

Precision Allocation for CKKS

Zhaoyang Liang¹ and Dan Ding^{1,2,3(✉)}

¹ College of Cryptology and Cyber Sciences, Nankai University, Tianjin 300350, China P. R.

liangzhaoyang@mail.nankai.edu.cn, dingdan@nankai.edu.cn

² College of Computer Sciences, Nankai University, Tianjin 300350, China P. R.
dingdan@nankai.edu.cn

³ Serica Laboratory, Beijing 100084, China P. R.
dingdan@serica-tech.com.cn

Abstract. xxx

Keywords: A · B

1 Introduction

1.1 Our Contributions

1.2 Technical Overview

Let $R = \mathbb{Z}[X]/(X^N + 1)$ for a power-of-two N , and let σ_j denote its $r = N/2$ CKKS slot embeddings. Our technique uses a public ring unit $u \in R^\times$, with $u^{-1} \in R$ and $\prod_{j=0}^{r-1} |\sigma_j(u)| = 1$. Thus, smaller error weights in some slots are balanced by larger ones elsewhere. We represent a message x_j in slot j by $z_j = x_j/\sigma_j(u)$; if it acquires an error η_j , restoration gives

$$\sigma_j(u)(z_j + \eta_j) = x_j + \sigma_j(u)\eta_j.$$

The messages remain independent, while $|\sigma_j(u)|$ determines their restored error weights.

Figure 1(a) shows requirements $\mathbf{b} = (160, 32, 32, 32)$ with average $\bar{b} = 64$ bits. For an idealized common error level of 2^{-64} before restoration, the target weights and corresponding error bounds are

$$\begin{aligned}\mathbf{w} &= (2^{-96}, 2^{32}, 2^{32}, 2^{32}), \\ 2^{-64}\mathbf{w} &= (2^{-160}, 2^{-32}, 2^{-32}, 2^{-32}).\end{aligned}$$

These weights satisfy the unit constraint. Figures 1(b) and (c) illustrate the flexibility of this allocation.

For general requirements b_j , set $\bar{b} = r^{-1} \sum_j b_j$ and $t_j = \bar{b} - b_j$. To construct a unit approximating these targets, let ζ be a primitive $4r$ -th root of unity, index

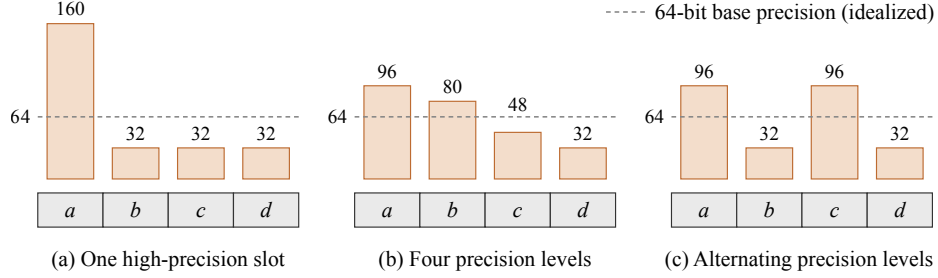


Fig. 1. Examples of precision requirements for four independent CKKS slots. The dashed line indicates the idealized 64-bit average requirement in each case.

the embeddings by $\sigma_j(\zeta) = \zeta^{5^j}$, and set $\tau(\zeta) = \zeta^5$. For integer exponents n_k , choose a real cyclotomic unit and form products of its conjugates:

$$v = \zeta^{-2} + \zeta^{-1} + 1 + \zeta + \zeta^2, \quad u(\mathbf{n}) = \prod_{k=0}^{r-1} \tau^{-k}(v)^{n_k},$$

$$\ell_j = \log_2 |\sigma_j(v)|, \quad \log_2 |\sigma_j(u(\mathbf{n}))| = \sum_{k=0}^{r-1} \ell_{j-k} n_k.$$

This reduces unit construction to a circular convolution solved using the discrete Fourier transform (DFT):

$$\hat{a}_0 = 0, \quad \hat{a}_k = \frac{\hat{t}_k}{\ell_k} \quad (1 \leq k < r), \quad n_j = \lfloor a_j \rfloor,$$

where $\mathbf{a}$ is obtained by the inverse DFT. Rounding yields a unit approximating the target logarithmic magnitudes with an explicit error bound.

We analyze how the supported CKKS operations preserve this representation. In particular, ciphertext multiplication takes the form

$$\mathbf{d}_{\text{out}} = \text{RS}_D(\text{Relin}_{\text{evk}}(u(\mathbf{d}_a \otimes \mathbf{d}_b))).$$

The public correction by u precedes relinearization and requires no additional plaintext scale.

The allocation can permit smaller CKKS scales. Under an idealized common error bound E/Δ before restoration, the target weights give

$$\Delta_{\text{unit}} \gtrsim E2^{\bar{b}}, \quad \Delta_{\text{std}} \gtrsim E2^{b_{\max}}, \quad b_{\max} = \max_j b_j.$$

The encoding also satisfies, for any $\lambda > 0$,

$$\frac{\Delta x_j}{\sigma_j(u)} = \frac{(\Delta/\lambda)x_j}{\sigma_j(u)/\lambda}.$$

This equivalent rewriting alone does not reduce modulus consumption. Smaller feasible scales can reduce rescaling divisors and the number of RNS primes, subject to evaluation errors and message recovery constraints.

1.3 Related Work

Our work is related to several lines of research, but differs in its underlying techniques and objectives, as discussed below.

Algebraic Representations and Scaling. CHIMERA [4] considers a ring element $P \in R$ that is invertible in $R_{\mathbb{R}}$ and represents the plaintext quotient R/PR as $P^{-1}R/R$, enabling switching between different homomorphic encryption schemes. GBFV [13] generalizes BFV by replacing its scalar plaintext modulus with a polynomial $t(X)$, changing the plaintext ring from R/pR to R/tR and introducing an algebraic scaling factor $q/t(X)$ to reduce noise growth.

While our approach differs in three respects: (i) we preserve the original CKKS plaintext space rather than defining a new plaintext quotient; (ii) we retain the standard scalar scale instead of introducing an algebraic scaling factor; and (iii) we apply an invertible coordinate transformation directly to the messages using a public ring unit. Our goal is to allocate errors according to different precision requirements across slots while preserving independent messages and their homomorphic evaluation.

High-Precision Homomorphic Computation. Tuple-CKKS [7] decomposes ciphertexts into multiple components and adapts homomorphic multiplication to reduce modulus consumption for high-precision arithmetic. Bian et al. [2] employ Fourier extensions to improve function approximation accuracy in CKKS functional bootstrapping. Unlike their use of Fourier series for function approximation, we employ the discrete Fourier transform to construct cyclotomic units with prescribed logarithmic embedding magnitudes. Our approach targets heterogeneous precision requirements across independent slots, without decomposing ciphertexts or relying on bootstrapping for precision allocation.

Modulus Reduction.

Precision and Scale Management. Existing works [16,15,8] improve the efficiency of RNS-CKKS through compiler-level optimization or RNS arithmetic and modulus management. In contrast, our work operates at the message-representation level, introducing an additional optimization dimension that is complementary to existing scale and modulus management techniques.

Other works [6,3,5,14,12,1,17] improve homomorphic computation through alternative plaintext representations, error reduction, or high-precision bootstrapping, whereas our work focuses on allocating heterogeneous precision requirements across independent CKKS slots through a public unit.

1.4 Paper Organization.

Section 2 introduces the notation and preliminaries on CKKS, RNS representations, and algebraic units. Section 3 presents our precision allocation framework, analyzes its homomorphic operations, correctness, and security, and provides a preliminary analysis of the constraints on precision allocation. Section 4 develops an explicit Fourier-based construction of ring units for prescribed precision

requirements. Section 5 investigates recovery constraints and the limits of precision allocation. Section 6 examines scale and modulus reduction and analyzes the associated computation and storage costs. [To Be Done] Implementation

2 Preliminaries

2.1 Notations

Let ζ be a primitive m th root of unity, where $m > 2$, and let $N := \varphi(m)$, $K := \mathbb{Q}(\zeta)$, and $R := \mathbb{Z}[\zeta] \simeq \mathbb{Z}[X]/(\Phi_m(X))$, where Φ_m is the m th cyclotomic polynomial and X is identified with ζ . The embeddings $\sigma : K \hookrightarrow \mathbb{C}$ are determined by $\sigma(\zeta) = \zeta^a$ for $1 \leq a < m$ with $\gcd(a, m) = 1$. They form conjugate pairs $\sigma, \bar{\sigma}$, where $\bar{\sigma}(z) = \overline{\sigma(z)}$. Since $|\bar{\sigma}(z)| = |\sigma(z)|$, choose one representative σ_j from each pair and set $r := N/2$. Extending these maps to polynomials with real coefficients, write $\sigma : \mathbb{R}[X]/(\Phi_m(X)) \rightarrow \mathbb{C}^r$, $f \mapsto (\sigma_j(f))_{j=0}^{r-1}$. The inverse $\sigma^{-1}(\mathbf{z})$ denotes the unique polynomial of degree less than N with real coefficients whose selected embeddings equal $\mathbf{z}$.

For a nonnegative integer n , write $[n] := \{0, \dots, n\}$. For an odd integer $Q > 1$, put $R_Q := R/QR$. The notation $[f]_Q$ means reduction of each coefficient to $(-Q/2, Q/2)$; we also use this representative for an element of R_Q . The operation $\lfloor f \rfloor$ rounds each real coefficient to the nearest integer, with a fixed rule for ties. Vector products and quotients are taken coordinate by coordinate.

For a vector $\mathbf{z}$, write $\|\mathbf{z}\|_\infty := \max_j |z_j|$. For $f = \sum_{k=0}^{N-1} f_k X^k$ with real coefficients, define $\|f\|_{\text{coef}, \infty} := \max_k |f_k|$ and $\|f\|_{\sigma, \infty} := \max_j |\sigma_j(f)|$. All logarithms are to base two unless stated otherwise.

2.2 CKKS Ciphertexts and Precision

CKKS [10] uses $m = 2N$ with N a power of two, so $R = \mathbb{Z}[X]/(X^N + 1)$. The $r = N/2$ coordinates of $\sigma(f)$ are called slots. CKKS encodes a vector $\mathbf{x} \in \mathbb{C}^r$ at a real scale $\Delta > 0$ as

$$\mu = \lfloor \Delta \sigma^{-1}(\mathbf{x}) \rfloor \in R. \quad (1)$$

The encoding error $e_{\text{enc}} := \mu - \Delta \sigma^{-1}(\mathbf{x})$ satisfies $\|e_{\text{enc}}\|_{\text{coef}, \infty} \leq 1/2$. Let $s \in R$ be the secret key and pk its public key. Encrypting $[\mu]_Q$ gives a ciphertext $\text{ct} = (c_0, c_1) \in R_Q^2$. Let $e_{\text{dec}} \in R$ denote the decryption error arising from encryption. The ciphertext phase $c_0 + c_1 s$ satisfies $c_0 + c_1 s \equiv \mu + e_{\text{dec}} \pmod{Q}$.

Decryption computes $\text{Dec}_s(\text{ct}) := [c_0 + c_1 s]_Q$. If $\|\mu + e_{\text{dec}}\|_{\text{coef}, \infty} < Q/2$, decryption returns $\mu + e_{\text{dec}}$, so $e_{\text{dec}} = \text{Dec}_s(\text{ct}) - \mu$. Decoding then gives $\hat{x}_j := \sigma_j(\text{Dec}_s(\text{ct}))/\Delta = x_j + \sigma_j(e_{\text{enc}} + e_{\text{dec}})/\Delta$. A target of b_j bits of absolute precision requires $|\hat{x}_j - x_j| \leq 2^{-b_j}$.

For $f, g \in \mathbb{R}[X]/(X^N + 1)$, the coefficient and embedding norms satisfy

$$\|f\|_{\sigma, \infty} \leq N \|f\|_{\text{coef}, \infty}, \quad \|f\|_{\text{coef}, \infty} \leq \frac{1}{r} \sum_{j=0}^{r-1} |\sigma_j(f)| \leq \|f\|_{\sigma, \infty}. \quad (2)$$

Appendix G proves (2). Multiplication also satisfies $\|fg\|_{\sigma, \infty} \leq \|f\|_{\sigma, \infty} \|g\|_{\sigma, \infty}$.

