:*

1. *after the row stage of $\mathsf{CtS}^{\mathsf{mm}}$, RLWE ciphertext k has the slots $T[:,k,:]D_W^{\mathsf{T}}/\Delta$;*
2. *after $\mathsf{CtS}^{\mathsf{mm}}$, RLWE ciphertext k has the slots $T[:,k,:]/\Delta$, i.e., slot (j,κ) holds the coefficient of $c_j(X)c_k(Y)\lambda_{\kappa+1}(W)$ in T, divided by Δ;*
3. *if RLWE ciphertext k of the input of $\mathsf{StC}^{\mathsf{mm}}$ has the slots $\mathsf{PT}[:,k,:]/\Delta$ for a real tensor PT, then the output has the slot matrices of $\mathsf{PT}\times_1 D_n \times_2 D_n \times_3 D_W/\Delta$.*

Axes 2 and 3 of a V-tuple both use the basis $\{\lambda_u\}$, so σ_{WV} is the transpose $T \mapsto T^{\mathsf{T}_{23}}$, and the tensor algorithms of Section 5.2 implement these maps.

Proposition 11 (cost of matrix-product bootstrapping)**.** *Matrix-product bootstrapping consumes $4 + \ell_{\mathsf{EvalMod}}$ levels, where ℓ_{EvalMod} is the number of levels consumed by $\mathsf{EvalMod}$; $\mathsf{ModRaise}$ consumes none. Each of $\mathsf{CtS}^{\mathsf{mm}}$ and $\mathsf{StC}^{\mathsf{mm}}$ uses 2 levels and, per CI^3-GL ciphertext, 2 $\mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{big}}$ and $2\lceil d/\mathfrak{k}\rceil$ $\mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{aux}}$. It uses the keys $\mathsf{ksk}_{s^{\mathsf{T}}\to s}$ (the first key of $\mathsf{evk}_{\circledast}$) and $\mathsf{ksk}^{\mathsf{aux}}$, the latter of the size of $\mathfrak{k}$ small keys; bootstrapping adds the relinearization key evk of $\mathsf{EvalMod}$. Neither a rotation key nor a conjugation key is needed.*

Corollary 2 (bootstrapping by matrix products)**.** $\mathsf{Boot}^{\mathsf{CI^3\text{-}GL}}$ *consists of* ModRaise, $\mathsf{CtS}^{\mathsf{mm}}$, EvalMod *on the d RLWE ciphertexts, and* $\mathsf{StC}^{\mathsf{mm}}$*. Under the assumptions of Theorem 3, if the entries of* $\mathsf{PT} + E$ *lie in* $(-q_0/2, q_0/2)$*, those of* E_{ModRaise} *are bounded by* B_{ModRaise}*, and* EvalMod *is accurate on the interval set by* B_{ModRaise}*, then* $\mathsf{Boot}^{\mathsf{CI^3\text{-}GL}}$ *returns an encryption of* PT *at level* $L - 4 - \ell_{\mathsf{EvalMod}}$*, where L is the level after* ModRaise*, up to the errors of the transforms, of* EvalMod*, and of key switching.* EvalMod *runs once per RLWE ciphertext, at rank* $N(p-1)/2$*, on real slots.*

These counts do not include the two shortcuts of Crockett et al. [CGK+26]: skipping the first transpose by partitioning the ciphertext along X instead of Y, so that the transform runs along Y, and omitting the last transpose. The sparse-secret encapsulation of Crockett et al. [CGK+26] also applies: one $\mathsf{Switch}^{\mathsf{CI^3\text{-}GL}}_{\mathsf{small}}$ to a sparse key s^{sp} at q_0 before ModRaise, and the switch back merged into the first transpose by using $\mathsf{ksk}_{(s^{\mathsf{sp}})^{\mathsf{T}} \to s}$ in place of $\mathsf{ksk}_{s^{\mathsf{T}} \to s}$. This adds one key switch per RLWE ciphertext. Alternatively, ModRaise runs under the main key s, with no extra key switch.

