Compressed FHE

Accelerating Encrypted Matrix Multiplication in CKKS with Precision-Balanced Low-Rank Factor Chains

Dimitrios Schoinianakis[1] and Maryam Sabzevari[2]

[1] Nokia Bell Labs, Athens, Greece
[2] Nokia Bell Labs, Espoo, Finland

Abstract

This work establishes cFHE (compressed FHE), a unified analytical and empirical framework that integrates low-rank matrix factorization techniques into the CKKS homomorphic encryption scheme. Theoretical bounds are derived for the accumulation of relative error across sequences of factorized matrices, leading to an explicit expression for the attainable computation depth as a function of target accuracy, norm amplification behavior, and per-layer approximation quality. Extensions to tree-based evaluation structures are also formulated, allowing depth to scale logarithmically with the number of factors.

The analytical results are linked to CKKS arithmetic through a precision-balancing model that connects low-rank approximation errors and ciphertext noise. This connection is at the core of cFHE; it enables the automatic selection of CKKS parameters (polynomial modulus degree, modulus chain, and scaling factor) for a desired accuracy, ensuring that low-rank tolerances and cryptographic precision are jointly optimized.

Experimental evaluations demonstrate that encrypted low-rank matrix multiplications achieve both significant runtime improvements and reduction of ciphertext sizes over direct or tree-based encrypted multiplications while maintaining the prescribed accuracy. cFHE is agnostic to other CKKS optimizations and can be combined with them for further gains.

Keywords: privacy, homomorphic encryption, low-rank factorization

1 Introduction

Fully homomorphic encryption (FHE) enables arithmetic operations to be performed directly on encrypted data, providing a powerful mechanism for privacy-preserving computations [Gen09]. Among existing schemes, the CKKS protocol, which supports approximate arithmetic on complex vectors and matrices, has emerged as a suitable candidate for real-world applications in machine learning, predictive maintenance, and encrypted statistics [CKKS17]. Despite its flexibility, the computational and memory costs associated with CKKS remain a major bottleneck for large-scale encrypted linear algebra, where repeated multiplications amplify ciphertext noise and rapidly consume the available precision budget.

To address this limitation, low-rank factorization (LRF) techniques can be employed as an algebraic compression mechanism [EY36]. This work establishes cFHE (compressed FHE), a unified theoretical and empirical framework for integrating low-rank factorizations into the CKKS pipeline. By expressing matrices as products of thinner, low-rank factors,

Emails: dimitrios.schoinianakis@nokia-bell-labs.com (Dimitrios Schoinianakis), maryam.sabzevari@nokia-bell-labs.com (Maryam Sabzevari)

Date of this document: 2026-04-30.

both the arithmetic complexity and the ciphertext noise accumulation can be reduced. The study comprises three main components.

The first component introduces an analytical model for the propagation of approximation errors in chains of low-rank factorizations. This model characterizes how the relative error accumulates as a function of the per-layer approximation quality, the amplification factor of intermediate products, and the overall computation depth. From this analysis, a depth bound is derived that specifies the maximum number of consecutive multiplications that can be performed while satisfying a desired accuracy constraint.

The second component connects this error model to the arithmetic properties of the CKKS scheme. A precision-balancing mechanism is formulated, relating the error introduced by low-rank approximations to the quantization and rescaling noise intrinsic to CKKS ciphertext operations. This connection provides a methodology for translating analytical error bounds into concrete cryptographic parameters, such that numerical accuracy and encryption parameters are co-optimized rather than tuned independently.

The third component focuses on experimental validation. Across the evaluated configurations, encrypted low-rank matrix multiplication yields up to 300× speedup relative to direct serial chains and about 100× relative to direct tree-based chains, while reducing ciphertext footprint and preserving the prescribed accuracy.

The rest of the paper is organized as follows. Section 3 summarizes the CKKS primitives and fixed-point arithmetic knobs employed in the proposed co-design framework. Sections 4–5 introduce the LRF model and present the proposed methodology to embed precision-balanced LRF arithmetic into the CKKS pipeline. Section 6 presents the theoretical depth bounds together with empirical validation and Section 7 discusses performance results. Conclusions are offered in Section 8.

2 State-of-the-art

During the last five years several algorithmic and hardware improvements to the CKKS FHE scheme have been proposed. Bootstrapping operations, the primary computational bottleneck, achieved dramatic performance gains ranging from 15.39× through hybrid CKKS-TFHE scheme-switching [ACJ24] to 257× with GPU memory-centric optimizations [JKA+21], with recent functional bootstrapping methods reaching sub-millisecond amortized times [AKP25].

Hardware accelerators also demonstrated considerable improvements across platforms: FPGA implementations achieved considerable speedups for key-switching operations and up to 7300× for NTT-accelerated homomorphic evaluation [LLSL24, YYZ+23]. GPU implementations with optimized kernels showed up to 74× speedup over CPU baselines [ADVLG+25]. Unified multi-modal architectures like Trinity demonstrated that supporting multiple FHE schemes within a single accelerator could outperform specialized designs by 1.49-4.23× while requiring only 85% of their combined circuit area [DFH+24].

Algorithmic innovations proved equally significant, with scalar encoding optimizations achieving 3-88× speedup [BCCW19], error variance minimization reducing ciphertext sizes by 25% [LLK+20], and memory-aware design techniques improving arithmetic intensity by 3× [ADCJ+23]. In the following, we review the most relevant works in hardware acceleration and algorithmic optimizations for CKKS in more detail. Note that the proposed cFHE framework is orthogonal and complementary to these works, as it can be integrated with any of the aforementioned optimizations to further enhance CKKS performance.

2.1 Hardware acceleration

2.1.1 Bootstrapping Operations

Bootstrapping, the operation of homomorphically decrypting and re-encrypting a ciphertext without revealing the plaintext, remains one of the most computationally intensive operation in CKKS and, as such, it has received extensive research attention. In this direction, hybrid scheme approaches proved particularly effective. HEAP's scheme-switching between CKKS and TFHE enabled parallelized bootstrapping by converting a single RLWE ciphertext into multiple LWE ciphertexts [ACJ24], achieving 15.39× improvement compared to other state-of-the-art accelerators of that time like FAB [AdCY+22] while requiring 18× less memory for bootstrapping keys. For neural network applications, this translated to 14.71× and 11.57× improvements in logistic regression training compared to FAB and subsequent FAB-2 implementations.