2.3 RNS Representation and Parameters

A residue number system (RNS) stores a polynomial separately modulo each prime factor of its modulus [9]. Write $Q = \prod_{i=0}^{L_Q-1} q_i$ for the ciphertext modulus and $P = \prod_{i=0}^{L_P-1} p_i$ for the auxiliary modulus used in key switching; L_Q and L_P are their respective prime counts. The primes are pairwise distinct, fit in one machine word, and satisfy $q_i \equiv p_j \equiv 1 \pmod{2N}$ to support the number theoretic transform (NTT).

Each of the N coefficients of a polynomial modulo Q is stored modulo L_Q primes, giving NL_Q residue coefficients. In NTT form, one polynomial multiplication uses NL_Q pointwise modular products. We count one NTT or inverse NTT of length N modulo one prime as one transform. An RNS conversion from modulus A to B computes $[a]_B$ for each coefficient, where a is its chosen integer representative modulo A . Modular products, transforms, and RNS conversions are counted separately; the first count excludes work inside the other two.

Key switching $\text{KS}_{s' \rightarrow s}$ uses a public evaluation key evk to convert a ciphertext under s' into one under s . We use an RNS decomposition with blocks of $L_P \geq 1$ primes of Q , except possibly the last block, and no further binary decomposition [18]. There are $\beta := \lceil L_Q/L_P \rceil$ blocks. Each block uses an evaluation key pair $(b_k, a_k) \in R_{QP}^2$, whose two polynomials contribute to the two output components. The key therefore contains 2β polynomials. Key switching computes modulo QP and returns a ciphertext modulo Q at the same scale. Appendix F gives the details and error bounds.

2.4 Units and Logarithmic Embeddings

Let $R^\times = \{u \in R : \exists v \in R, uv = 1\}$ denote the group of units. Dirichlet's unit theorem states that a number field with r_1 real embeddings and r_2 pairs of nonreal complex embeddings has $r_1 + r_2 - 1$ fundamental units. Here $r_1 = 0$ and $r_2 = r$, giving $r_1 + r_2 - 1 = r - 1$. Thus every unit $u \in R^\times$ can be written as $u = \pm \zeta^k \prod_{i=1}^{r-1} u_i^{n_i}$, where $k, n_i \in \mathbb{Z}$ and $u_1, \dots, u_{r-1}$ are fixed multiplicatively independent units. Since $|\sigma_j(\pm \zeta^k)| = 1$, we have $|\sigma_j(u)| = \prod_{i=1}^{r-1} |\sigma_j(u_i)|^{n_i}$.

For nonzero $z \in K$, define

$$\text{Log}(z) = (\log_2 |\sigma_j(z)|)_{j=0}^{r-1}, \quad H = \left\{ \mathbf{t} \in \mathbb{R}^r : \sum_{j=0}^{r-1} t_j = 0 \right\}.$$

For a unit u , the norm identity $\prod_j |\sigma_j(u)|^2 = \text{Norm}_{K/\mathbb{Q}}(u) = 1$ places $\text{Log}(u)$ in H . Products become sums, and roots of unity have logarithm zero. Under this embedding, the unit group maps to a lattice of rank $r - 1$ in H : $\Lambda = \text{Log}(R^\times) = \left\{ \sum_{i=1}^{r-1} n_i \text{Log}(u_i) : n_i \in \mathbb{Z} \right\}$. The vectors $\text{Log}(u_i)$ form a basis of H over $\mathbb{R}$, and their integer combinations are exactly the logarithms of units. For targets $w_j > 0$ with $\prod_j w_j = 1$, we seek a lattice point close to $(\log_2 w_j)_j \in H$ to obtain a unit u with $|\sigma_j(u)| \approx w_j$.

The cyclotomic unit group is $C_m = R^\times \cap \langle -1, \zeta, 1 - \zeta^a : 1 \leq a < m \rangle$, where the generated group is taken in $K^\times$. For integers a coprime to m , put

$c_a = (1 - \zeta^a)/(1 - \zeta)$. Since $c_{-a} = -\zeta^{-a}c_a$, we have $\text{Log}(c_{-a}) = \text{Log}(c_a)$. To avoid repetition, use $G = (\mathbb{Z}/m\mathbb{Z})^\times / \{\pm 1\}$, which groups the indices a and $-a$ modulo m corresponding to conjugate embeddings. We identify G with a set of integer representatives containing 1. We use the following two classical results.

Lemma 1 ([19, Lemma 8.1]). *If m is a prime power, C_m is generated by $\pm\zeta$ and c_a for $a \in G \setminus \{1\}$.*

Theorem 1 ([11, Theorem 2.8]). *If $m > 2$ is a prime power, then $[\Lambda : \text{Log}(C_m)] = h^+(m)$, where $h^+(m)$ is the class number of $\mathbb{Q}(\zeta + \zeta^{-1})$.*

For CKKS, $m = 2N = 4r$ is a power of two. The preceding results then give an explicit basis.

Corollary 1 (An explicit basis). *If $m = 4r$ with $r \geq 2$ a power of two, then the vectors $\text{Log}(c_{5^h})$, $1 \leq h < r$, form a basis of H over $\mathbb{R}$.*

Proof. Since 5 has order r modulo $4r$ and $5^h \equiv 1 \pmod{4}$ for $0 \leq h < r$, the elements $5^0, 5^1, \dots, 5^{r-1}$ represent G . Lemma 1 shows that the stated $r-1$ vectors generate $\text{Log}(C_m)$ over $\mathbb{Z}$, as roots of unity have logarithm zero. Theorem 1 implies $[\Lambda : \text{Log}(C_m)] = h^+(m) < \infty$. Thus every $\mathbf{v} \in \Lambda$ satisfies $n\mathbf{v} \in \text{Log}(C_m)$ for some integer $n > 0$. Writing $\mathbf{v} = \frac{1}{n}(n\mathbf{v})$ gives $\text{span}_{\mathbb{R}} \Lambda \subseteq \text{span}_{\mathbb{R}} \text{Log}(C_m)$. The reverse inclusion follows from $\text{Log}(C_m) \subseteq \Lambda$. Their common real span is H , so the $r-1$ vectors form a basis, since $\dim H = r-1$. $\square$

3 Precision Allocation via Ring Multipliers

3.1 Encoding and Message Recovery

We use a public ring multiplier, namely a nonzero element p of the plaintext ring R , to allocate errors across slots. All slots remain available for independent messages, with none reserved for auxiliary data.

For a message x_j in slot j , we encode $x_j/\sigma_j(p)$ rather than x_j . If this value acquires an additive error η_j , restoration gives

$$\sigma_j(p) \left(\frac{x_j}{\sigma_j(p)} + \eta_j \right) = x_j + \sigma_j(p)\eta_j.$$

Let $B > 0$ bound $|\eta_j|$ in every slot before restoration, including accumulated encoding, decryption, and homomorphic evaluation errors. Restoration reduces this bound when $|\sigma_j(p)| < 1$ and increases it when $|\sigma_j(p)| > 1$, thus allocating the precision. For target errors $\boldsymbol{\varepsilon} = (\varepsilon_0, \dots, \varepsilon_{r-1})$, where $\varepsilon_j = 2^{-b_j}$ specifies b_j bits of absolute precision, we therefore seek p satisfying $|\sigma_j(p)|B \leq \varepsilon_j$ for every j .

Specifically, to encrypt a message vector $\mathbf{x} = (x_0, \dots, x_{r-1}) \in \mathbb{C}^r$ at modulus Q and scale $\Delta > 0$, we use the standard CKKS encoder to compute

$$\mu = \left\lfloor \Delta \sigma^{-1} \left(\frac{x_0}{\sigma_0(p)}, \dots, \frac{x_{r-1}}{\sigma_{r-1}(p)} \right) \right\rfloor \in R.$$

The positive real scale Δ is shared by all slots, but the absolute value of the scaling factor in slot j is $\Delta/|\sigma_j(p)|$.

1. *Encryption.* Compute $\text{ct} = (c_0, c_1) = \text{Enc}_{pk}([\mu]_Q)$ using standard CKKS encryption.
2. *Message recovery.* First compute the ciphertext $\text{pct} = ([pc_0]_Q, [pc_1]_Q)$. Then $\text{Dec}_s(\text{pct}) = [p(c_0 + c_1s)]_Q$ and $\hat{x}_j = \sigma_j(\text{Dec}_s(\text{pct}))/\Delta$ for each slot j .

Multiplication by p uses its integer coefficients directly and requires no additional plaintext scale.

To establish correctness, let e_{enc} be μ minus its unrounded value. Standard CKKS encryption satisfies $c_0 + c_1s \equiv \mu + e_{\text{dec}} \pmod{Q}$, where $e_{\text{dec}} \in R$ is the decryption error. For this encoding, $\eta_j = \sigma_j(e_{\text{enc}} + e_{\text{dec}})/\Delta$. These definitions give

$$p(\mu + e_{\text{dec}}) = \Delta \sigma^{-1}(\mathbf{x}) + p(e_{\text{enc}} + e_{\text{dec}}). \quad (3)$$

Multiplication by p cancels the division by its embeddings in the message term, but also multiplies the errors by p .

Since each embedding sums the N coefficients with unit-modulus weights and rounding changes each coefficient by at most $1/2$, we obtain $\|\mu\|_{\text{coef},\infty} \geq \Delta|x_j|/(N|\sigma_j(p)|) - 1/2$ for each slot j . Even if some coefficients of μ lie outside $(-Q/2, Q/2)$, Lemma 2 gives a sufficient condition for correct message recovery after reduction modulo Q . Its proof is given in Appendix A.

Lemma 2 (Correctness of modular restoration). *If $\Delta\|\sigma^{-1}(\mathbf{x})\|_{\text{coef},\infty} + \|p(e_{\text{enc}} + e_{\text{dec}})\|_{\text{coef},\infty} < Q/2$, then $\text{Dec}_s(\text{pct}) = p(\mu + e_{\text{dec}})$ and $\hat{x}_j = x_j + \sigma_j(p)\sigma_j(e_{\text{enc}} + e_{\text{dec}})/\Delta$ for each slot j .*

The following proposition gives finer control over p by bounding its embeddings. The proof is given in Appendix B.

Proposition 1 (Multiplier bounds for target precisions). *If $|\sigma_j(e_{\text{enc}} + e_{\text{dec}})|/\Delta \leq B_j$ and $|\sigma_j(p)| \leq \varepsilon_j/B_j$ for every slot j , with possibly different bounds $B_j > 0$, then $\|p(e_{\text{enc}} + e_{\text{dec}})\|_{\text{coef},\infty} \leq (\Delta/r) \sum_{j=0}^{r-1} \varepsilon_j$. If also*

$$\Delta\left(\|\sigma^{-1}(\mathbf{x})\|_{\text{coef},\infty} + \frac{1}{r} \sum_{j=0}^{r-1} \varepsilon_j\right) < \frac{Q}{2}, \quad (4)$$

then $|\hat{x}_j - x_j| \leq \varepsilon_j$ for every slot j .

Together, Lemma 2 and Proposition 1 give sufficient conditions for recovering the messages at the target precisions. The range condition applies to the polynomial $p(\mu + e_{\text{dec}})$ after restoration. The polynomial $\mu + e_{\text{dec}}$ before restoration need not be a centered representative modulo Q .

In addition, for any $\lambda > 0$, the same encoding can be written using scale Δ/λ and values $x_j/(\sigma_j(p)/\lambda)$, since $(\Delta/\lambda)x_j/(\sigma_j(p)/\lambda) = \Delta x_j/\sigma_j(p)$. With the same rounding, both forms give the same polynomial μ . For fixed p and an unchanged evaluation circuit, this rewriting alone does not reduce computation or storage costs: corresponding executions use the same modulus chain and integer operations. Proposition 5 in Appendix C gives the formal statement and proof.

3.2 Homomorphic Evaluation

We show how to maintain this representation during homomorphic evaluation in this subsection. For a ciphertext $\text{ct} = (c_0, c_1)$ at modulus Q and scale Δ , the multiplier remains fixed throughout evaluation, we use $M \in R$ to denote the polynomial representing the message after restoration. It satisfies

$$p(c_0 + c_1 s) \equiv M \pmod{Q}, \quad \sigma(M) = \Delta(\mathbf{x} + \mathbf{e}), \quad (5)$$

where $\mathbf{x}$ is the intended value and $\mathbf{e}$ is its error. For a fresh ciphertext, $M = p(\mu + e_{\text{dec}})$, where μ is obtained by applying (1) to $(x_j/\sigma_j(p))_{j=0}^{r-1}$.