At the same d and number of matrices ($\mathsf{CI^3}$-GL at batch index $3p$, Section 4.5), the two routes have the same structure: a row stage between two transposes with big keys, and a batch stage on stacked RLWE ciphertexts between two swaps with an auxiliary key. In $\mathsf{CI^3}$-GL, the transposes need no conjugation, every key and RLWE ciphertext has half the rank of its GL counterpart, and EvalMod runs on real slots. Table 3 compares the two step by step.

It sets matrix-product bootstrapping next to GL bootstrapping (Crockett et al. [CGK+26], Algorithm 1 in their Section 5.1) at the same d and $\phi(p)$ matrices per plaintext, with $\mathsf{CI^3}$-GL at batch index $3p$ (Section 4.5), so that $\mathfrak{k} = \phi(p)$. The GL counts follow Algorithm 1 of Crockett et al. [CGK+26], which uses the sparse-secret encapsulation and skips the first conjugate transpose ConjTranspose (called Herm in Crockett et al. [CGK+26]; see Section 2.3); the $\mathsf{CI^3}$-GL counts exclude both the encapsulation and the shortcuts.

item	GL [CGK+26]	CI^3-GL (this work), matrix-product bootstrapping
condition on p	Algorithm 1 assumes $\phi(p) = d$ (general case by padding or grouping); benchmarks: $\phi(p) = 4d$	$3p$ odd squarefree: every odd prime $p \neq 3$
ModRaise	per RLWE ciphertext, to $Q_L P$	per slice: lift of A, B; overflow in $\tilde{\mathcal{R}}^{\mathsf{CI^3\text{-}GL}}$
CtS, X axis	ConjTranspose, ct–pt product with the encoding of $D_X^{\mathsf{GL}^{-1}}$, ConjTranspose (Algorithm 1 skips the first ConjTranspose)	transpose, $\mathsf{MatMultPt}^{\mathsf{CI^3\text{-}GL}}$ with PT_X (slots D_{4d}^{-1}), transpose
CtS, W axis	$\phi(p)$ RLWE ciphertexts stacked along V; ConjTranspose, ct–pt product with $D_W^{\mathsf{GL}^{-1}}$, ConjTranspose	per V-tuple: swap $W \leftrightarrow V$, BatchMult with PT_W (slots D_W^{-1}), swap back
key switches in CtS	Algorithm 1, per GL ciphertext (d RLWE ciphertexts): 1 small (to s^{sp}) + 1 ConjTranspose on $\mathcal{R}^{\mathsf{GL}}$ + 2 ConjTranspose on the V-stacked ring	per ciphertext: 2 big + $2\lceil d/\phi(p) \rceil$ auxiliary; $p = 257$: 2 + 2 at $d \leq 256$, 2 + 4 at $d = 512$
key switches in StC	as CtS; with Algorithm 1 reversed, the final ConjTranspose is omitted	as in CtS
levels per direction	CtS: 1; StC: 2	2
EvalMod	per RLWE ciphertext, rank $2d\phi(p)$; sine approximation of degree 49	per RLWE ciphertext, rank $d\phi(p)$, real slots
conjugations	ConjTranspose in CtS and StC	none
sparse-secret encapsulation	1 extra key switch per RLWE ciphertext	1 extra key switch per RLWE ciphertext (not counted above)
auxiliary key	$\phi(p)$ GL small keys (256 at $p = 257$)	$\phi(p)$ CI^3-GL small keys, each half a GL small key (256 at $p = 257$)

Table 3. GL bootstrapping [CGK+26] vs. CI^3-GL matrix-product bootstrapping, same d and $\phi(p)$ matrices per plaintext

6. Implementation and Security

6.1. Prototype

We implemented the algorithms of Section 3 to Section 5 in Python, with exact integer arithmetic in RNS form and hybrid key switching. This implementation is a prototype, not an optimized library; its goal is to check the correctness of our algorithms on encrypted data.