2.1.2 Key-Switching and Relinearization

Key-switching operations, the primary performance bottleneck after bootstrapping, have also received several optimizations through both algorithmic and architectural improvements. For example, a configurable arithmetic core supporting NTT, INTT, and multiply-accumulate operations achieved 11.33× speedup compared to classical architectures and 1.07× versus state-of-the-art designs [LLSL24]. When compared to CPU implementations, FPGA-based key-switching accelerators demonstrated 1600-2700× speedup over OpenFHE.

For large-scale deployments, TurboHE's FPGA cluster architecture with 9 Xilinx VU13P FPGAs achieved 1096× throughput improvement compared to single-threaded CPU implementation [LED+23]. The scalable architecture supported CKKS parameters with 32,768 polynomial coefficients and 594-bit coefficient bitwidth decomposed into 11 Residue Number System (RNS) moduli.

2.1.3 Number Theoretic Transform (NTT) Optimizations

NTT operations, fundamental to polynomial multiplication in CKKS, are typically the main bottleneck when we consider the problem of CKKS optimization from the core arithmetic level point of view. Efficient data access pattern optimizations have been proposed that achieve 46-80% hardware resource reduction while improving FPGA performance by 4.9×, with homomorphic evaluation showing 7300× improvement over software implementation [YYZ+23]. Parallel NTT multiplier architectures achieved throughput of 32.7 Gbps with at least 1.5× improvement over recent works [NKL23] .

Novel architectural approaches included configurable NTT units supporting all polynomial degrees in SEAL-Embedded library [DMGS23], achieving approximately 1000× speedup over pure software implementation. For resource-constrained platforms, CASA's partially reduction-free modular arithmetic achieved 10.8× speedup over CPU implementations [HOMK+24] with 138× and 105× improvements in power-delay product versus GPU implementations. Lately, novel architectures that combine digit-serial modulo arithmetic with appropriate redundant data representation have also been proposed [ASD25].

2.1.4 Unified and Multi-Modal Architectures

Finally, several works have developed unified architectures supporting multiple FHE schemes or operation types, addressing system complexity and support of different arithmetic operations. Trinity, the first multi-modal FHE accelerator, efficiently supported CKKS, TFHE, and their conversion within a unified architecture [DFH+24]. Through adaptive component allocation for NTT and MAC operations, Trinity maintained high utilization across various polynomial lengths and imbalanced workloads, achieving 48%

utilization for CKKS and 64% for TFHE workloads. Performance improvements reached 1.49× over SHARP for CKKS and 4.23× over Morphling for TFHE [PPK24], with 919.3× improvement for FHE-conversion schemes over CPU implementations. Notably, the hardware overhead was only 85% of the summed circuit areas of SHARP and Morphling.

2.2 Algorithmic Optimizations

Beyond hardware acceleration, pure algorithmic improvements significantly enhanced CKKS efficiency through novel mathematical approaches and optimization techniques. Scalar encoding optimizations in nGraph-HE2 achieved 3-88× runtime speedup, while the work in [BCCW19] introduced optimizations to ciphertext-plaintext operations which yielded 2.6-4.2× runtime speedup .

Other efforts focused on reducing approximation errors and depth consumption. The lazy baby-step giant-step (BSGS) algorithm reduced computational complexity by half by utilizing lazy relinearization and rescaling techniques [LLK+22]. Error variance minimization approaches enabled reduction of ciphertext polynomial degree from 2^{16} to 2^{15}, decreasing key and ciphertext sizes by at least 25% [LLK+20]

The reduced-error CKKS RNS variant removed approximation errors due to LWE noise in encryption and key switching operations [KPP20]. Through level-specific scaling factors and automated rescaling operations, the implementation achieved 6-9 bits smaller approximation error for uniform ternary secret keys and up to 12 bits smaller error for sparse secret settings, though with 1.2-1.9× runtime slowdown.

3 CKKS background and preliminaries

CKKS enables approximate arithmetic over packed plaintext slots by encoding vectors into ring elements and evaluating arithmetic directly on ciphertexts. We only recall the knobs needed by our co-design: the fixed-point *scale* Δ, the leveled modulus chain, and the resulting depth-driven precision budgeting.

CKKS operates over the cyclotomic ring

$$\mathcal{R} = \mathbb{Z}[X]/(X^N + 1), \qquad \mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1),$$

where N is a power of two and q is a coefficient modulus. A packed plaintext is represented as an element of $\mathcal{R}$, while ciphertexts live in $\mathcal{R}_q^2$ under the standard RLWE form [CKKS17]. CKKS represents fixed-point data by associating a *scale* $\Delta > 0$ to each ciphertext: conceptually, the encoded plaintext carries values multiplied by Δ, so the discretization error after decoding is on the order of $1/\Delta$. Increasing Δ improves fractional precision but reduces the safe dynamic range due to the no-wrap condition $|\Delta x| \ll q/2$ for all intermediate slot magnitudes x. This precision–range tension is central to our balancer.

CKKS is a *leveled* approximate HE scheme: evaluation proceeds under a descending sequence of moduli. Concretely, the coefficient modulus is a product of pairwise coprime RNS primes,

$$Q = \prod_{i=0}^{L} q_i, \qquad \text{so that} \qquad \mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1).$$

At level ℓ, ciphertexts live in the smaller ring

$$Q_\ell := \prod_{i=0}^{\ell} q_i, \qquad \text{hence} \qquad \mathcal{R}_{Q_\ell} \subset \mathcal{R}_Q.$$

Homomorphic addition leaves the level unchanged. Homomorphic multiplication increases noise and also squares the scale. CKKS therefore applies *rescaling* to divide by the current

prime q_ℓ and move from Q_ℓ down to $Q_{\ell-1} = Q_\ell/q_\ell$:

$$\mathsf{Res}(\mathbf{c}) \;=\; \left\lfloor \frac{1}{q_\ell}\,\mathbf{c} \right\rceil \quad \text{in } \mathcal{R}^2_{Q_{\ell-1}}.$$

Each multiplication followed by a rescale consumes one level, so the total allowable ciphertext depth is L. Under standard parameter choices, the resulting precision loss is essentially governed by this multiplicative depth and stays within about one bit of the loss in unencrypted approximate arithmetic [CKKS18].