Evaluation uses standard CKKS ciphertexts, only with the message in slot j represented as $x_j/\sigma_j(p)$. Therefore, relinearization $\text{Relin}_{\text{evk}}$ and key switching $\text{KS}_{s' \rightarrow s}$ require no modification. We next consider the other homomorphic operations. We write M_+ and $M_\times$ for the polynomials after addition and multiplication, respectively.

Add(ct_a, ct_b). Let $\text{ct}_a = (a_0, a_1)$ and $\text{ct}_b = (b_0, b_1)$ have the same modulus Q , scale Δ , and secret key. Return $\text{ct}_+ = ([a_0 + b_0]_Q, [a_1 + b_1]_Q)$. The output has modulus Q , scale Δ , and polynomial $M_+ = M_a + M_b$. It represents $\mathbf{x}_a + \mathbf{x}_b$ with error $\mathbf{e}_a + \mathbf{e}_b$. Subtraction follows the same rule with a change of sign.

AddPlain($\text{ct}, \mathbf{a}$). To add a public vector $\mathbf{a} \in \mathbb{C}^r$, we encode $a_i/\sigma_i(p)$ in each slot, since the stored value is $x_i/\sigma_i(p)$. Compute $\mu_+ = \lfloor \Delta \sigma^{-1}((a_i/\sigma_i(p))_{i \in [r-1]}) \rfloor$. Return $\text{ct}_+ = ([c_0 + \mu_+]_Q, c_1)$, since adding μ_+ to c_0 increases the decryption expression $c_0 + c_1 s$ by μ_+ . The output retains modulus Q and scale Δ , with $M_+ = M + p\mu_+$. Up to the existing error and the error from encoding $\mathbf{a}$, it therefore represents the logical sum $\mathbf{x} + \mathbf{a}$.

MultPlain($\text{ct}, \mathbf{a}; \Delta_{\text{pt}}$). For a public vector $\mathbf{a}$ and a plaintext scale $\Delta_{\text{pt}} > 0$, compute $\mu_\times = \lfloor \Delta_{\text{pt}} \sigma^{-1}(\mathbf{a}) \rfloor$ and return $\text{ct}_\times = ([\mu_\times c_0]_Q, [\mu_\times c_1]_Q)$. Since $(x_j/\sigma_j(p)) \cdot a_j = (x_j a_j)/\sigma_j(p)$, the public vector is encoded without division by $\sigma_j(p)$. The output has modulus Q and scale $\Delta \Delta_{\text{pt}}$, with $M_\times = \mu_\times M$. It therefore represents the logical product $(x_j a_j)_{j \in [r-1]}$.

Mult_{evk}(ct_a, ct_b). Let the inputs represent $\mathbf{x}_a$ and $\mathbf{x}_b$ with modulus Q , secret key s , and scales Δ_a and Δ_b . First form the usual product with components $d_0 = [a_0 b_0]_Q$, $d_1 = [a_0 b_1 + a_1 b_0]_Q$, and $d_2 = [a_1 b_1]_Q$. Multiply every component by p , and return

$$\text{ct}_\times = \text{Relin}_{\text{evk}}([pd_0]_Q, [pd_1]_Q, [pd_2]_Q).$$

This correction gives $\sigma_j(p) \cdot (x_{a,j} x_{b,j} / \sigma_j(p)^2) = x_{a,j} x_{b,j} / \sigma_j(p)$ in slot j , preserving the required representation. The output has modulus Q and scale $\Delta_a \Delta_b$. If $\kappa \in R$ is the phase error added by relinearization, then its polynomial in (5) is $M_\times = M_a M_b + p\kappa$. Rescaling is a separate operation. The output therefore represents the logical product $(x_{a,j} x_{b,j})_j$.

RS_D(ct). Let $Q = DQ'$, where $D > 1$ is odd. The components c_i are already written as centered integer representatives modulo Q . Set $\rho_i = [c_i]_{D}$ and return

$\text{ct}' = (c'_0, c'_1)$, where $c'_i = [(c_i - \rho_i)/D]_{Q'}$ for $i \in \{0, 1\}$. For each coefficient $a = Dk + b$ with centered residue b , we have $|b| < D/2$, so $(a - b)/D = k = \lfloor a/D \rfloor$. Thus the formula divides each coefficient by D and rounds to the nearest integer. The output has modulus Q' and scale $\Delta' = \Delta/D$. The result is independent of the chosen representatives. Lemma 3 below shows that this standard rescaling procedure preserves our representation.

ModReduce $_{Q'}(\text{ct})$. For a divisor Q' of Q , return $\text{ct}' = ([c_0]_{Q'}, [c_1]_{Q'})$. The scale remains Δ , and the same polynomial M satisfies (5) modulo Q' . Recovery at this modulus requires $\|M\|_{\text{coef}, \infty} < Q'/2$.

Auto $_{\tau}(\text{ct})$. For a ring automorphism τ , applying τ to $p(c_0 + c_1 s) \equiv M \pmod{Q}$ gives $\tau(p)(\tau(c_0) + \tau(c_1)\tau(s)) \equiv \tau(M) \pmod{Q}$. The secret key and restoration factor become $\tau(s)$ and $\tau(p)$, respectively. To restore with p , choose a public $q_{\tau} \in R_Q$ satisfying $pq_{\tau} = \tau(p)$, when a solution exists, and return

$$\text{ct}' = \text{KS}_{\tau(s) \rightarrow s}([q_{\tau}\tau(c_0)]_Q, [q_{\tau}\tau(c_1)]_Q).$$

If key switching adds phase error $\kappa_{\tau} \in R$, its output satisfies $c'_0 + c'_1 s \equiv q_{\tau}(\tau(c_0) + \tau(c_1)\tau(s)) + \kappa_{\tau} \pmod{Q}$. Restoration multiplies this relation by p ; using $pq_{\tau} = \tau(p)$ gives $M' = \tau(M) + p\kappa_{\tau}$. One may take $q_{\tau} = \tau(p)p^{-1}$ when p is invertible modulo Q , or $q_{\tau} = 1$ when $\tau(p) = p$, even without invertibility. The output retains modulus Q and scale Δ and represents the permutation and possible conjugation of the logical slots induced by τ .

For a linear transformation L on the logical slots, write the stored message vector as $\mathbf{z} = U^{-1}\mathbf{x}$, where $U = \text{diag}(\sigma_0(p), \dots, \sigma_{r-1}(p))$. The desired output $L\mathbf{x}$ must be stored as $U^{-1}L\mathbf{x} = U^{-1}LU\mathbf{z}$. If $LU = UL$, this equals $L\mathbf{z}$, so the same map L can be applied directly to the stored values. This holds for sums and averages within groups whose values $\sigma_j(p)$ are equal, since their common factor $1/\sigma_j(p)$ can be taken outside each sum. Equal absolute values do not suffice, since signs and complex phases also affect the sum. In other words, $|\sigma_j(p)|$ control precision allocation, while the full values $\sigma_j(p)$ determine the corrections that preserve the logical computation.

If every rotation used to implement L fixes p , then these rotations use $q_{\tau} = 1$ and require no public correction.

The following lemma gives the exact polynomial after rescaling. It does not require p to be invertible or the input phase $c_0 + c_1 s$ to lie in the centered interval modulo Q . Its proof is given in Appendix D.

Lemma 3 (Preservation under rescaling). *Let $Q = DQ'$ with odd $D > 1$. Suppose an input at scale $\Delta > 0$ has a polynomial $M \in R$ satisfying $p(c_0 + c_1 s) \equiv M \pmod{Q}$ and $\sigma(M) = \Delta(\mathbf{x} + \mathbf{e})$. With ρ_0, ρ_1 and ct' as in RS_D , the polynomial*

$$M' = \frac{M - p(\rho_0 + \rho_1 s)}{D} \tag{6}$$

belongs to R and satisfies $p(c'_0 + c'_1 s) \equiv M' \pmod{Q'}$. At scale $\Delta' = \Delta/D$, we have $\sigma_j(M')/\Delta' = x_j + e'_j$ for every slot j , where $e'_j = e_j - \sigma_j(p)\sigma_j(\rho_0 + \rho_1 s)/(D\Delta')$.

For multiplication followed by rescaling, the lemma gives

$$M' = \frac{M_a M_b + p\kappa - p(\rho_0 + \rho_1 s)}{D}, \quad \Delta' = \frac{\Delta_a \Delta_b}{D}.$$

Both new error terms carry the factor p upon restoration. Again, p is a polynomial in R , so under σ this factor becomes $\sigma_j(p)$ in slot j . Bounds for these terms and for the propagation of existing errors are deferred to Section 3.3.

3.3 Error Analysis and Security

We bound the difference between the recovered value and the intended result in each slot, and give conditions that guarantee the required precision. At each computation node v , the ciphertext $\text{ct}_v = (c_{v,0}, c_{v,1})$ has modulus Q_v and scale Δ_v . Let $M_v \in R$ be the polynomial representing the message after restoration, with $M_v \equiv p(c_{v,0} + c_{v,1}s) \pmod{Q_v}$. For the intended value $x_{v,j}$, the error in slot j is $e_{v,j} = \sigma_j(M_v)/\Delta_v - x_{v,j}$. We write $e_{v,j} = \sigma_j(p)\eta_{v,j}$ and use bounds

$$|\eta_{v,j}| \leq B_{v,j}, \quad |x_{v,j}| \leq A_{v,j}.$$

Thus $\eta_{v,j}$ is the accumulated residual normalized by Δ_v , with the restoration factor $\sigma_j(p)$ kept outside. These quantities are used only for analysis; intermediate ciphertexts need not be decrypted.

For a fresh ciphertext encrypting $\mathbf{x}_v$, let $e_{\text{enc},v}$ and $e_{\text{dec},v}$ denote its encoding and decryption errors. At an evaluation step v , let κ_v denote the phase error introduced by relinearization or key switching, and let $\rho_{v,0}, \rho_{v,1}$ be the residues used in rescaling.⁴ For every slot j , we use bounds

$$\begin{aligned} |\sigma_j(e_{\text{enc},v})| &\leq B_{\text{enc},v}, & |\sigma_j(e_{\text{dec},v})| &\leq B_{\text{dec},v}, \\ |\sigma_j(\kappa_v)| &\leq K_v, & |\sigma_j(\rho_{v,0} + \rho_{v,1}s)| &\leq R_v. \end{aligned} \quad (7)$$

These bounds are taken before restoration and division by the message scale. In particular, the rescaling phase error is $-(\rho_{v,0} + \rho_{v,1}s)/D_v$, whose embeddings are bounded in absolute value by R_v/D_v . For the RNS method, K_v can be chosen using (14) in Appendix F. We assume that the bounds hold simultaneously for the actual operation inputs, including those after public corrections. For a fresh ciphertext, the triangle inequality gives the initial bound

$$|\eta_{v,j}| = \frac{|\sigma_j(e_{\text{enc},v} + e_{\text{dec},v})|}{\Delta_v} \leq \frac{B_{\text{enc},v} + B_{\text{dec},v}}{\Delta_v} =: B_{v,j}.$$

We analyze multiplication and rescaling together:

$$\text{ct}_v = \text{RS}_{D_v}(\text{Relin}_{\text{evk}}(p(\text{ct}_a \otimes \text{ct}_b))), \quad \Delta_v = \frac{\Delta_a \Delta_b}{D_v}. \quad (8)$$

Here $\otimes$ forms the usual product with three components, and p multiplies each component before relinearization. The inputs share a modulus and secret key, and $D_v > 1$ is an odd divisor of that modulus, as required by RS_{D_v} .

⁴ For input components c_i modulo $D_v Q_v$, we have $\rho_{v,i} = [c_i]_{D_v}$ and $c_{v,i} = [(c_i - \rho_{v,i})/D_v]_{Q_v}$ for $i \in \{0, 1\}$.

Lemma 4 (Error after multiplication and rescaling). *For (8), the logical value is $x_{v,j} = x_{a,j}x_{b,j}$, and its error satisfies*

$$|e_{v,j}| \leq |\sigma_j(p)| \left(A_{a,j}B_{b,j} + A_{b,j}B_{a,j} + |\sigma_j(p)|B_{a,j}B_{b,j} + \frac{K_v + R_v}{D_v\Delta_v} \right). \quad (9)$$

The expression in parentheses is a valid choice of $B_{v,j}$.