6.2. Security

All schemes in this paper are RLWE-based. We summarize what is known about their rings; we give no new reduction.

The secret key of CI-CKKS lies in $\mathcal{R}^{\mathsf{CI\text{-}CKKS}}$, of rank N. Kim and Song [KS19] show that $\mathcal{R}^{\mathsf{CI\text{-}CKKS}} = \mathbb{Z}[\xi]$ is the ring of integers of $\mathbb{Q}(\xi)$, $\xi = \zeta + \zeta^{-1}$, so that the reduction of Peikert, Regev, and Stephens-Davidowitz [PRS17] from SIVP to RLWE over the ring of integers of a number field applies. However, its hypotheses (spherical Gaussian errors in the canonical embedding, a uniform secret) are not met by the sparse ternary secrets and coordinate-wise errors used in practice, so it provides no concrete security guarantee. They also observe that an RLWE sample over $\mathcal{R}^{\mathsf{CI\text{-}CKKS}}$ is an N-dimensional LWE sample with a structured matrix, and that no known attack exploits this structure; concrete security is therefore estimated as for LWE of dimension N.

The secret key of CI^3-GL lies in $\tilde{\mathcal{R}}^{\mathsf{CI^3\text{-}GL}} = \mathcal{R}_{\{X\}}^{\mathsf{CI},X} \otimes \mathcal{R}_{\{W\}}^{\mathsf{CI},W}$, of rank $N(p-1)/2 = d\phi(p)/2$ for prime p (Section 4.2). It is the subring of $\tilde{\mathcal{R}}_{n,n,p}$ fixed by $X \mapsto X^{-1}$ and by $W \mapsto W^{-1}$ separately, generated by ξ and $\eta + \eta^{-1}$. For every odd p, both key rings are rings of integers of totally real fields: $\tilde{\mathcal{R}}^{\mathsf{CI^3\text{-}GL}} = \mathcal{R}_{\{X\}}^{\mathsf{CI},X} \otimes \mathcal{R}_{\{W\}}^{\mathsf{CI},W}$ is that of $\mathbb{Q}(\xi, \eta + \eta^{-1})$, because $\mathcal{R}_{\{X\}}^{\mathsf{CI},X}$ and $\mathcal{R}_{\{W\}}^{\mathsf{CI},W}$ are the rings of integers of the linearly disjoint fields $\mathbb{Q}(\xi)$ and $\mathbb{Q}(\eta + \eta^{-1})$,

whose discriminants (a power of 2 and an odd number) are coprime, and $\tilde{\mathcal{R}}^{\mathsf{CI}^1\text{-}\mathsf{GL}}$ is that of $\mathbb{Q}(\zeta\eta + (\zeta\eta)^{-1})$, because $\tilde{\mathcal{R}}_{n,n,p} \cong \mathbb{Z}[\zeta\eta]$ with $\zeta\eta$ a primitive np-th root of unity and $\tilde{\mathcal{R}}^{\mathsf{CI}^1\text{-}\mathsf{GL}}$ is its subring fixed by complex conjugation. The reduction of Peikert, Regev, and Stephens-Davidowitz [PRS17] therefore applies formally to RLWE over both key rings, with the worst-case problem on ideal lattices of these fields. As for CI-CKKS, its hypotheses are not met by the secrets and errors used in practice, so it provides no concrete security guarantee.