3.1 Precision budgeting (balancer)

CKKS tracks numerical stability via the relative error notion in [CKKS17]. In practice, and in the regime we target, the effective loss of reliable fractional bits is well approximated as roughly linear in the number of ciphertext–ciphertext multiplications. We therefore adopt a conservative rule that each multiplication consumes a fixed $\mu \in \{2, 3\}$ fractional bits.

Setting up CKKS involves selecting the initial scale Δ, the polynomial degree N, and a modulus chain $q_i, i \in [0, L]$. We view this as budgeting over two resources: fractional precision (scale bits) and integer headroom (to avoid wrap-around). Let $p_0 = \log_2 \Delta$ denote the initial scale bits. After L multiplications, we retain at least $p_0 - L\mu$ usable fractional bits. To guarantee a final precision of p_{target} bits, we choose

$$p_0 \;=\; p_{\text{target}} + L\,\mu + p_s, \qquad \Delta = 2^{p_0}, \tag{1}$$

where p_s is a depth-independent slack (typically a few bits) that absorbs rounding and rescaling headroom.

On the modulus side, we assemble a chain with two outer primes of ψ bits and $L-1$ inner primes of b bits:

$$[q_L, \ldots, q_0] = [\psi, \underbrace{b, \ldots, b}_{L-1}, \psi], \qquad B_{\text{total}} \approx 2\psi + (L-1)\,b.$$

Rescaling divides by the current modulus q_ℓ and renormalizes by Δ, so numerical stability requires $\log_2 q_\ell \approx \log_2 \Delta \approx p_0$ at each active level. In practice, we tie the inner primes to the scale bits:

$$b \;\approx\; p_0. \tag{2}$$

It is convenient to interpret p_0 as a fixed-point budget split into integer and fractional bits, $p_0 = p_{\text{int}} + p_{\text{frac}}$. Assume plaintext slots satisfy $|m| \le 2^{p_{\text{int}}}$ and we encode at $\Delta = 2^{p_0}$. Avoiding wrap-around at the top level requires an outer-prime headroom of approximately p_{int} bits (plus a small safety margin r), which we encode by setting

$$\psi \;=\; b + p_{\text{int}} + r, \tag{3}$$

so that $\psi - b \ge p_{\text{int}} + r$. We finally select the smallest polynomial modulus degree N whose library security table supports B_{total} bits at the desired security level [SEA23].

Equations (1), (2) and (3) together define the CKKS precision–range balancer used throughout this paper. In the next section we extend this balancer by splitting the total error tolerance between CKKS noise and low-rank factorization.

4 Low-rank factorization

LRF acts as an algebraic compression mechanism for matrix operators. Each matrix $\mathbf{A}_i \in \mathbb{R}^{m_i \times n_i}$ is approximated by a rank-r product

$$\mathbf{A}_i \;\approx\; \widehat{\mathbf{A}}_i := \mathbf{W}_i\mathbf{H}_i, \qquad \mathbf{W}_i \in \mathbb{R}^{m_i \times r},\ \mathbf{H}_i \in \mathbb{R}^{r \times n_i},$$

for $i = 1, \ldots, L$, where L is the length of the matrix chain and $r \ll \min(m_i, n_i)$. To simplify notation, we assume $m_i = n_i = n$ in the remainder. The per-layer residual is

$$\mathbf{E}_i := \mathbf{A}_i - \widehat{\mathbf{A}}_i.$$

Its relative magnitude is summarized by the *tail ratio* τ_i. In spectral norm,

$$\tau_i \equiv \tau_i^{(2)} \;=\; \frac{\|\mathbf{E}_i\|_2}{\|\mathbf{A}_i\|_2}. \tag{4}$$

and analogously $\tau_i^{(F)} = \|\mathbf{E}_i\|_F / \|\mathbf{A}_i\|_F$ in Frobenius norm. In practice we compute $(\mathbf{W}_i, \mathbf{H}_i)$ by truncated or randomized Singular Value Decomposition (SVD) and measure τ_i empirically.

4.1 Error propagation and amplification

Let $\mathbf{P}_L = \mathbf{A}_1 \cdots \mathbf{A}_L$ be the exact product, $\widehat{\mathbf{P}}_L = \widehat{\mathbf{A}}_1 \cdots \widehat{\mathbf{A}}_L$ its LRF approximation and define $\mathcal{E}_L := \mathbf{P}_L - \widehat{\mathbf{P}}_L$. Using $\mathbf{A}_L = \widehat{\mathbf{A}}_L + \mathbf{E}_L$ and adding/subtracting $\widehat{\mathbf{P}}_{L-1}\mathbf{A}_L$ yields the identity

$$\mathcal{E}_L = \mathcal{E}_{L-1}\mathbf{A}_L + \widehat{\mathbf{P}}_{L-1}\mathbf{E}_L, \qquad \mathcal{E}_0 = 0.$$

Taking spectral norms and defining the per-layer *amplification factor* $\alpha_i := \|\mathbf{A}_i\|_2$ gives the recurrence

$$\|\mathcal{E}_L\|_2 \;\le\; \alpha_L \, \|\mathcal{E}_{L-1}\|_2 \;+\; \|\widehat{\mathbf{P}}_{L-1}\|_2 \, \|\mathbf{E}_L\|_2. \tag{5}$$

With $\|\mathbf{E}_j\|_2 \le \tau_j \alpha_j$ and $\|\widehat{\mathbf{P}}_j\|_2 \le \prod_{k=1}^{j} \alpha_k$, this unrolls to

$$\|\mathcal{E}_L\|_2 \;\lesssim\; \sum_{j=1}^{L} \tau_j \, \alpha_j \prod_{k=j+1}^{L} \alpha_k. \tag{6}$$

Homogeneous regime. When $\alpha_i \approx \alpha$ and $\tau_i \approx \tau \ll 1$, the resulting relative error admits the unified estimate

$$\varepsilon_{\mathrm{LRF}} \;:=\; \frac{\|\mathbf{P}_L - \widehat{\mathbf{P}}_L\|_2}{\|\mathbf{P}_L\|_2} \;\approx\; \begin{cases} \tau \, \dfrac{\alpha^L - 1}{\alpha - 1}, & \alpha > 1, \\ L\,\tau, & \alpha \approx 1. \end{cases} \tag{7}$$

Thus, $\alpha > 1$ yields geometric amplification of residuals, whereas normalizing layers to keep $\alpha \approx 1$ yields near-linear accumulation.