Proof. Lemma 3 gives $M_v = (M_aM_b + p\kappa_v - p(\rho_{v,0} + \rho_{v,1}s))/D_v$. Substitute $\sigma_j(M_t) = \Delta_t(x_{t,j} + \sigma_j(p)\eta_{t,j})$ for $t \in \{a,b\}$ and use $D_v\Delta_v = \Delta_a\Delta_b$. After division by Δ_v , expanding the product and subtracting $x_{a,j}x_{b,j}$ gives the first three terms in (9) after taking absolute values. The remaining term is bounded by $|\sigma_j(p)|(K_v + R_v)/(D_v\Delta_v)$. Since $\sigma_j(p) \neq 0$, the same inequality bounds $|\eta_{v,j}|$ by the expression in parentheses. $\square$

The first three terms propagate existing errors, while the last accounts for relinearization and rescaling in this step. The outer factor $|\sigma_j(p)|$ comes from restoration. Starting with the input bounds, we propagate $B_{v,j}$ in circuit order to obtain $B_{\text{out},j}$. For addition at a common scale, take $B_{v,j} = B_{a,j} + B_{b,j}$. Appendix E gives the remaining rules, including approximation of public constants and movement between slots. The accumulated bound can itself depend on p , so the final precision must be assessed using the full propagation.

In summary, recovery at the required precisions is guaranteed by the following precision and recovery conditions:

$$\begin{cases} |\sigma_j(p)|B_{\text{out},j} \leq \varepsilon_j, & \text{for each required output slot } j, \\ \frac{\|M_{\text{out}}\|_{\text{coef},\infty}}{\Delta_{\text{out}}} \leq \|\sigma^{-1}(\mathbf{x}_{\text{out}})\|_{\text{coef},\infty} + \frac{1}{r} \sum_{j=0}^{r-1} |\sigma_j(p)|B_{\text{out},j} < \frac{Q_{\text{out}}}{2\Delta_{\text{out}}}. \end{cases} \quad (10)$$

The second condition includes all slots, even those whose outputs are not required. As in Proposition 1, these conditions ensure that centered decryption of $p\text{ct}_{\text{out}}$ returns M_{out} , giving $|\hat{x}_{\text{out},j} - x_{\text{out},j}| \leq |\sigma_j(p)|B_{\text{out},j} \leq \varepsilon_j$ for each required output slot.

Next, we use the following proposition to establish security.

Proposition 2 (Security Reduction). *Fix the public parameters and p before CKKS key generation. If a PPT adversary $\mathcal{A}$ breaks the IND-CPA security of our construction, then a PPT adversary $\mathcal{B}_p$ breaks CKKS:*

$$\text{Adv}_{\text{ours},p,\text{evk}}^{\text{IND-CPA}}(\mathcal{A}) = \text{Adv}_{\text{CKKS},\text{evk}}^{\text{IND-CPA}}(\mathcal{B}_p).$$

Proof. The reduction $\mathcal{B}_p$ interacts with the CKKS challenger $\mathcal{C}$ as follows.

1. $\mathcal{B}_p$ receives (pk, evk) from $\mathcal{C}$ and gives (pk, evk, p) to $\mathcal{A}$.
2. $\mathcal{A}$ sends $(\mathbf{x}_0, \mathbf{x}_1)$ to $\mathcal{B}_p$, which computes

$$\mu_t = \left\lceil \left\lfloor \Delta \sigma^{-1}((x_{t,j}/\sigma_j(p))_j) \right\rfloor \right\rceil_Q, \quad t \in \{0, 1\},$$

and submits (μ_0, μ_1) to $\mathcal{C}$.

3. $\mathcal{C}$ returns $\text{ct} = \text{Enc}_{pk}(\mu_b)$ for its hidden bit $b \in \{0, 1\}$. $\mathcal{B}_p$ forwards ct to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$, and $\mathcal{B}_p$ returns the same b' to $\mathcal{C}$.

For each b , $\mathcal{A}$'s view has the same distribution as in our scheme. Hence the advantages are equal. $\square$

4 Construction of Ring Multipliers

4.1 From Precision Requirements to Embedding Weights

We determine target values for $|\sigma_j(p)|$ from the required precisions $\mathbf{b} = (b_j)_{j=0}^{r-1} \in \mathbb{R}_{\geq 0}^r$. With $\varepsilon_j := 2^{-b_j}$, the precision condition in (10) is $|\sigma_j(p)|B_{\text{out},j} \leq \varepsilon_j$.

For any $p \in R \setminus \{0\}$, $\prod_{j=0}^{r-1} |\sigma_j(p)|^2 = \text{Norm}_{K/\mathbb{Q}}(p) \geq 1$, with equality if and only if $p \in R^\times$. Thus $|\sigma_j(p)| < 1$ cannot hold for every j . For a unit $u \in R^\times$, $|\sigma_j(u)| < 1$ in some slots requires $|\sigma_j(u)| > 1$ in others, with $\prod_j |\sigma_j(u)| = 1$. If p is not a unit, then $\prod_j |\sigma_j(p)| > 1$. Thus the same reductions in some slots require greater overall compensation from the others than for a unit. We therefore choose a unit $u \in R^\times$ as the multiplier p .

Set $B := \max_{0 \leq j < r} B_{\text{out},j}$ and $\bar{b} := \sum_{j=0}^{r-1} b_j/r$. Choose target weights $w_j \propto \varepsilon_j$ for $|\sigma_j(u)|$. Normalizing them so that $\prod_j w_j = 1$ gives

$$w_j := 2^{\bar{b}-b_j}, \quad t_j := \log_2 w_j = \bar{b} - b_j.$$

If $B \leq 2^{-\bar{b}}$, the ideal error bounds satisfy $Bw_j \leq 2^{-b_j}$. For example, suppose half the slots require 60 bits of precision and the other half require 20 bits. Then $\bar{b} = 40$, giving logarithmic targets $(-20, 20)$ and weights $(2^{-20}, 2^{20})$, which at $B = 2^{-40}$ yield ideal error bounds $(2^{-60}, 2^{-20})$.

Set the target vector of logarithmic weights $\mathbf{t} := (t_j)_{j=0}^{r-1}$. Since $\sum_j t_j = 0$, we have $\mathbf{t} \in H$ and seek $u \in R^\times$ with $\text{Log}(u) \approx \mathbf{t}$. The lattice Λ is discrete, so exact equality need not be possible. The next subsection gives a construction and bounds its deviation from $\mathbf{t}$.

4.2 Explicit Construction via Fourier Method

We construct $u \in R^\times$ from the target $\mathbf{t} \in H$. Let $r \geq 2$ be a power of two and $\zeta := e^{2\pi i/(4r)}$. Define the automorphism $\tau : K \rightarrow K$ by $\tau(\zeta) = \zeta^5$. Use the slot order $\sigma_j(\zeta) = \zeta^{5^j}$, $0 \leq j < r$, and take subscripts modulo r . Choose the unit

$$v := \zeta^{-2}c_5 = \zeta^{-2} + \zeta^{-1} + 1 + \zeta + \zeta^2.$$

The symmetric expression makes every $\sigma_j(v)$ real. Since ζ^{-2} is a root of unity, $\text{Log}(v) = \text{Log}(c_5)$. The ratio $c_5 = (1 - \tau(\zeta))/(1 - \zeta)$ matches τ , so common factors cancel when we multiply the conjugates, as shown below.

Set $\ell_j := \log_2 |\sigma_j(v)|$. Since $\sigma_j(\tau^{-k}(v)) = \sigma_{j-k}(v)$, products of conjugates give the circular convolution

$$u(\mathbf{n}) := \prod_{k=0}^{r-1} \tau^{-k}(v)^{n_k}, \quad \mathbf{n} \in \mathbb{Z}^r, \quad \log_2 |\sigma_j(u(\mathbf{n}))| = \sum_{k=0}^{r-1} \ell_{j-k} n_k. \quad (11)$$

Negative exponents are allowed because every factor is a unit. We use the Fourier transform to solve the convolution equation. For any vector $\mathbf{z} = (z_j)_{j=0}^{r-1} \in \mathbb{C}^r$, define $\widehat{z}_k := \sum_{j=0}^{r-1} z_j e^{-2\pi i j k / r}$, with inverse $z_j = \frac{1}{r} \sum_{k=0}^{r-1} \widehat{z}_k e^{2\pi i j k / r}$. The following lemma ensures that the convolution can realize every target in H with real coefficients.

Lemma 5 (Spanning by logarithms of conjugates). *The vectors $\text{Log}(\tau^k(v))$, $0 \leq k < r$, span H over $\mathbb{R}$. Consequently, $\widehat{\ell}_0 = 0$ and $\widehat{\ell}_k \neq 0$ for $1 \leq k < r$.*

Proof. For $1 \leq h < r$, $\prod_{k=0}^{h-1} \tau^k(v) = \zeta^{(1-5^h)/2} c_{5^h}$, hence $\sum_{k=0}^{h-1} \text{Log}(\tau^k(v)) = \text{Log}(c_{5^h})$. Corollary 1 and $\tau^k(v) \in R^\times$ therefore give $\text{span}_{\mathbb{R}}\{\text{Log}(\tau^k(v)) : 0 \leq k < r\} = H$. The convolution matrix $L = (\ell_{j-k})_{0 \leq j, k < r}$ has columns $\text{Log}(\tau^{-k}(v))$. Since $\tau^r = \text{id}$, these span H , so $\text{rank } L = r - 1$. Fourier diagonalization gives $\text{diag}(\widehat{\ell}_0, \dots, \widehat{\ell}_{r-1})$ with the same rank. The rank of a diagonal matrix equals the number of its nonzero diagonal entries, so exactly one Fourier coefficient is zero. Since $\widehat{\ell}_0 = \sum_j \ell_j = \log_2 \prod_j |\sigma_j(v)| = 0$, we have $\widehat{\ell}_k \neq 0$ for $1 \leq k < r$. $\square$

Theorem 2 (Construction and rounding bound). *For $\mathbf{t} \in H$, take the inverse Fourier transform $\mathbf{a} \in \mathbb{R}^r$ of $\widehat{a}_0 := 0$ and $\widehat{a}_k := \widehat{t}_k / \widehat{\ell}_k$ for $1 \leq k < r$. Set $n_j := \lfloor a_j \rfloor$ and form $u(\mathbf{n})$ as in (11). Then $u(\mathbf{n}) \in R^\times$ and*

$$\|\text{Log}(u(\mathbf{n})) - \mathbf{t}\|_\infty \leq \frac{1}{2} \sum_{j=0}^{r-1} |\ell_j|.$$

Proof. Conjugate symmetry of the transforms of the real vectors $(t_j)_j$ and $(\ell_j)_j$ makes $\mathbf{a}$ real. Since $\widehat{t}_0 = 0$, Lemma 5 and Fourier inversion give $\sum_k \ell_{j-k} a_k = t_j$. Using $|n_k - a_k| \leq 1/2$, we obtain $|\log_2 |\sigma_j(u(\mathbf{n}))| - t_j| = |\sum_k \ell_{j-k} (n_k - a_k)| \leq \sum_k |\ell_{j-k}| |n_k - a_k| \leq \frac{1}{2} \sum_{k=0}^{r-1} |\ell_k|$. The product defining $u(\mathbf{n})$ is a unit. $\square$

The bound gives $2^{t_j - \frac{1}{2} \sum_{k=0}^{r-1} |\ell_k|} \leq |\sigma_j(u(\mathbf{n}))| \leq 2^{t_j + \frac{1}{2} \sum_{k=0}^{r-1} |\ell_k|}$. It measures the deviation of the weights from their targets.

For periodic targets $t_{j+d} = t_j$, where $d \geq 2$ divides r , apply the same construction in $\mathbb{Z}[\zeta_{4d}]$ with $\zeta_{4d} := \zeta^{r/d}$. Use $(t_0, \dots, t_{d-1})$ as the target, replacing r and ζ by d and ζ_{4d} in the construction and its bound. This target also has coordinate sum zero. The resulting unit satisfies $\sigma_{j+d}(u) = \sigma_j(u)$, giving d groups of r/d slots with deviation bound $\frac{1}{2} \sum_{j=0}^{d-1} |\ell_j|$, using the subfield values of ℓ_j . Slots in each group retain independent messages.