Remark 4 (reduction to RLWE over the key rings). As in GL [GL25], a ciphertext is a collection of RLWE ciphertexts over the key ring under one secret: d for CI^3-GL and CI^1-GL, $2d$ for CI^2-GL, with errors and randomness sampled independently per RLWE ciphertext. GL reduces its security in this way to RLWE over $\tilde{\mathcal{R}}^{\mathsf{GL}}$; the same argument reduces our schemes to RLWE over their key rings. Since each key ring is the ring of integers of a totally real field (above; CI^2-GL has the key ring of CI^3-GL), the reduction of [PRS17] applies to RLWE over it formally, with the same caveat as for CI-CKKS; we claim no hardness result beyond this. The big keys $\mathsf{ksk}_{s^\intercal \to s}$ and $\mathsf{ksk}_{ss^\intercal \to s}$ encrypt a linear and a quadratic function of s, with $d \cdot \mathsf{dnum}$ RLWE samples each, the relinearization and rotation keys encrypt s^2 and automorphic images of s, and $\mathsf{ksk}^{\mathsf{aux}}$ encrypts $s(X, V)$. As for GL and CKKS, this requires a circular-security assumption.

References

[ANR74] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[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 Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 387–421. Springer, Cham, August 2024.

[BMTH21] Jean-Philippe Bossuat, Christian Mouchet, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 587–617. Springer, Cham, October 2021.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 34–54. Springer, Cham, May 2019.

[CGK+26] Eric Crockett, Craig Gentry, Hyojun Kim, Yeongmin Lee, and Yongwoo Lee. Efficient bootstrapping in fully homomorphic encryption for matrix arithmetic. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026*, volume 16801 of *Lecture Notes in Computer Science*, pages 492–524. Springer, 2026.

[CHK+18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 360–384. Springer, Cham, April / May 2018.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Cham, December 2017.

[CKL25] Jung Hee Cheon, Minsik Kang, and Junho Lee. Fast batch matrix multiplication in ciphertexts. Cryptology ePrint Archive, Report 2025/1957, 2025.

[CKY18] Jung Hee Cheon, Andrey Kim, and Donggeon Yhee. Multi-dimensional packing for HEAAN for approximate matrix arithmetics. Cryptology ePrint Archive, Report 2018/1245, 2018.

[CS25] Jean-Sébastien Coron and Tim Seuré. PaCo: Bootstrapping for CKKS via partial CoeffToSlot. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part VII*, volume 16251 of *LNCS*, pages 36–67. Springer, Singapore, December 2025.

[GL25] Craig Gentry and Yongwoo Lee. Fully homomorphic encryption for matrix arithmetic. Cryptology ePrint Archive, Report 2025/1935, 2025.

[HS18] Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in HElib. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 93–120. Springer, Cham, August 2018.

[İKOS26] Efe İzbudak, Kubra Kaytanci, Ferruh Ozbudak, and Erkay Savas. Sub-cubic homomorphic matrix multiplication via self-dual normal bases. Cryptology ePrint Archive, Paper 2026/1592, 2026.

[Ins24] Lattigo v6. Online: `https://github.com/tuneinsight/lattigo`, August 2024. EPFL-LDS, Tune Insight SA.

[JLK+22] Jaehee Jang, Younho Lee, Andrey Kim, Byunggook Na, Donggeon Yhee, Byounghan Lee, Jung Hee Cheon, and Sungroh Yoon. Privacy-preserving deep sequential model with matrix homomorphic encryption. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIACCS 22*, pages 377–391. ACM Press, May / June 2022.

[KS19] Duhyeong Kim and Yongsoo Song. Approximate homomorphic encryption over the conjugate-invariant ring. In Kwangsu Lee, editor, *ICISC 18*, volume 11396 of *LNCS*, pages 85–102. Springer, Cham, November 2019.

[LMWW26] Jiaming Liu, Shihe Ma, Anyu Wang, and Xiaoyun Wang. Efficient (functional) bootstrapping for CKKS over the conjugate-invariant ring. Cryptology ePrint Archive, Paper 2026/2618, 2026.

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 461–473. ACM Press, June 2017.

[Seu25] Tim Seuré. SCORE: A SlotToCoeff optimization for real-vector encryption in CKKS. Cryptology ePrint Archive, Report 2025/1983, 2025.

[ST26] Rostin Shokri and Nektarios Georgios Tsoutsos. Low-depth bootstrapping for matrix-native FHE. Cryptology ePrint Archive, Paper 2026/811, 2026.