4.2 Serial vs. tree evaluation

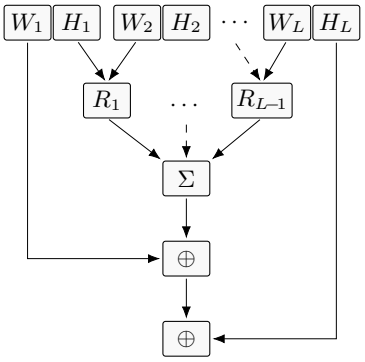
In practice, factor chains are commonly evaluated by a binary contraction tree, which reduces sequential depth and is therefore attractive for leveled CKKS. For the LRF chain

$$\mathbf{P}_L = (\mathbf{W}_1\mathbf{H}_1)(\mathbf{W}_2\mathbf{H}_2)\cdots(\mathbf{W}_L\mathbf{H}_L),$$

a standard strategy first forms the $r \times r$ *bridge* products $\mathbf{H}_i\mathbf{W}_{i+1}$ in parallel and then reduces these bridges in a binary tree, followed by boundary contractions with $\mathbf{W}_1$ and $\mathbf{H}_L$ (Figure 1). This changes the effective sequential depth from $L - 1$ (serial) to $\mathcal{O}(\log L)$ (tree), which directly lowers CKKS level requirements.

4.3 Effective amplification and stable depth

We now summarize the longest stable chain lengths under (i) serial left-to-right multiplication and (ii) binary-tree contraction. We use a target relative tolerance $\delta > 0$ and the homogeneous regime $\alpha_i \approx \alpha$, $\tau_i \approx \tau$.



S0: Low-rank factors
L pairs (W_i, H_i), $\mathcal{O}(Lnr)$

S1: Bridge contractions
$(L-1)$ mults, $\mathcal{O}(Lnr^2)$

S2: Reduction tree
depth $= \lceil \log_2(L-1) \rceil$, $\mathcal{O}(Lr^3)$

S3: Boundary product
1 mult, $\mathcal{O}(nr^2)$

S4: Boundary product
1 mult, $\mathcal{O}(n^2 r)$

Figure 1: Parallel LRF contraction: bridge contractions (S1), tree reduction (S2), and boundary products (S3–S4).

4.3.1 Serial depth bound

Imposing $\varepsilon_{\mathrm{LRF}} \le \delta$ on the homogeneous estimate in (7) yields the sufficient condition

$$L \le L_{\max}^{\mathrm{serial}}(\alpha, \tau, \delta) := \begin{cases} \dfrac{\log\!\left(1 + \frac{\delta(\alpha-1)}{\tau}\right)}{\log(\alpha)}, & \alpha > 1, \\ \dfrac{\delta}{\tau}, & \alpha \le 1. \end{cases} \tag{8}$$

4.3.2 Tree contraction and effective depth

In a binary-tree contraction, errors propagate along paths of length $\mathcal{O}(\log L)$ through $r \times r$ bridge products. In the normalized regime $\alpha \approx 1$, a standard sub-multiplicativity argument yields the logarithmic growth proxy

$$\|\mathbf{P}_L - \mathbf{P}_L^{(\mathrm{tree})}\|_2 \;\lesssim\; c_r\, \tau\, \log L, \qquad c_r = \mathcal{O}(r), \tag{9}$$

so the tree structure reduces error propagation depth from $\Theta(L)$ to $\Theta(\log L)$, thereby limiting the multiplicative amplification of per-layer residuals relative to serial evaluation..

Figure 1 highlights that the parallel bridge contractions (S1) contribute +1 level, reduction (S2) contributes logarithmic sequential depth, while boundary contractions (S3–S4) add a constant +2 levels, motivating the required leveled depth to be

$$D_{\mathrm{eff}}(L) := \lceil \log_2(L-1) \rceil + 3, \tag{10}$$

which converts reduction-tree depth into CKKS level consumption.

To express a tree budget in terms of equivalent original factors, we convert the serial budget into a tree-level budget and then back into a factor count:

$$\begin{aligned} L_{\max}^{\mathrm{levels}}(\alpha, \tau, \delta) &:= \frac{L_{\max}^{\mathrm{serial}}(\alpha, \tau, \delta)}{\log_2(\alpha)}, \\ L_{\max}^{\mathrm{tree}}(\alpha, \tau, \delta) &:= 2^{\,L_{\max}^{\mathrm{levels}}(\alpha, \tau, \delta)} - 1. \end{aligned} \tag{11}$$

Remark. When $\alpha > 1$, the tree formulation can admit substantially larger chains due to its logarithmic-depth structure; when $\alpha \approx 1$, both models recover a near-linear $L_{\max} \approx \delta/\tau$ regime. In encrypted pipelines, this reduction in effective depth is critical because $D_{\mathrm{eff}}(L)$ determines CKKS level consumption.

5 LRF–CKKS Co-Design and Error Bounds

Recall from Section 3 that the working precision at the start of a CKKS computation is

$$p_0 \;=\; p_{\text{target}} + L\,\mu + p_s,$$

where L denotes the effective multiplicative depth (serial: $L = L_{\max}^{\text{serial}}$, tree: $L = L_{\max}^{\text{tree}}$ per (11)), $\mu \in \{2, 3\}$ is the per-multiplication fractional-bit loss, and p_s (typically 5–10 bits) is a safety margin for rescaling and rounding. The plaintext scale is set as $\Delta = 2^{p_0}$, and the coefficient modulus chain is constructed to sustain this budget across L multiplications.

5.1 Estimator-Oriented Precision Mapping

We embed the LRF bounds directly into the CKKS balancer so that an estimator can construct a CKKS context from measured quantities (α, τ) and a user-specified tolerance δ.