For example, with $d = 2$, set $h := \zeta_8 + \zeta_8^{-1}$. Then $h^2 = 2$, and the chosen unit is $v = 1 + h$, with $v^{-1} = h - 1$. Since $\sigma_j(h) = (-1)^j \sqrt{2}$, the multiplier $u = v^k$ has weights (α^k, α^{-k}) on the even and odd slots, where $\alpha := 1 + \sqrt{2}$. For a target (t_0, t_1) with $t_1 = -t_0$, choose $k := \lfloor t_0 / \log_2 \alpha \rfloor$. Rounding this single exponent gives the tighter deviation bound $\frac{1}{2} \log_2 \alpha \approx 0.636$ bits for either logarithmic weight.

5 Precision Allocation under Recovery Constraints

Precision targets concern the outputs that are used, whereas recovery of M_{out} depends on errors in every slot. Can selected slots gain arbitrarily much precision by allowing larger errors elsewhere? We use the recovery condition in (10) to give a range of allocations that guarantees correct recovery.

5.1 Effects of Precision Differences on Error

Consider a unit u with weights $2^{-\theta}$ on half the slots and 2^θ on the other half, where $\theta \geq 0$. Use a common bound $B > 0$ with $B_{\text{out},j} \leq B$ for every j . The restored error bounds are $2^{-\theta}B$ and $2^\theta B$, so their corresponding precision bounds differ by 2θ bits. Each required output must also satisfy $|\sigma_j(u)|B_{\text{out},j} \leq \varepsilon_j$. A dummy output has no precision target, but its error still enters the recovery condition.

To see how θ affects the polynomial coefficients, define the average embedding weight

$$\bar{w}(u) := \frac{1}{r} \sum_{j=0}^{r-1} |\sigma_j(u)|.$$

The coefficient error term in (10) satisfies $\frac{1}{r} \sum_j |\sigma_j(u)|B_{\text{out},j} \leq B\bar{w}(u)$. For the two equal groups, $\bar{w}(u) = (2^{-\theta} + 2^\theta)/2$. It equals 1 at $\theta = 0$ and increases with θ . Thus increasing θ reduces the error bound in one group but increases the common bound $B\bar{w}(u)$ on the coefficient error. For large θ , this bound is approximately $B2^{\theta-1}$. The following proposition states when it still guarantees recovery.

Proposition 3 (Allocation bound for two levels). *For the two groups above, set $C := \frac{2}{B} \left(\frac{Q_{\text{out}}}{2\Delta_{\text{out}}} - \|\sigma^{-1}(\mathbf{x}_{\text{out}})\|_{\text{coef},\infty} \right)$. If $C > 2$ and*

$$0 \leq \theta < \log_2 \left(\frac{C + \sqrt{C^2 - 4}}{2} \right), \quad (12)$$

then centered decryption of uct_{out} returns the entire polynomial M_{out} .

The proof is given in Appendix H.

At fixed $Q_{\text{out}}, \Delta_{\text{out}}, \mathbf{x}_{\text{out}}$ and B , increasing θ eventually makes $\bar{w}(u) \geq C/2$. Thus this bound cannot guarantee arbitrarily large precision differences. Exceeding (12) removes the guarantee; it does not by itself prove that decryption fails. Keeping θ and the equal group proportions unchanged, increasing N provides more slots at each weight, with $N/4$ slots assigned the smaller weight $2^{-\theta}$.

For multiple levels, $\bar{w}(u)$ also records the fraction of slots assigned each weight. If $|\log_2 |\sigma_j(u)|| \leq \theta$ for every j , then $1 \leq \bar{w}(u) \leq (2^{-\theta} + 2^\theta)/2$. Both inequalities follow from convexity of 2^x and $\sum_j \log_2 |\sigma_j(u)| = 0$; the upper bound uses the line joining the values at $-\theta$ and θ for $\theta > 0$. The two equal groups attain the upper bound, whereas other distributions can give a smaller average

at the same maximum logarithmic deviation θ . Thus $\bar{w}(u) < C/2$ can guarantee recovery even when the uniform condition (12) does not hold.

These comparisons keep B fixed. For any evaluated circuit, the more precise contribution is $\frac{1}{r} \sum_j |\sigma_j(u)| B_{\text{out},j}$ in (10), using the bounds propagated for the actual unit and parameters.

5.2 Allocation with Dummy Slots

When a task needs outputs at different precisions, both groups can produce useful results. If all useful outputs require high precision, we can instead fill some slots with dummy values and assign smaller weights to the useful slots. Here N is fixed, so dummy values occupy some of the existing $r = N/2$ slots. Even if a dummy output has ideal value $x_{\text{out},j} = 0$, its actual embedding $\sigma_j(M_{\text{out}}) = \Delta_{\text{out}} \sigma_j(u) \eta_{\text{out},j}$ can be nonzero and must be included in (10).

To choose target weights, let $\mathcal{I}, \mathcal{D}$ be nonempty sets of useful and dummy slots that partition the r slots. Fix their sizes, the useful targets t_i for $i \in \mathcal{I}$, and a common residual bound B . The unit constraint requires $\sum_{j \in \mathcal{D}} t_j = -\sum_{i \in \mathcal{I}} t_i$. By convexity of 2^t , the dummy contribution $(B/r) \sum_{j \in \mathcal{D}} 2^{t_j}$ is smallest when all dummy targets equal their average, namely $t_j = -\sum_{i \in \mathcal{I}} t_i / |\mathcal{D}|$. If every useful target is $t_i = -\theta$, this gives $t_j = \theta |\mathcal{I}| / |\mathcal{D}|$ on the dummy slots and hence two levels. Equal group sizes recover the targets $-\theta, \theta$ above. This therefore minimizes the common bound for target weights.

For independent computations, a batch is a set of inputs packed into slots and processed by one circuit evaluation. Dummy slots reduce the number of useful inputs per batch, so the same task may require more batches. These batches can run in parallel, so their number need not increase the running time.

6 From Precision Allocation to Resource Savings

We next show that our precision allocation method can allow smaller scales Δ_v and ciphertext moduli Q_v at the required precision. We give conditions for these reductions and compare their savings with the costs of public corrections.

Ordinary CKKS ($p = 1$) and our construction ($p = u$) choose their own N, Δ_v, Q_v, P for the same logical circuit, input bounds $A_{v,j}$, precision targets 2^{-b_j} , and number of useful inputs. Both use the same security target, secret and error distributions, and failure probability bound.

6.1 Scale and Modulus Reduction

We give sufficient conditions for smaller scales and show when they lead to a smaller initial modulus. For a fresh ciphertext, a weight $|\sigma_j(u)| < 1$ permits a smaller scale because $|\sigma_j(u)| / (|\sigma_j(u)| \Delta) = 1/\Delta$. With common encoding and decryption bounds $B_{\text{enc}}, B_{\text{dec}}$, ordinary CKKS at scale Δ and our method at scale $|\sigma_j(u)| \Delta$ therefore have the same restored error bound $(B_{\text{enc}} + B_{\text{dec}})/\Delta$.

Multiplication introduces further errors and can limit this reduction. Superscripts o, u denote ordinary CKKS and our method, respectively. At each node v , define the actual scale reduction factor $\gamma_v := \Delta_v^o / \Delta_v^u$. We compare the same circuit at the same N , using additions and multiplications within each slot and rescaling after every multiplication. In each evaluation, addition inputs have equal scales, operation inputs have the same modulus, and each rescaling divisor satisfies $D_v > 1$, $D_v \equiv 1 \pmod{2}$, and $D_v \mid Q$, where Q is its input modulus. For a multiplication node v with inputs a, b , the scale updates give $D_v^u = D_v^o \gamma_v / (\gamma_a \gamma_b)$.

Use common encoding and decryption bounds and a common relinearization bound K_v from (7). The rescaling bounds R_v^o, R_v^u use the respective divisors D_v^o, D_v^u . If both secret keys satisfy $\|s\|_{\sigma, \infty} \leq B_s$, exact rescaling permits the common estimate $R_v/D_v \leq N(1 + B_s)/2$; see Appendix E. The new error term in Lemma 4 satisfies

$$|\sigma_j(u)| \frac{K_v + R_v^u}{D_v^u \Delta_v^u} \leq |\sigma_j(u)| \left(\frac{\gamma_a \gamma_b K_v}{\Delta_a^o \Delta_b^o} + \frac{\gamma_v N(1 + B_s)}{2 \Delta_v^o} \right). \quad (13)$$

Relative to ordinary CKKS, the relinearization and rescaling bounds have factors $|\sigma_j(u)| \gamma_a \gamma_b$ and $|\sigma_j(u)| \gamma_v$, respectively. Both methods propagate errors using these common estimates and the same value bounds $A_{v,j}$.

Proposition 4 (Scale reduction). *For the two evaluations above, fix a slot j with $|\sigma_j(u)| < 1$. If*

$$1 \leq \gamma_v \leq |\sigma_j(u)|^{-1/2} \quad \text{for every node } v,$$

then $|\sigma_j(u)| B_{v,j}^u \leq B_{v,j}^o$ at every node. The scale at node v decreases by $\log_2 \gamma_v$ bits, at most $-\frac{1}{2} \log_2 |\sigma_j(u)|$ under this condition. The reduced parameters must also satisfy the precision and recovery conditions in (10).

Proof. The fresh input bounds have factor $|\sigma_j(u)| \gamma_v \leq 1$. At each multiplication, both $|\sigma_j(u)| \gamma_a \gamma_b \leq 1$ and $|\sigma_j(u)| \gamma_v \leq 1$, so the new error bounds in (13) do not increase. Addition and the propagated terms in Lemma 4 are nondecreasing in the restored input bounds; induction gives the stated comparison. Finally, $\log_2 \Delta_v^o - \log_2 \Delta_v^u = \log_2 \gamma_v \leq -\frac{1}{2} \log_2 |\sigma_j(u)|$. $\square$

A smaller relinearization bound can allow still smaller scales. Let $K_v^o, K_v^u > 0$ be the bounds obtained from each method's P and decomposition. This error contribution does not increase if $|\sigma_j(u)| \gamma_a \gamma_b K_v^u \leq K_v^o$. For squaring, $a = b$, and this condition becomes $\Delta_a^u / \Delta_a^o \geq \sqrt{|\sigma_j(u)| K_v^u / K_v^o}$. If $K_v^u < K_v^o$, this threshold is strictly below $\sqrt{|\sigma_j(u)|}$ from the common bound, permitting a smaller input scale. The other error constraints still apply.

The recovery condition also limits the modulus reduction. Condition (10) requires every coefficient of $M_{\text{out}} / \Delta_{\text{out}}$ to have absolute value below $Q_{\text{out}} / (2 \Delta_{\text{out}})$. Larger errors in other slots can require a larger ratio $Q_{\text{out}} / \Delta_{\text{out}}$ to satisfy this condition, even when the bounds $|\sigma_j(u)| B_{\text{out},j}$ meet all required precision targets.

To compare initial moduli, consider $d \geq 1$ successive squarings with rescaling and no other modulus reductions. For either method, let Q_t, Δ_t denote the modulus and scale after step t , starting from Q_0, Δ_0 . The updates $D_t = \Delta_{t-1}^2/\Delta_t$ and $Q_{t-1} = D_t Q_t$ give $Q_0 = (Q_d/\Delta_d)\Delta_0^2 \prod_{t=1}^{d-1} \Delta_t$. Applying this identity to both methods yields

$$Q_0^u < Q_0^o \iff \frac{Q_d^u/\Delta_d^u}{Q_d^o/\Delta_d^o} < \gamma_0^2 \prod_{t=1}^{d-1} \gamma_t.$$

The fraction compares the final moduli relative to scale; the product is the modulus reduction factor from the smaller scales if Q_d/Δ_d stays fixed. At fixed scales, increasing Q_d increases Q_0 by the same factor. Thus a larger final ratio reduces the savings from the scales. For example, if $\gamma_0^2 \prod_{t=1}^{d-1} \gamma_t \approx 16$ and $(Q_d^u/\Delta_d^u)/(Q_d^o/\Delta_d^o) \approx 4$, then $Q_0^u \approx Q_0^o/4$. This comparison uses the actual scales determined by the RNS primes. Other circuits use their own scale and modulus updates. The next subsection compares computation and storage using the selected prime counts.

6.2 Computation and Storage Costs

Smaller moduli may reduce the number of RNS primes, while public corrections add modular products. We compare costs using each method's own parameters that meet the same precision targets for the required output slots and the same security requirement.

Computational Cost Recall from Section 2.3 the ring degree N , the moduli Q_v, P , their prime counts L_{Q_v}, L_P , and the block count $\beta_v = \lceil L_{Q_v}/L_P \rceil$.