[ZLW25] Xiaopeng Zheng, Hongbo Li, and Dingkang Wang. A new framework for fast homomorphic matrix multiplication. *DCC*, 93(7):2671–2693, 2025.

Appendix A. Construction of realfft3

The construction is the real analog of the CKKS FFT: up to a row permutation and a factor of 2, D_n is a DCT-III (discrete cosine transform [ANR74]), whose even/odd recursion turns the odd half into a complex FFT of half the length with real and imaginary parts $N/2$ slots apart. The product identity is checked in floating point: entrywise for $N \le 2^9$, and on 16 columns of the decoding matrix for $N = 2^{12}, 2^{14}, 2^{15}, 2^{16}$ (largest error $1.4 \cdot 10^{-10}$), and the condition numbers are computed for $N \le 2^9$; we do not give a proof for every N, so the results below that use the factorization assume the factorization realfft3 (Section 3.3). For comparison, a complex Cooley–Tukey stage of CKKS is a scaled unitary map, with condition number 1, so a CI stage can amplify a relative error by a factor of up to 2, compared with 1 for a complex stage, while D_n itself has condition number $\sqrt{2}$ (Lemma 3).

The stages come from the classical DCT-III recursion, rewritten so that every butterfly is a slot rotation. With $D_n[j,u] = c_u(\zeta^{5^j}) = 2\cos(2\pi 5^j u/n)$ for $u \ge 1$ and $D_n[j,0] = 1$, folding the exponent $r = 5^j \bmod n$ to $\min(r, n-r) = 2k+1$ gives $D_n[j,u] = 2\cos(\pi u(2k+1)/(2N))$, so the rows of D_n are those of $2\,C_N^{\mathsf{III}}$ (DCT-III with weight $1/2$ on column 0, absorbed by Λ_n) in a permuted order. For example, at $n = 16$, the exponents $1, 5, 9, 13$ fold to $1, 5, 7, 3$, i.e., $k = 0, 2, 3, 1$.

The recursion has three ingredients. Write $M := N/2$.

1. Even/odd split: $y_j = y_j^{\mathsf{e}} + y_j^{\mathsf{o}}$ and $y_{j+M} = y_j^{\mathsf{e}} - y_j^{\mathsf{o}}$, where $\vec{y}^{\mathsf{e}}$ is the CI decoding of size M of the even coordinates and $\vec{y}^{\mathsf{o}}$ a DCT-IV-type kernel on the odd ones. On slots this is the CI butterfly $(x_j, x_{K+j}, x_{j+M}, x_{K+j+M}) \leftarrow (x_j + x_{K+j},\ x_{j+M} + x_{K+j+M},\ x_j - x_{K+j},\ x_{j+M} - x_{K+j+M})$ on the block of width K being processed.
2. Complex FFT of half length: the odd coordinates are packed as $z_k = v_{4k+1} - \mathsf{i}\, v_{N-1-4k}$ and transformed by a CKKS-type FFT in slot order 5^j, with real parts in slots $[K, 2K)$ and imaginary parts in slots $[K+M, 2K+M)$; the complex butterfly $(z_E, z_O) \leftarrow (z_E + \varpi z_O,\ z_E - \varpi z_O)$, $|\varpi| = 1$, acts on $z_r = x_r + \mathsf{i}\, x_{r+M}$.
3. Twist: a real 2×2 post-twiddle $(x_r, x_{r+M}) \leftarrow (2\mathsf{Re}\,\varpi\, x_r - 2\mathsf{Im}\,\varpi\, x_{r+M},\ c_\psi(\mathsf{Im}\,\varpi\, x_r + \mathsf{Re}\,\varpi\, x_{r+M}))$, with ϖ a root of unity and c_ψ a real constant of the level, which recovers the real odd part.