5.1.1 Error allocation

Let $\varepsilon_{\text{LRF}}(L)$ denote the LRF truncation error at depth L as bounded in Section 4 (e.g., serial case $\varepsilon_{\text{LRF}}(L) \approx \tau(\alpha^L - 1)/(\alpha - 1)$, with the tree case obtained by replacing L by D_{eff}). Assuming independence between approximation and quantization noise, the relative error satisfies

$$\varepsilon_{\text{total}}^2 \;\approx\; \varepsilon_{\text{LRF}}^2 + \varepsilon_{\text{CKKS}}^2 \;\le\; \delta^2. \tag{12}$$

Once ε_{LRF} is fixed (via rank and (α, τ)), the remaining CKKS budget is

$$\varepsilon_{\text{CKKS}} \;=\; \sqrt{\delta^2 - \varepsilon_{\text{LRF}}^2}\,,\; p_{\text{target}} \;=\; \Big\lceil -\log_2\big(\varepsilon_{\text{CKKS}}\big) \Big\rceil, \tag{13}$$

where p_{target} is the number of fractional bits that must survive after all multiplications.

5.1.2 Fractional precision

The initial fractional precision p_{frac} must be large enough (i) to leave p_{target} bits after L rescalings and (ii) to keep CKKS noise at or below the LRF truncation error implied by (α, τ) and the matrix dimension n:

$$\begin{aligned} p_{\text{frac}} = \max\Big(&\underbrace{\lceil -\log_2 \varepsilon_{\text{CKKS}} \rceil + L\mu}_{\text{survivor floor}}, \\ &\underbrace{\lceil -\log_2 \tau + \log_2(\beta\,\kappa\,L\,\sqrt{n}) \rceil + s_{\text{dom}}}_{\text{dominance floor}}\Big). \end{aligned} \tag{14}$$

The first term ensures that L multiplications with loss μ bits each leave at least p_{target} fractional bits. The second term enforces the LRF–CKKS dominance boundary: β and κ absorb implementation-dependent factors (e.g., normalization and rescale tightness), $L\sqrt{n}$ captures depth and dimensionality, and s_{dom} (2–4 bits) accounts for model slack. In practice we also enforce $p_{\text{frac}} \ge \lceil -\log_2 \delta \rceil$ so that the requested digits remain achievable when ε_{LRF} is small.

5.1.3 Integer headroom

The integer budget p_{int} must accommodate amplification along the chain and bounds worst-case slot magnitudes implied by (α, τ):

$$p_{\text{int}} \;\ge\; \max\big(10,\; L\max(0, \log_2 \alpha)\big). \tag{15}$$

Here p_{int} provides headroom under the assumption that inputs are pre-scaled to unit range, with packing/layout constants absorbed into the fixed 10-bit floor; in our experiments this bound is conservative for all measured intermediate magnitudes.

5.1.4 Initial precision and modulus chain

With p_{frac} and p_{int} fixed, the estimator sets:

$$\begin{aligned} p_0 &= p_{\text{frac}} + p_s, \\ \Delta &= 2^{p_0}, \\ b &\equiv p_0, \\ \psi &= b + p_{\text{int}} + \gamma, \\ [q_L, \ldots, q_0] &= [\psi, \underbrace{b, \ldots, b}_{L-1}, \psi], \end{aligned} \tag{16}$$

with $p_s \in [5, 10]$ the safety bits for rescaling and γ a small range margin. The total modulus budget is $B_{\text{total}} \approx 2\psi + (L-1)b$. The smallest polynomial modulus degree $N = 2^k$ satisfying the library security constraint at bit-length B_{total} is then selected [SEA23]. Overall, (14)–(16) implement the direct mapping

$$(\delta, \alpha, \tau, n, L) \longmapsto \big(p_{\text{frac}}, p_{\text{int}}, p_0, \Delta, [q_L, \ldots, q_0], N\big), \tag{17}$$

which defines the proposed **co-optimized LRF-CKKS balancer**, thereby embedding LRF error bounds into CKKS parameter selection.

5.2 Joint LRF–CKKS Noise Dynamics

Recall that in CKKS, ciphertexts encode scaled fixed-point values with a constant scale $\Delta = 2^{p_0} = 2^{p_{\text{frac}}+p_s}$; After L computation levels, the accumulated CKKS error is well approximated by

$$\varepsilon_{\text{CKKS}}(L) \approx \frac{L}{2^{p_{\text{frac}}}}. \tag{18}$$

To compare the two error sources from CKKS and LRF we use

$$\rho(L) = \frac{\varepsilon_{\text{LRF}}(L)}{\varepsilon_{\text{CKKS}}(L)}.$$

This partitions the design space into three regimes:

- **LRF-dominated** ($\rho(L) \gg 1$): typically when $\alpha > 1$ or τ is large; improving CKKS precision has limited effect and better factorization (higher rank, normalization) is required.
- **CKKS-dominated** ($\rho(L) \ll 1$): for $\alpha \approx 1$ and small τ; accuracy is limited by ciphertext quantization, and increasing p_{frac} improves accuracy.
- **Balanced** ($\rho(L) \approx 1$): both contributions are comparable, yielding near-optimal utilization of CKKS precision.

The boundary is captured by the approximate equilibrium

$$\tau \cdot \frac{\alpha^L - 1}{\alpha - 1} \approx \frac{1}{2^{p_{\text{frac}}}}, \tag{19}$$

which equates the LRF bound with the CKKS noise floor (18). The second term in (14) is precisely a lower bound on p_{frac} obtained by enforcing $\varepsilon_{\text{CKKS}} \lesssim \varepsilon_{\text{LRF}}$ at the chosen depth.

Figure 2 visualizes these regimes in the (α, τ) plane for fixed L and p_{frac}. Pre-scaling the factors so that $\alpha \to 1$ moves operating points from LRF-dominated to balanced regions, enabling deeper circuits at the same δ.