Table 1. Differences in modular product counts between original CKKS and our method. The moduli Q, P and their prime counts L_Q, L_P may differ between methods.

Operation	Original CKKS	Our Method
Mult	$N^o(6L_Q^o + 2L_{Q'}^o + 2\beta^o(L_Q^o + L_P^o))$	$N^u(9L_Q^u + 2L_{Q'}^u + 2\beta^u(L_Q^u + L_P^u))$
Auto ($q_\tau \neq 1$)	$2N^o(L_Q^o + \beta^o(L_Q^o + L_P^o))$	$2N^u(2L_Q^u + \beta^u(L_Q^u + L_P^u))$
Restore	0	$2N^u L_{Q_{\text{out}}}^u$

Table 1 lists only operations with different counting formulas. All operands and outputs use NTT form. Ciphertext multiplication includes relinearization and one rescaling from Q to $Q' = Q/D$, with each method's own moduli and decomposition. The counts for our method include the public corrections; conditions on q_τ refer to our method.

In NTT form, multiplying one component by a public ring element uses N^u modular products per prime. Multiplication produces three components, each multiplied by u before relinearization, giving $3N^u L_Q^u$ additional modular products. An automorphism multiplies its two components by q_τ before key

switching, adding $2N^u L_Q^u$ modular products when $q_\tau \neq 1$. The factors u and q_τ introduce no additional encoding scale, so these corrections preserve the scale and need no extra rescaling.

The remaining operations have unchanged costs at equal parameters. For $h \in \{o, u\}$, addition uses $2N^h L_Q^h$ modular additions, and MultPlain uses $2N^h L_Q^h$ modular products; neither requires transforms or RNS conversions. Auto with $q_\tau = 1$ uses $2N^h(L_Q^h + \beta^h(L_Q^h + L_P^h))$ modular products. All automorphisms also permute $2N^h L_Q^h$ entries.

Public corrections add no transforms or RNS conversions. Mult uses $(\beta^h + 2)(L_Q^h + L_P^h) + 2L_Q^h$ transforms and $\beta^h + 4$ RNS conversions; Auto uses $(\beta^h + 2)(L_Q^h + L_P^h)$ transforms and $\beta^h + 2$ conversions. MultPlain and Restore use neither. These formulas are evaluated with each method's own parameters, so the resulting counts may differ. Squaring reuses the product $c_0 c_1$ and requires $N^h L_Q^h$ fewer modular products than Mult, with all other counts unchanged.

For the d successive squarings above, sum the adjusted modular product counts at each step's modulus and add final restoration. Within our total, the public corrections and restoration account for

$$3N^u \sum_{t=1}^d L_{Q_{t-1}}^u + 2N^u L_{Q_d}^u.$$

Storage Cost For storage, we count uncompressed residue coefficients in 64 bit words, with eight bytes per coefficient. A ciphertext with two components at modulus Q and one switching key with β polynomial pairs at modulus QP require, respectively,

$$S_{\text{ct}} = 16NL_Q \text{ bytes}, \quad S_{\text{evk}} = 16N\beta(L_Q + L_P) \text{ bytes}.$$

For a public key stored as two polynomials modulo QP , the corresponding size is $16N(L_Q + L_P)$ bytes.

At fixed N , a smaller Q may allow fewer RNS primes and therefore smaller ciphertexts. If L_Q remains unchanged, their storage in machine words is unchanged. Changing L_P affects both β and the width $L_Q + L_P$ of each key component. Key size can therefore change discontinuously, and the parameters with the smallest keys need not give the shortest running time.

Our construction of u also requires extra computation and storage. We count constructing u , computing $\sigma_j(u)$, and preparing public NTT representations separately from homomorphic evaluation. Each stored NTT polynomial modulo Q uses $8NL_Q$ bytes. We also count the storage for the description of u and the embeddings $(\sigma_j(u))_j$ used by the encoder.

7 Implementation

References

1. Bae, Y., Cheon, J.H., Cho, W., Kim, J., Kim, T.: META-BTS: Bootstrapping precision beyond the limit. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). ACM (2022), <https://eprint.iacr.org/2022/1167>
2. Bian, S., Fu, Y., Shen, R., Pan, H., Wang, A., Guan, Z.: High-precision functional bootstrapping for CKKS from fourier extension. Cryptology ePrint Archive, Paper 2026/367 (2026), <https://eprint.iacr.org/2026/367>, eUROCRYPT 2026; revised 2026-03-09
3. Bootland, C., Castryck, W., Iliashenko, I., Vercauteren, F.: Efficiently processing complex-valued data in homomorphic encryption. Journal of Mathematical Cryptology **14**(1), 55–65 (2020). <https://doi.org/10.1515/jmc-2015-0051>
4. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: CHIMERA: Combining ring-LWE-based fully homomorphic encryption schemes. Journal of Mathematical Cryptology **14**(1), 316–338 (2020). <https://doi.org/10.1515/jmc-2019-0026>
5. Chen, H., Iliashenko, I., Laine, K.: When HEAAN meets FV: a new somewhat homomorphic encryption with reduced memory overhead. In: Cryptography and Coding (IMACC 2021). Lecture Notes in Computer Science, vol. 13129, pp. 265–285. Springer (2021). https://doi.org/10.1007/978-3-030-92641-0_13
6. Chen, H., Laine, K., Player, R., Xia, Y.: High-precision arithmetic in homomorphic encryption. In: Topics in Cryptology – CT-RSA 2018. pp. 116–136. Springer (2018). https://doi.org/10.1007/978-3-319-76953-0_7
7. Cheon, J.H., Cho, W., Kim, J., Stehlé, D.: Homomorphic multiple precision multiplication for CKKS and reduced modulus consumption. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). ACM (2023). <https://doi.org/10.1145/3576915.3623086>
8. Cheon, J.H., Choe, H., Kang, M., Kim, J., Kim, S., Mono, J., Noh, T.: Grafting: Decoupled scale factors and modulus in RNS-CKKS. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). ACM (2025), <https://eprint.iacr.org/2024/1014>
9. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A full RNS variant of approximate homomorphic encryption. In: Selected Areas in Cryptography – SAC 2018. Lecture Notes in Computer Science, vol. 11349, pp. 347–368. Springer (2019). https://doi.org/10.1007/978-3-030-10970-7_16, <https://eprint.iacr.org/2018/931>
10. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology – ASIACRYPT 2017. pp. 409–437. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_15
11. Cramer, R., Ducas, L., Peikert, C., Regev, O.: Recovering short generators of principal ideals in cyclotomic rings. In: Advances in Cryptology – EUROCRYPT 2016. Springer (2016), <https://eprint.iacr.org/2015/313>
12. Dumezy, J., Suvanto, E.: Character block encodings for discrete CKKS: Single-level LUTs and low-depth arithmetic. Cryptology ePrint Archive, Paper 2026/1200 (2026), <https://eprint.iacr.org/2026/1200>, revised 2026-09-02
13. Geelen, R., Vercauteren, F.: Fully homomorphic encryption for cyclotomic prime moduli. In: Advances in Cryptology – EUROCRYPT 2025. Lecture Notes in Computer Science, vol. 15603, pp. 366–397. Springer (2025). https://doi.org/10.1007/978-3-031-91131-6_13, <https://eprint.iacr.org/2024/1587>
14. Kim, A., Papadimitriou, A., Polyakov, Y.: Approximate homomorphic encryption with reduced approximation error. In: Topics in Cryptology – CT-RSA 2022. Lec-

- ture Notes in Computer Science, vol. 13161, pp. 120–144. Springer (2022). https://doi.org/10.1007/978-3-030-95312-6_6, <https://eprint.iacr.org/2020/1118>
15. Lee, Y., Cheon, S., Kim, D., Lee, D., Kim, H.: ELASM: Error-latency-aware scale management for fully homomorphic encryption. In: 32nd USENIX Security Symposium. pp. 4697–4714. USENIX Association (2023), <https://www.usenix.org/conference/usenixsecurity23/presentation/lee-yongwoo>
 16. Lee, Y., Heo, S., Cheon, S., Jeong, S., Kim, C., Kim, E., Lee, D., Kim, H.: HECATE: Performance-aware scale optimization for homomorphic encryption compiler. In: IEEE/ACM International Symposium on Code Generation and Optimization (CGO). pp. 193–204 (2022). <https://doi.org/10.1109/CGO53902.2022.9741265>
 17. Sung, H., Seo, S., Kim, T., Min, C.: EvalRound+ bootstrapping and its rigorous analysis for CKKS scheme. Cryptology ePrint Archive, Paper 2024/1379 (2024), <https://eprint.iacr.org/2024/1379>
 18. Tune Insight SA: Lattigo: Lattice-based multiparty homomorphic encryption in Go. Software, version 6.2.0 (2026), <https://github.com/tuneinsight/lattigo/tree/v6.2.0>
 19. Washington, L.C.: Introduction to Cyclotomic Fields, Graduate Texts in Mathematics, vol. 83. Springer, New York, 2 edn. (1997). <https://doi.org/10.1007/978-1-4612-1934-7>

A Proof of Lemma 2

Proof. In ordinary CKKS, the triangle inequality guarantees no wraparound if $\Delta \|\sigma^{-1}(\mathbf{x})\|_{\text{coef},\infty} + \|e_{\text{enc}} + e_{\text{dec}}\|_{\text{coef},\infty} < Q/2$. By (3), restoration replaces $e_{\text{enc}} + e_{\text{dec}}$ with $p(e_{\text{enc}} + e_{\text{dec}})$ while leaving the message term unchanged, giving the stated condition and decoded values. $\square$

B Proof of Proposition 1

Proof. Applying (2) to $f = p(e_{\text{enc}} + e_{\text{dec}})$ and using the assumed embedding bounds yields

$$\begin{aligned} \|p(e_{\text{enc}} + e_{\text{dec}})\|_{\text{coef},\infty} &\leq \frac{1}{r} \sum_{j=0}^{r-1} |\sigma_j(p)| |\sigma_j(e_{\text{enc}} + e_{\text{dec}})| \\ &\leq \frac{\Delta}{r} \sum_{j=0}^{r-1} |\sigma_j(p)| B_j \leq \frac{\Delta}{r} \sum_{j=0}^{r-1} \varepsilon_j. \end{aligned}$$

Under (4), this implies the range condition in Lemma 2, which therefore gives the decoded values. Their errors satisfy $|\hat{x}_j - x_j| \leq |\sigma_j(p)| B_j \leq \varepsilon_j$ by the same embedding bounds. $\square$

C Modulus Length under Equivalent Encoding

Proposition 5 (Modulus length under equivalent encoding). *Fix the ring, p , and a circuit evaluated as in Section 3.2. With identical rounding and exact scale bookkeeping, the rewriting $(\Delta_t, \sigma_j(p)) \mapsto (\Delta_t/\lambda, \sigma_j(p)/\lambda)$ ⁵ for a common $\lambda > 0$ leaves the feasible modulus chains and their total lengths unchanged.*

Proof. For any feasible original execution, use the same input modulus, keys, and randomness. Tildes denote the rewritten form, with $\tilde{\Delta}_t = \Delta_t/\lambda$. Input and additive-plaintext encodings give $\tilde{\mu} = \mu$, since $(\Delta/\lambda)(\lambda x_j/\sigma_j(p)) = \Delta x_j/\sigma_j(p)$ and rounding is identical. The correction p has plaintext scale λ , so the rescaling divisor is

$$\tilde{D} = \frac{\lambda \tilde{\Delta}_a \tilde{\Delta}_b}{\tilde{\Delta}_c} = \frac{\lambda(\Delta_a/\lambda)(\Delta_b/\lambda)}{\Delta_c/\lambda} = \frac{\Delta_a \Delta_b}{\Delta_c} = D.$$

For the remaining rules, retain the multiplicative plaintext scale Δ_{pt} and the same public corrections. Addition preserves scale alignment because $\tilde{\Delta}_a = \tilde{\Delta}_b \iff \Delta_a = \Delta_b$. Plaintext multiplication and rescaling preserve the correspondence: $\tilde{\Delta}_t \Delta_{\text{pt}} = (\Delta_t \Delta_{\text{pt}})/\lambda$ and $\tilde{\Delta}_t/D = (\Delta_t/D)/\lambda$. Automorphisms and linear maps use $\tilde{q}_r = q_r$ and $(U/\lambda)^{-1} L(U/\lambda) = U^{-1} L U$. Induction on the integer operations gives $(\tilde{\text{ct}}_t, \tilde{Q}_t, \lambda \tilde{\Delta}_t) = (\text{ct}_t, Q_t, \Delta_t)$.