The base stage is $(x_0, x_M) \leftarrow (x_0 + \sqrt{2}\, x_M,\ x_0 - \sqrt{2}\, x_M)$, with Ω placing v_0 at slot 0 and $v_{N/2}$ at slot M (indeed $c_{N/2}(\zeta^{5^j}) = 2\cos(\pi 5^j/4) = \pm\sqrt{2}$). To save depth, the twist is fused into the last FFT stage of its block, and the CI butterfly runs in the same stage as the FFT stage of the same half-width; this gives exactly $\log_2 N$ stages. Real and imaginary parts sit M slots apart, so a complex butterfly at distance w needs the offsets $\{0, \pm w, M, M \pm w\}$, i.e., the sets O_i of the factorization realfft3 (Section 3.3). The layout Ω collects the bit reversals of all recursion levels. For example, at $n = 32$ ($N = 8$),

$$\Omega = ((0,+), (2,+), (1,+), (5,+), (4,+), (6,-), (7,-), (3,-))$$

as (index, sign) per slot, with offset sets $\{0,4\}$, $\{0, \pm 1, 4, 4 \pm 1\}$, $\{0, \pm 2, 4\}$ (mod 8): $1 + 3 + 2 = 6$ rotations and $J_{\mathsf{rot}} = \{4, 2, \pm 1\}$.

Appendix B. Constants of Cl^3-GL

The products and automorphisms of Section 4.4 become fixed integer tensors, obtained from the reductions $\Phi_n(X) = 0$ and $\Phi_p(W) = 0$ on the invariant bases. Recall that $N =$

$\phi(n)/2 = n/4$ and $X^{2N} = -1$. For an integer t, let $|t|_n := \min(t \bmod n,\ n - (t \bmod n))$ and likewise $|t|_p$.

1. The reduction along X is the coefficient vector $\vec{\omega}_X(t) \in \mathbb{Z}^N$ of $X^t + X^{-t}$ in $\bar{c}_n^{\mathsf{CI}}$. With $t^* := |t|_n$, its entries $\omega_X(t)_u$ are $2\delta_{u,0}$ if $t^* = 0$; δ_{u,t^*} if $0 < t^* < N$; 0 if $t^* = N$; $-\delta_{u,2N-t^*}$ if $N < t^* < 2N$; $-2\delta_{u,0}$ if $t^* = 2N$.
2. The reduction along W is the coefficient vector $\vec{\omega}_W(t) \in \mathbb{Z}^{(p-1)/2}$ of λ_t in $\{\lambda_u\}$: $\omega_W(t)_u = \delta_{u,|t|_p}$ if $|t|_p \geq 1$, and $\vec{\omega}_W(t) = -2\,\vec{1}$ if $p \mid t$ (from $\lambda_0 = 2$).
3. The structure tensor $\mathrm{Y}^X \in \mathbb{Z}^{N\times N\times N}$ holds the coefficients of $c_a c_{a'}$: $\mathrm{Y}^X[0, a', l] = \delta_{l,a'}$, $\mathrm{Y}^X[a, 0, l] = \delta_{l,a}$, and $\mathrm{Y}^X[a, a', :] = \vec{\omega}_X(a + a') + \vec{\omega}_X(a - a')$ for $a, a' \geq 1$. It is also used for axis 2. The structure tensor Y^W holds the coefficients of $\lambda_u \lambda_{u'}$: $\mathrm{Y}^W[u, u', :] = \vec{\omega}_W(u + u') + \vec{\omega}_W(u - u')$.
4. The ring product (Section 4.4) uses Y^X on axes 1 and 2 and Y^W on axis 3.
5. $\Psi_\nu \in \mathbb{Z}^{N\times N}$ is the matrix of $X \mapsto X^{5^\nu}$ on $\mathcal{R}^{\mathsf{CI},X}_{\{X\}}$: column 0 is the first unit vector and column $a \geq 1$ is $\vec{\omega}_X(5^\nu a)$; it is a signed permutation. $\Psi^W_\nu \in \mathbb{Z}^{(p-1)/2\times(p-1)/2}$ is the matrix of $W \mapsto W^{\gamma^\nu}$ on $\mathcal{R}^{\mathsf{CI},W}_{\{W\}}$: column u is $\vec{\omega}_W(\gamma^\nu u)$; it is a permutation, and $\Psi^W_{(p-1)/2}$ is the identity.