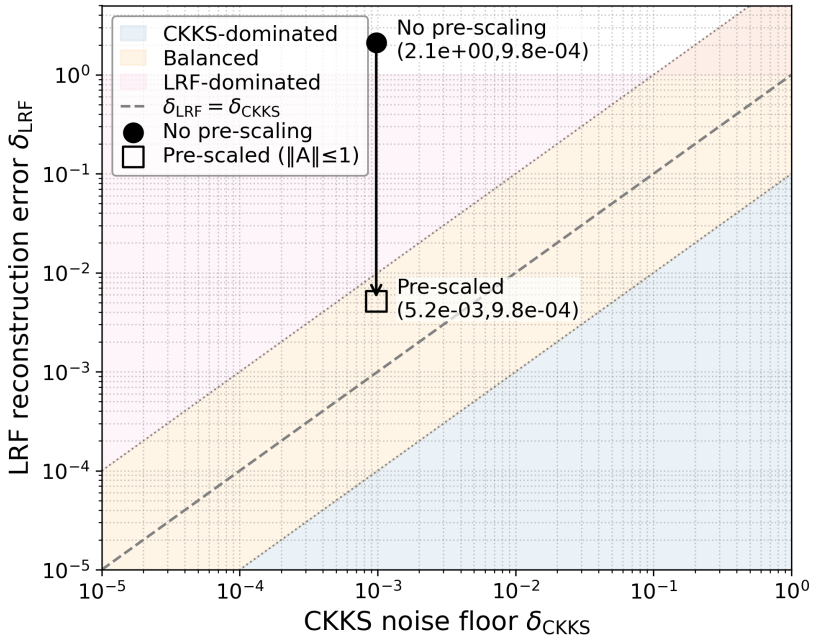


Figure 2: Noise balance map of LRF vs. CKKS regimes in (α, τ) plane (fixed L, p_{frac}).

5.3 The Co-Optimized LRF–CKKS Balancer

We now translate the balancer of (17) into a step-by-step procedure for constructing CKKS parameters from measured (α, τ) and a target tolerance δ.

i. **LRF calibration:** Choose rank r and measure (α, τ). Use the LRF model of Section 4 to compute $\varepsilon_{\mathrm{LRF}}(L)$ for the chosen depth (serial) or effective depth (tree).

ii. **Stable depth:** For tolerance δ, determine an admissible depth by enforcing $\varepsilon_{\mathrm{LRF}}(L) \le \delta$, and optionally also $\varepsilon_{\mathrm{CKKS}}(L) \le \delta$ using (18).

iii. **CKKS allocation:** Fix L and evaluate $\varepsilon_{\mathrm{LRF}}(L)$. The remaining CKKS budget $(\varepsilon_{\mathrm{CKKS}}, p_{\mathrm{target}})$ follows from (12) and (13). A balanced operating point targets $\varepsilon_{\mathrm{LRF}} \approx \varepsilon_{\mathrm{CKKS}}$.

iv. **Precision partitioning:** Compute $(p_{\mathrm{frac}}, p_{\mathrm{int}})$ using (14)–(15).

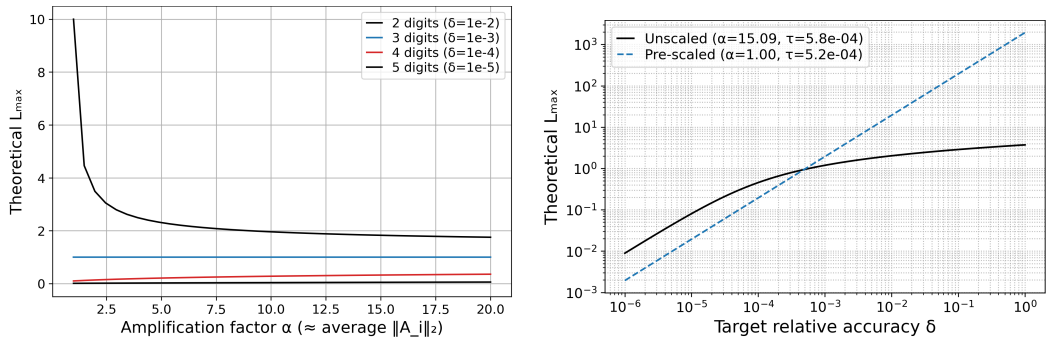
v. **CKKS parameters:** Set $p_0 = p_{\mathrm{frac}} + p_s$ and $\Delta = 2^{p_0}$, build the modulus chain via (16), and select the smallest N satisfying the security constraint for B_{total}.

Figure 3 illustrates how the LRF bounds constrain depth and motivate pre-scaling. When $\alpha \approx 1$, loose targets can tolerate chain lengths on the order of $L_{\max} \sim 10$, while $L_{\max}$ collapses rapidly as α grows. Using measured parameters, pre-scaling that enforces $\alpha \approx 1$ moves the system into the linear regime $\varepsilon_{\mathrm{LRF}}(L) \approx L\tau$, increasing admissible depth for the same δ.

Overall, the LRF–CKKS balancer provides a systematic co-design methodology that controls approximation and encryption errors while minimizing cryptographic resources.

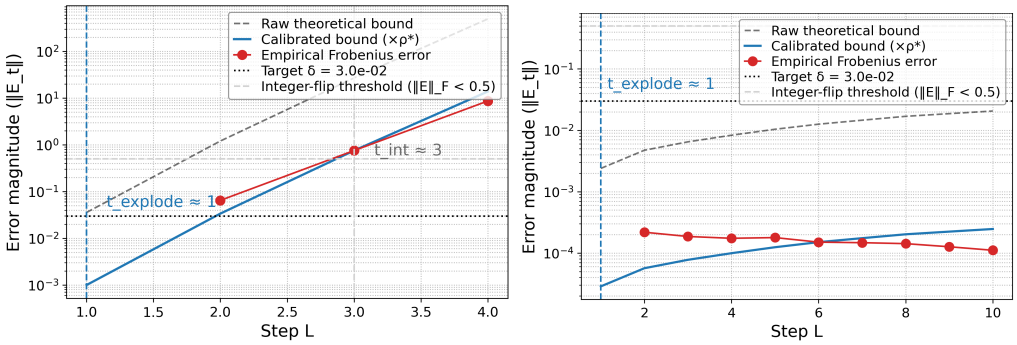
6 Theoretical Analysis and Empirical Results

The worst-case bounds in Section 4 are intentionally conservative (spectral-norm upper bounds compounded across depth). We introduce a simple *calibration factor* that preserves the bound's trajectory (geometric vs. linear) while matching observed magnitudes. We



(a) $L_{\max}$ vs. α **(b)** $L_{\max}$ vs. δ

Figure 3: Stable-depth diagnostics (theory and measured (α, τ)).