⁵ The identity $\sigma_j(p/\lambda) = \sigma_j(p)/\lambda$ holds for every $\lambda > 0$, even when $p/\lambda \notin R$.

Recovery then gives $\tilde{x}_j = \sigma_j(\text{Dec}_s(\text{pct}_{\text{out}}))/(\lambda\tilde{\Delta}_{\text{out}}) = \hat{x}_j$. The correspondence is reversible. For a chain $Q_0, \dots, Q_L$ with $D_t = Q_{t-1}/Q_t$,

$$\begin{aligned}\log_2 \tilde{Q}_0 &= \log_2 \tilde{Q}_L + \sum_{t=1}^L \log_2 \tilde{D}_t \\ &= \log_2 Q_L + \sum_{t=1}^L \log_2 D_t = \log_2 Q_0.\end{aligned}$$

□

D Proof of Lemma 3

Proof. Using the integer representatives $c_0, c_1 \in R$, the hypothesis gives an $I \in R$ such that

$$p(c_0 + c_1 s) = M + DQ'I.$$

Reducing modulo D yields $M \equiv p(\rho_0 + \rho_1 s) \pmod{D}$, so the numerator in (6) is divisible by D in R . Subtracting $p(\rho_0 + \rho_1 s)$ from the displayed identity and dividing by D gives

$$p\left(\frac{c_0 - \rho_0}{D} + \frac{c_1 - \rho_1}{D}s\right) = M' + Q'I.$$

Reduction modulo Q' proves the claimed congruence. Replacing c_i by $c_i + Qh_i$ leaves ρ_i unchanged and changes the quotient by $Q'h_i$, so the output does not depend on the representatives. Finally, apply σ_j to (6) and divide by $\Delta' = \Delta/D$. Using $\sigma_j(M) = \Delta(x_j + e_j)$ and $D\Delta' = \Delta$ gives the stated identity. □

E Additional Error Propagation

We use the notation and primitive bounds of Section 3.3. The following rules give bounds at output node v from the bounds at its input nodes. All scales follow the exact updates in Section 3.2.

AddPlain(ct_a, **u**). For addition of a public vector **u**, write $\mu_+ = \Delta_a \sigma^{-1}((u_j/\sigma_j(p))_j) + e_+$ and let $\overline{B_{+,v}}$ bound $|\sigma_j(e_+)|$ in every slot. Since $M_v = M_a + p\mu_+$, one may take $B_{v,j} = B_{a,j} + B_{+,v}/\Delta_a$.

MultPlain(ct_a, **u**; Δ_{pt}). For multiplication by a public vector **u** encoded at scale Δ_{pt} , define its encoding discrepancy $\delta_j = \sigma_j(\mu_{\times})/\Delta_{\text{pt}} - u_j$ and choose $\beta_j \geq |\delta_j|$. The identity $M_v = \mu_{\times} M_a$ gives $\eta_{v,j} = (u_j + \delta_j)\eta_{a,j} + x_{a,j}\delta_j/\sigma_j(p)$, hence

$$B_{v,j} = (|u_j| + \beta_j)B_{a,j} + \frac{A_{a,j}\beta_j}{|\sigma_j(p)|}.$$

Thus approximation of the public multiplier contributes an error whose restored bound includes $A_{a,j}\beta_j$.

Mult_{evk}(ct_a, ct_b). For multiplication without rescaling, the same expansion as in the proof of Lemma 4 gives (9) with $D_v = 1$ and $R_v = 0$. Rescaling already included in that lemma is not counted again.

RS_{D_v}(ct_a). For rescaling alone, Lemma 3 gives $B_{v,j} = B_{a,j} + R_v/(D_v \Delta_v)$.

ModReduce_{Q'}(ct_a). Modulus reduction leaves $B_{v,j} = B_{a,j}$, since M_a and the scale remain unchanged; the smaller modulus must still satisfy the range condition at recovery.

Auto_τ(ct_a). For Auto_τ, assume the public correction in Section 3.2 exists. If τ moves input slot k to output slot j , possibly with conjugation, the identity $M_v = \tau(M_a) + p\kappa_v$ gives

$$B_{v,j} = \frac{|\sigma_k(p)|}{|\sigma_j(p)|} B_{a,k} + \frac{K_v}{\Delta_v}.$$

Conjugation preserves absolute values, so the same bound applies in both cases. If $\tau(p) = p$, the ratio is one.

Linear transformations. Linear transformations are bounded by propagating these rules through their actual sequence of operations.

Primitive bounds. The bounds above do not require independence of the errors. For probabilistic primitive bounds, the failure probability of their joint event is at most the sum of their failure probabilities. For rescaling by D_v , the centered residues satisfy $\|\rho_{v,i}\|_{\text{coef},\infty} \leq (D_v - 1)/2$ for $i \in \{0, 1\}$. Equation (2) therefore gives $|\sigma_j(\rho_{v,i})| \leq N(D_v - 1)/2$. If $\|s\|_{\sigma,\infty} \leq B_s$, then

$$\begin{aligned} |\sigma_j(\rho_{v,0} + \rho_{v,1}s)| &\leq |\sigma_j(\rho_{v,0})| + |\sigma_j(\rho_{v,1})| |\sigma_j(s)| \\ &\leq \frac{N(D_v - 1)}{2} + \frac{N(D_v - 1)}{2} B_s \\ &= \frac{N(D_v - 1)}{2} (1 + B_s). \end{aligned}$$

Thus $R_v = N(D_v - 1)(1 + B_s)/2$ is a valid bound, with $R_v/D_v = N(1 + B_s)(1 - 1/D_v)/2 \leq N(1 + B_s)/2$.

F Details of RNS Key Switching

We give the procedure and bounds for the RNS decomposition of Section 2.3, using exact conversion and centered residues, followed by the counts in Table 1. Procedure. Let $\text{ct} = (c_0, c_1)$ be a ciphertext under s' . For block k , let Q_k be the product of its primes and take $d_k := [c_1]_{Q_k} \in R$, for $0 \leq k < \beta$. Choose integer weights g_k satisfying $g_k \equiv 1 \pmod{Q_k}$ and $g_k \equiv 0 \pmod{Q/Q_k}$. The Chinese remainder theorem gives $c_1 \equiv \sum_{k=0}^{\beta-1} g_k d_k \pmod{Q}$, and hence $c_1 s' \equiv \sum_{k=0}^{\beta-1} d_k (g_k s') \pmod{Q}$. Evaluation key pair $(b_k, a_k) \in R_{QP}^2$ has error $e_k \in R$ and satisfies $b_k + a_k s \equiv P g_k s' + e_k \pmod{QP}$. Choose integer representatives

t_0, t_1 for $\sum_k d_k b_k, \sum_k d_k a_k$ modulo QP , respectively. Then $t_0 + t_1 s \equiv Pc_1 s' + \sum_k d_k e_k \pmod{QP}$. Put $\rho_i := [t_i]_P$ for $i \in \{0, 1\}$. Modulus descent divides by P with rounding and returns $\mathbf{ct}' = (c'_0, c'_1)$, where $c'_0 = [c_0 + (t_0 - \rho_0)/P]_Q$ and $c'_1 = [(t_1 - \rho_1)/P]_Q$. This division cancels the factor P in the key messages, preserving the message scale.

Error bounds. The output satisfies

$$c'_0 + c'_1 s \equiv c_0 + c_1 s' + \kappa \pmod{Q}, \quad \kappa = \frac{\sum_k d_k e_k - \rho_0 - \rho_1 s}{P}.$$

The numerator is divisible by P , since $\rho_0 + \rho_1 s \equiv t_0 + t_1 s \equiv \sum_k d_k e_k \pmod{P}$. Thus $\kappa \in R$. With $\xi := -(\rho_0 + \rho_1 s)/P \in K$, the triangle inequality gives

$$\|\kappa\|_{\sigma, \infty} \leq \frac{1}{P} \sum_{k=0}^{\beta-1} \|d_k\|_{\sigma, \infty} \|e_k\|_{\sigma, \infty} + \|\xi\|_{\sigma, \infty}. \quad (14)$$

Equation (2) gives $\|d_k\|_{\sigma, \infty} \leq N(Q_k - 1)/2$ and $\|\rho_i\|_{\sigma, \infty} \leq N(P - 1)/2$. Consequently,

$$\|\kappa\|_{\sigma, \infty} \leq \frac{N}{2P} \sum_{k=0}^{\beta-1} (Q_k - 1) \|e_k\|_{\sigma, \infty} + \frac{N(P - 1)}{2P} (1 + \|s\|_{\sigma, \infty}).$$

This bound holds for every $c_1 \in R_Q$, including components multiplied by a public ring element. Relinearization has the same error bound when switching the third component from s^2 to s . When Q , P , or the decomposition changes, we recompute the blocks and the bound. An approximate RNS conversion may use different lifts of the digits or residues [9]; its corresponding bounds replace the centered bounds above in (14).

Operation counts. We count operations with inputs, outputs, and keys in NTT form, reusing the input NTT values within each decomposition block [18]. Let L_{Q_k} be the number of primes in block k , so $\sum_k L_{Q_k} = L_Q$. Decomposition first uses L_Q inverse transforms. For each d_k , one RNS conversion extends its coefficients from Q_k to QP/Q_k , followed by $L_Q + L_P - L_{Q_k}$ forward transforms; the NTT values modulo Q_k are already available. The transform count is therefore $L_Q + \sum_k (L_Q + L_P - L_{Q_k}) = \beta(L_Q + L_P)$, with β conversions. Multiplication by the β evaluation key pairs uses $2\beta N(L_Q + L_P)$ modular products.

For modulus descent, each of the two components uses L_P inverse transforms, one conversion from P to Q , and L_Q forward transforms. Subtracting the converted residues and multiplying by $P^{-1} \bmod Q$ uses $2NL_Q$ modular products in total. Thus modulus descent uses $2(L_Q + L_P)$ transforms and two conversions. For rescaling by D in one step, write $Q = DQ'$ and let L_D be the number of removed primes, so $L_D + L_{Q'} = L_Q$. The same steps with bases D and Q' use $2L_Q$ transforms, two conversions, and $2NL_{Q'}$ modular products. Each conversion acts on all N coefficients; its cost depends on the source and destination bases.

G Proof of Equation (2)

Proof. Write $f = \sum_{h=0}^{N-1} f_h X^h$ with $f_h \in \mathbb{R}$. Since $|\sigma_j(X)| = 1$, the triangle inequality gives $\|f\|_{\sigma, \infty} \leq \sum_{h=0}^{N-1} |f_h| \leq N \|f\|_{\text{coef}, \infty}$.

For the reverse bound, let $\omega_t := e^{(2t+1)\pi i/N}$, $0 \leq t < N$, be the roots of $X^N + 1$. For $0 \leq h, k < N$, geometric summation gives $\frac{1}{N} \sum_{t=0}^{N-1} \omega_t^{h-k} = 1$ when $h = k$ and zero otherwise. Expanding $f(\omega_t)$ and combining conjugate terms, using $f_h \in \mathbb{R}$ and $N = 2r$, yields

$$\begin{aligned} f_k &= \frac{1}{N} \sum_{t=0}^{N-1} f(\omega_t) \omega_t^{-k} = \frac{1}{N} \sum_{j=0}^{r-1} \left(\sigma_j(f) \sigma_j(X)^{-k} + \overline{\sigma_j(f) \sigma_j(X)^{-k}} \right) \\ &= \frac{1}{r} \sum_{j=0}^{r-1} \text{Re}(\sigma_j(f) \sigma_j(X)^{-k}). \end{aligned}$$

Thus $|f_k| \leq \frac{1}{r} \sum_{j=0}^{r-1} |\sigma_j(f)| \leq \|f\|_{\sigma, \infty}$ for every k . Taking the maximum over k completes (2). The multiplication bound follows from $\sigma_j(fg) = \sigma_j(f) \sigma_j(g)$ by taking absolute values and the maximum over j . $\square$

H Proof of Proposition 3

Proof. The recovery condition in (10) holds if $\bar{w}(u) < C/2$, or equivalently $2^{-\theta} + 2^\theta < C$. Setting $y := 2^\theta \geq 1$ gives $y^2 - Cy + 1 < 0$. For $C > 2$, its roots are $(C \pm \sqrt{C^2 - 4})/2$, with the smaller root below 1; taking logarithms gives the stated bound. $\square$

I A Concrete Example of the Error Bounds

We compute the bounds for the profile (160, 32, 32, 32), apply the recovery condition, and compare the bounds with a recorded execution.