Slots. Since $m \in \mathcal{R}^{\mathsf{CI}^3\text{-}\mathsf{GL}}$ is a polynomial in $X + X^{-1}$, $Y + Y^{-1}$ and $W + W^{-1}$, slot (j, k, κ) depends only on the values $\xi_j = \zeta^{5^j} + \zeta^{-5^j}$, ξ_k (Section 2.4) and $\eta^{\gamma^\kappa} + \eta^{-\gamma^\kappa}$ of these three sums. Since $\gamma^{(p-1)/2} \equiv -1 \pmod p$, the points η^{γ^κ}, $0 \leq \kappa < (p-1)/2$, take one point from each pair $\{\eta^x, \eta^{-x}\}$.

Composite Squarefree p. Comparisons with GL use the batch index $3p$ (Section 4.5), for which the prime-case formulas change. The axis-3 size becomes $\phi(p)/2$, and λ_u runs over representatives of $\mathbb{Z}_p^*/\{\pm 1\}$. In items 2 and 5 and in the slots, $(p-1)/2$ becomes $\phi(p)/2$: $\Psi^W_{\phi(p)/2} = \mathsf{Id}$, and the points η^{γ^κ}, $0 \leq \kappa < \phi(p)/2$, take one point from each pair $\{\eta^x, \eta^{-x}\}$ because γ generates $\mathbb{Z}_p^*/\{\pm 1\}$ (the identity $\gamma^{\phi(p)/2} \equiv -1$, i.e. $\gamma^{(p-1)/2} \equiv -1$ for prime p, is prime-only: at $p = 771$, $\gamma = 5$ and $5^{256} \equiv 1$). The reduction along W becomes $\omega_W(t)_u = (-1)^{\omega(g)}\big([u \equiv t \bmod p/g] + [u \equiv -t \bmod p/g]\big)$ with $g = \gcd(t, p)$ and $\omega(g)$ the number of prime factors of g; for $g = 1$ this is $\delta_{u,|t|_p}$, and for prime p and $g = p$ it is -2 on every entry, as in item 2. Let $\mathsf{möb}$ be the Möbius function and $\mathsf{Ram}_p(t) := \sum_{u\in\mathbb{Z}_p^*} \eta^{ut}$ the Ramanujan sum. Then $\sum_{u\in\mathbb{Z}_p^*} W^u = \mathsf{möb}(p)$, so $1 = \mathsf{möb}(p)\sum_u \lambda_u$. The trace Gram matrix of $\{\lambda_u\}$ has entries $\mathsf{Ram}_p(u + v) + \mathsf{Ram}_p(u - v)$; the form $p\,\mathsf{Id} - 2\vec{1}\vec{1}^{\mathsf{T}}$ of Lemma 9 holds for p prime only. The inverse is $D_W^{-1}[u, \kappa] = \chi_c(u\gamma^\kappa)$, with $\chi_c(x) := p^{-1}\sum_S (-1)^{|P\setminus S|}\, 2\cos(2\pi x e_S/p)$, the sum over subsets S of the set P of prime factors of p, and e_S the CRT idempotent ($e_S \equiv 1$ modulo the primes in S, 0 modulo the others); for p prime, $\chi_c(x) = (2\cos(2\pi x/p) - 2)/p$. The condition number stays $\sqrt{p}$ for $p \geq 5$. The batch rotation requires $\mathbb{Z}_p^*/\{\pm 1\}$ to be cyclic, with γ a generator (for prime p, the primitive root γ of Section 2.3 is one): $\gamma = 3$ at $p = 17, 257$ and $\gamma = 5$ at $p = 771$. This fails at $p = 1285$, where the batch axis is a torus $\mathbb{Z}/2 \times \mathbb{Z}/256$.