(a) No pre-scaling **(b)** With pre-scaling

Figure 4: Calibrated bound: geometric growth (left) vs. near-linear (right).

empirically validate the analytical balancer by comparing its depth predictions against measured error growth and stable-depth observations across accuracy budgets.

6.1 Calibration of Error Bounds

The expressions in (6) and (7) correctly capture the *shape* of error growth but are typically pessimistic in magnitude. To obtain a practical predictor without abandoning theoretical safety, we introduce a calibration (tightness) factor computed from measured chains.

For each step $t = 1, 2, \ldots, L$, we define a normalized tightness ratio

$$\rho_t \;=\; \frac{\|\mathcal{E}_t\|_F}{\sqrt{n}\,\big(\tau\,(\alpha^t - 1)/(\alpha - 1)\big)}, \qquad \rho^* \;=\; \mathrm{median}_t\,\rho_t,$$

where we use Frobenius norm for empirical cumulative error and normalize by $\sqrt{n}$ to obtain a per-element measure. The calibrated predictor is then

$$\|\mathcal{E}_t\|_{\mathrm{cal}} \;=\; \rho^* \cdot \tau\,\frac{\alpha^t - 1}{\alpha - 1}.$$

This scaling preserves the theoretical trajectory (geometric or linear) while aligning magnitudes to observed errors. We use two conservative thresholds to flag loss of numerical stability:

$$\frac{\|\mathcal{E}_t\|_{\mathrm{cal}}}{\|\mathbf{P}_t\|_F} \geq \delta, \quad \text{or} \quad \|\mathcal{E}_t\|_{\mathrm{cal}} \geq 0.5,$$

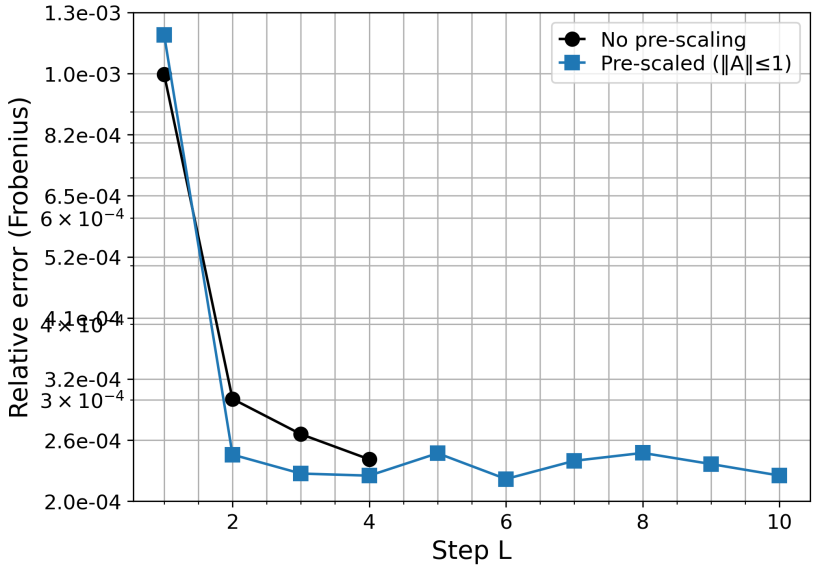


Figure 5: Empirical LRF error growth vs. depth L (no pre-scale vs. pre-scaled).

where δ is the target relative error. The first condition indicates loss beyond tolerance, while the second is a practical proxy for integer-part instability. With pre-scaling, we typically observe $\rho^* \in [10^{-3}, 10^{-1}]$, confirming that the raw bound is conservative in magnitude but useful for predicting growth regimes.

Figure 4 illustrates calibrated trajectories under geometric and near-linear regimes. Without pre-scaling (left), the raw bound crosses δ almost immediately, while the calibrated bound tracks empirical errors closely and predicts the onset of instability after only a few multiplications. With pre-scaling (right), the calibrated curve remains far below both δ and the integer threshold over the tested depth, indicating that LRF error is negligible for CKKS-relevant depths once $\alpha \approx 1$ is enforced.

Figure 5 reports the empirical relative error of (7) (in Frobenius norm) as a function of chain length L (target $\delta \approx 3 \times 10^{-2}$). In both configurations, measured relative error lies in the 10^{-4}–10^{-3} range, well below the target. Without pre-scaling, the generator halts around $L = 4$ due to amplified tail energy and subsequent loss of integer recovery; with pre-scaling, error remains essentially flat and chains reach the test value $L = 10$. These results support the linear regime predicted by $L_{\max} \approx \delta/\tau$ when $\alpha \approx 1$.

Finally, we cross-check the analytical depth predictions from (8) and (11) against empirical stable depths across accuracy budgets. Figure 6 shows that without pre-scaling, both analysis and measurements agree on small stable depth ($L_{\max} \leq 4$) across the explored δ range, reflecting geometric amplification when $\alpha > 1$. With pre-scaling, predicted $L_{\max}$ grows approximately linearly with δ, and empirical points track the analytical curve up to the experimental cap around $L \approx 10$ (the plateau reflects the configured test limit rather than a true depth limit).

7 Performance Analysis and Discussion

This section evaluates runtime, ciphertext footprint, and precision behavior of encrypted LRF-enabled matrix chains against two baselines: (i) direct encrypted multiplication in a serial chain and (ii) direct encrypted multiplication using a tree structure. For LRF we use the tree-based contraction with effective depth D_{eff}.