Parameters. With $N = 2^{15}$ and $r = 2^{14}$, group g contains the 2^{12} slots $j \equiv g \pmod{4}$. Inputs lie in $[3/4, 7/8]$; four successive squarings compute x_j^{16} , with the public correction by u before each relinearization. Put $Y = X^{4096}$ and use

$$u = -3645862971 + 834730735(Y - Y^7) \\ - 173731438(Y^2 - Y^6) + 3069345111(Y^3 - Y^5).$$

For $j \equiv g \pmod{4}$, write $w_g = |\sigma_j(u)|$. Certified enclosures give the displayed logarithms

$$(\log_2 w_g)_{g=0}^3 \approx (-96.435365, 30.604317, 32.857700, 32.973348).$$

Let $Q_0 = \prod_{i=0}^9 q_i$, with $q_i = 2^{\ell_i} - c_i$ and

$$(\ell_0, \dots, \ell_9) = (49, 48, 47, 46, 47, 46, 47, 46, 47, 46), \\ (c_0, \dots, c_4) = (720895, 393215, 1048575, 589823, 1638399), \\ (c_5, \dots, c_9) = (1769471, 1835007, 3407871, 2490367, 3735551).$$

The auxiliary primes are $(p_i)_{i=0}^3 = 2^{28} - (65535, 196607, 393215, 1376255)$, componentwise. Set $P = \prod_i p_i$ and $p_\star = p_0$. Encryption uses $p_\star$; relinearization uses P , grouping active q_i into consecutive blocks of at most four primes. At layer $t = 1, \dots, 4$, divide successively by $d_{t,1} = q_{11-2t}$ and $d_{t,2} = q_{10-2t}$, giving $D_t = d_{t,1}d_{t,2}$ and $Q_t = \prod_{i=0}^{9-2t} q_i$.

The Lattigo v6.2.0 execution [18] uses $\Delta_0 = 2^{93}$ and 1024-bit encoding, scale arithmetic, and decoding. The stored Δ_t is obtained by squaring Δ_{t-1} and dividing successively by $d_{t,1}, d_{t,2}$, rounding after each operation. Define $\chi_t = \Delta_{t-1}^2 / (D_t \Delta_t)$; here $|\chi_t - 1| < 2^{-1023}$. Rational interval propagation bounds the inverse-transform coefficient error by 2^{-788} , the subsequent floating error in scaling and rounding by 2^{-818} , and the final decoder error by 2^{-734} per slot, including weight approximation.

Auxiliary bounds. For this ring, use $\|f\|_{\sigma,\infty} \leq C_N \|f\|_{\text{coef},\infty}$ with $C_N = 20861 \geq \csc(\pi/(2N))$; the latter constant follows by maximizing the sum of absolute cosine projections of the N roots. Secret and encryption randomizer coefficients are uniform ternary. Gaussian error coefficients have sampler parameter 3.2, truncation parameter 19.2, and conservative absolute bound 20. We condition on $\|s\|_{\sigma,\infty} \leq B_s = 3473$ and $\|e_k\|_{\sigma,\infty} \leq B_e = 70202$ for evaluation-key errors. Hoeffding and union bounds give these constants as

$$B_s = \left\lceil \sqrt{2N(160 + 24)} \right\rceil, \quad B_e = \left\lceil 20\sqrt{2N(160 + 28)} \right\rceil,$$

covering 256 ternary and 4096 error polynomials with joint failure probability at most 2^{-159} . The archived security estimate for Q_0P exceeds the separate 128-bit security target. For binary64 CRT conversion, set

$$\delta_{\text{CRT}}(m) = \frac{m(m+3)2^{-53}}{1 - (m+3)2^{-53}}.$$

The bounds below assume round-to-nearest conversion and exact integer modular operations within their checked ranges. They cover all actual operation inputs, including those after public correction.

I.1 Explicit Bounds for the Example

Encoding error B_{enc} . Including coefficient rounding and the numerical errors above,

$$\|e_{\text{enc}}\|_{\text{coef}, \infty} \leq \frac{1}{2} + \Delta_0 2^{-788} + 2^{-818} < 1, \quad B_{\text{enc}} = C_N.$$

Encryption error B_{dec} . The sampled errors after division by p_* and the two modulus-down errors give, respectively,

$$B_{\text{dec}} = \frac{20C_N(C_N + 1 + B_s)}{p_*} + C_N(1 + B_s) \left(\frac{1}{2} + \delta_{\text{CRT}}(1) + \frac{1}{2p_*} \right).$$

Relinearization error K_t . Let $Q_{t,k}$ and $m_{t,k}$ be the prime product and prime count of block k in Q_{t-1} . The implementation's digit coefficient bound is

$$H_{t,k} = \begin{cases} (Q_{t,k} + 1)/2, & m_{t,k} = 1, \\ Q_{t,k}(1/2 + \delta_{\text{CRT}}(m_{t,k})) + 1/2, & m_{t,k} > 1. \end{cases}$$

Including centering and CRT errors, (14) gives

$$K_t = \frac{C_N B_e}{P} \sum_k H_{t,k} + C_N(1 + B_s) \left(\frac{1}{2} + \delta_{\text{CRT}}(4) + \frac{1}{2P} \right).$$

Rescaling residues R_t . Each prime division has phase error at most $C_N(1+B_s)/2$. The second division also divides the first error by $d_{t,2}$, so the aggregate numerator bound is

$$R_t = D_t \frac{C_N(1 + B_s)}{2} \left(1 + \frac{1}{d_{t,2}} \right).$$

Message amplitude $A_{t,j}$. Since $x_{t,j} = x_{0,j}^{2^t}$, take $A_{t,j} = (7/8)^{2^t}$ for $0 \leq t \leq 4$.

Initial residual bound $B_{0,j}$. The fresh encoding and encryption errors give

$$B_{0,j} = \frac{B_{\text{enc}} + B_{\text{dec}}}{\Delta_0}.$$

Accumulated residual bound $B_{t,j}$. Lemma 4, with the stored-scale correction, gives for $j \equiv g \pmod{4}$

$$B_{t,j} = |\chi_t|(2A_{t-1,j}B_{t-1,j} + w_g B_{t-1,j}^2) + \frac{K_t + R_t}{D_t \Delta_t} + \frac{|\chi_t - 1|A_{t-1,j}^2}{w_g}.$$

The last term accounts for scale rounding; exact scales give $\chi_t = 1$. Computations use weight enclosures and outward rounding. The implementation propagates $w_g B_{t,j}$; division by a certified lower bound on w_g recovers a valid residual bound.

Output residual bound $B_{\text{out},j}$. Restoration is exact, so $B_{\text{out},j} = B_{4,j}$. Numerical decoding adds at most 2^{-734} to the restored error bound.

Common residual bound B . Take $B = \max_{0 \leq j < r} B_{\text{out},j}$.

I.2 Applying the Recovery Condition

Recovery at the selected modulus. All ideal outputs are nonnegative real numbers, so the constant coefficient attains the inverse-embedding coefficient norm:

$$\|\sigma^{-1}(\mathbf{x}_{\text{out}})\|_{\text{coef},\infty} = r^{-1} \sum_j x_{0,j}^{16} \leq (7/8)^{16} \approx 2^{-3.082321}.$$

The propagated coefficient error term is approximately $2^{-33.275980}$; $B\bar{w}(u)$ has the same displayed logarithm. Groups 1, 2, 3 contribute 9.15%, 43.61%, and 47.25% of the error term, showing that low-precision outputs still enter recovery. For $Q_4 = q_0 q_1$, the endpoint-safe available range is

$$(Q_4 - 1)/(2\Delta_4) \approx 2^{2.999999}.$$

The message-plus-error bound uses only $2^{-6.082320}$ of this range. Thus (10) guarantees recovery for all allowed inputs on the stated event; the ideal-message term dominates.

Effect on modulus selection. Dropping q_1 after evaluation leaves the formal precision bounds unchanged, but reduces the available range to $2^{-45.000001}$. The ideal-message term for the recorded inputs is $2^{-4.463161}$, so this candidate fails the recovery check. At the fixed scale, the sufficient condition requires $\log_2 Q_{\text{out}} > 89.537$ approximately; the selected $\log_2 Q_4 \approx 97$ passes. This is a sufficient-condition check, not a global optimality claim. The selected parameters also certify 161.766, 34.727, 32.473, and 32.358 bits in the four groups, meeting every precision target.

I.3 Bounds and Observed Values

The experimental key is used to extract phase errors and rescaling residues; restored errors are measured against exact rational reference values. Embedding magnitudes are 1024-bit observations, while the recovered coefficient maximum is measured exactly. Weight approximation is included in the bounds and does not affect the displayed digits. In Table 2, o is the observed maximum, b its bound, and the gap is $\log_2(b/o)$. Write $\mathcal{I}_g = \{j : j \equiv g \pmod{4}\}$ and $B_{\text{out},g}$ for the common bound in group g . Checks use unrounded values.

Table 2. Observed quantities and explicit bounds for the four-squaring example. Unqualified maxima are over all r slots. The last row uses the input-uniform coefficient bound, not the sharper bound specific to the recorded inputs.

Quantity (corresponding bound)	$\log_2 o$	$\log_2 b$	$\log_2(b/o)$
$\ e_{\text{enc}}\ _{\sigma,\infty} (B_{\text{enc}})$	7.696	14.349	6.653
$\ e_{\text{dec}}\ _{\sigma,\infty} (B_{\text{dec}})$	15.539	25.111	9.572
$\ \kappa_1\ _{\sigma,\infty} (K_1)$	95.539	107.546	12.007
$\ \kappa_2\ _{\sigma,\infty} (K_2)$	95.491	107.546	12.055
$\ \kappa_3\ _{\sigma,\infty} (K_3)$	95.371	107.459	12.088
$\ \kappa_4\ _{\sigma,\infty} (K_4)$	95.268	107.459	12.191
$\ \rho_{1,0} + \rho_{1,1}s\ _{\sigma,\infty} (R_1)$	108.540	118.111	9.571
$\ \rho_{2,0} + \rho_{2,1}s\ _{\sigma,\infty} (R_2)$	108.429	118.111	9.682
$\ \rho_{3,0} + \rho_{3,1}s\ _{\sigma,\infty} (R_3)$	108.260	118.111	9.851
$\ \rho_{4,0} + \rho_{4,1}s\ _{\sigma,\infty} (R_4)$	108.702	118.111	9.409
$\max_j x_{4,j} (A_{4,j})$	-3.082	-3.082	0.000
$\max_j \eta_{0,j} (\max_j B_{0,j})$	-77.461	-67.888	9.572
$\max_j \eta_{1,j} (\max_j B_{1,j})$	-76.706	-66.429	10.277
$\max_j \eta_{2,j} (\max_j B_{2,j})$	-75.713	-65.507	10.206
$\max_j \eta_{3,j} (\max_j B_{3,j})$	-75.600	-65.059	10.541
$\max_{j \in \mathcal{I}_0} \eta_{4,j} (B_{\text{out},0})$	-75.980	-65.331	10.649
$\max_{j \in \mathcal{I}_1} \eta_{4,j} (B_{\text{out},1})$	-76.965	-65.331	11.634
$\max_{j \in \mathcal{I}_2} \eta_{4,j} (B_{\text{out},2})$	-76.578	-65.331	11.247
$\max_{j \in \mathcal{I}_3} \eta_{4,j} (B_{\text{out},3})$	-76.931	-65.331	11.600
$\max_j \eta_{4,j} (B)$	-75.980	-65.331	10.649
$\max_{j \in \mathcal{I}_0} e_{4,j} (w_0 B_{\text{out},0})$	-172.415	-161.766	10.649
$\max_{j \in \mathcal{I}_1} e_{4,j} (w_1 B_{\text{out},1})$	-46.361	-34.727	11.634
$\max_{j \in \mathcal{I}_2} e_{4,j} (w_2 B_{\text{out},2})$	-43.721	-32.473	11.247
$\max_{j \in \mathcal{I}_3} e_{4,j} (w_3 B_{\text{out},3})$	-43.958	-32.358	11.600
$\ M_4\ _{\text{coef},\infty} / \Delta_4$	-4.463	-3.082	1.381