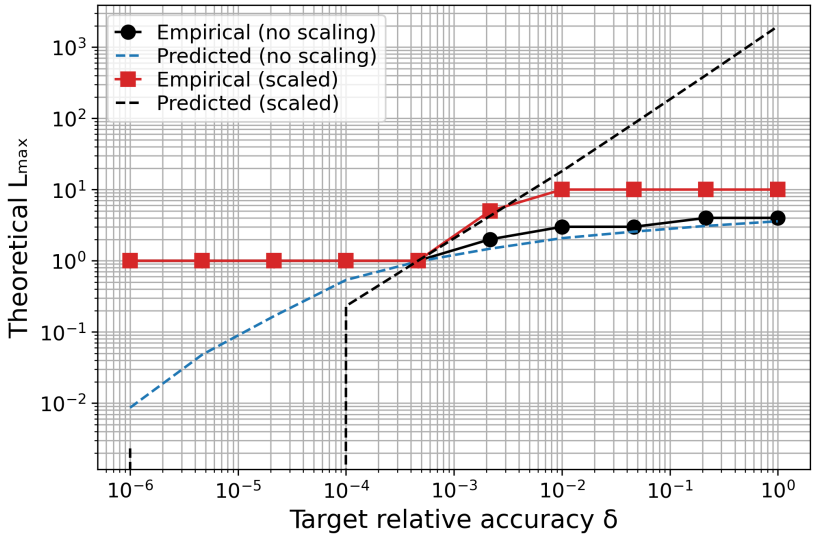


Figure 6: Stable depth $L_{\max}$: predicted vs. measured across δ.

7.1 Experimental setup

All experiments were executed on a Ray cluster (`Ray 2.39.0`) with 16 CPU cores per worker (mix of `AMD EPYC 9554` and `Intel Xeon Gold 6338N`) and 400 GB RAM. We implemented the error-bound analysis, the CKKS estimator, and the sweep harness in `Python 3.11.10`. The homomorphic encryption library is `TenSEAL 0.3.15` [BDPea21] built on `Microsoft SEAL 4.1.2` [SEA23].

7.2 Runtime and ciphertext footprint

To evaluate the performance improvements of the LRF integration into the CKKS pipeline, a sweep test across variable number of matrices $L = [2, 8]$ and dimensions $n = [16, 32, 48, 64]$ for square $n \times n$ matrices was conducted. The results are offered in Table 1 in terms of the Geometric mean (GM), median, and range [min,max] for the speedup (SU) and payload reduction (PR) factors of LRF vs. Direct Serial (DS) and Direct Tree (DT) scenarios. Each cell is formatted as *GM / median* on the first line and $[min, max]$ on the second. We report GM/median to summarize typical multiplicative gains and $[min, max]$ to show variability across n. The heatmaps in Figure 7 offer an alternative view of the speedup factors and payload reduction factors (accounting for the encryption of all input matrices).

The results demonstrate that the LRF approach consistently outperforms both direct and tree multiplications, with speedups ranging from ~4× to ~100× over direct trees and more than ~300× over direct serial, depending on the dimension and chain length. The LRF maintains comparable or slightly worse payload usage for smaller matrices but achieves significant savings when the number and size of matrices increase. This is an important outcome from a practical deployment perspective, as ciphertext size directly impacts storage and communication overheads in FHE applications.

For completeness, Table 2 reports the CKKS parameters selected by the estimator (polynomial modulus degree N, moduli chains) for representative configurations of (L, n, r). These parameters illustrate how the balancer adapts to different scenarios to maintain the desired accuracy while optimizing performance.

Table 1: Speed-up and encrypted payload reduction factors of LRF vs. direct serial and direct tree baselines.

	SU(×) t_{D*}/t_{LRF}		**PR(×)** m_{D*}/m_{LRF}	
L	**DS**	**DT**	**DS**	**DT**
2	19.19 | 19.67 [8.11, 43.52]	21.30 | 31.53 [5.45, 42.78]	1.59 | 1.77 [0.89, 2.32]	1.59 | 1.77 [0.89, 2.32]
3	20.53 | 14.10 [9.71, 99.17]	24.38 | 22.89 [8.00, 98.57]	2.10 | 2.35 [1.10, 3.28]	2.10 | 2.35 [1.10, 3.28]
4	12.55 | 8.56 [5.52, 65.71]	12.90 | 10.83 [4.46, 56.40]	1.24 | 1.40 [0.63, 2.01]	0.97 | 1.10 [0.48, 1.57]
5	16.81 | 11.81 [6.70, 91.32]	16.60 | 13.76 [5.49, 78.91]	1.57 | 1.77 [0.77, 2.58]	1.29 | 1.45 [0.64, 2.12]
6	68.73 | 90.57 [15.43, 213.95]	25.36 | 33.89 [5.61, 78.80]	3.07 | 3.54 [1.63, 4.39]	1.16 | 1.31 [0.56, 1.96]
7	91.72 | 121.95 [20.06, 286.66]	28.12 | 37.12 [6.27, 87.47]	4.22 | 5.12 [1.89, 6.98]	1.18 | 1.33 [0.57, 2.01]
8	110.55 | 146.31 [24.60, 339.27]	29.73 | 39.38 [6.57, 91.65]	4.04 | 4.51 [2.16, 6.10]	1.20 | 1.35 [0.57, 2.05]

Table 2: Representative CKKS parameters

		Poly modulus degree (N)			**Moduli chains**		
L	n	**Dense**	**Dense (tree)**	**LRF (tree)**	**Dense**	**Dense (tree)**	**LRF (tree)**
2	32	8192	8192	8192	[37,27,37]	[37,27,37]	[38, 28×3, 38]
2	64	8192	8192	8192	[37,27,37]	[37,27,37]	[39, 29×3, 39]
4	32	8192	8192	16384	[38, 28×3, 38]	[38, 28×2, 38]	[39, 29×5, 39]
4	64	8192	8192	16384	[39, 29×3, 39]	[38, 28×2, 38]	[40, 30×5, 40]
6	32	16384	8192	16384	[39, 29×5, 39]	[38, 28×3, 38]	[39, 29×6, 39]
6	64	16384	8192	16384	[40, 30×5, 40]	[39, 29×3, 39]	[40, 30×6, 40]
8	32	16384	8192	16384	[40, 30×7, 40]	[39, 29×3, 39]	[40, 30×6, 40]
8	64	16384	8192	16384	[40, 30×7, 40]	[39, 29×3, 39]	[40, 30×6, 40]

7.3 Noise-balance behavior

A noise balance sweep was also conducted for $L = [2, 6]$ and $n = [16, 32, 48, 64]$ to validate the balancer's ability to converge to the desired regime where LRF and CKKS errors are comparable. Figure 8 illustrates the trajectories of the noise balance sweeps, confirming that the analytical balancer effectively guides the selection of parameters to achieve balanced noise contributions from both sources ($\varepsilon_{\text{LRF}} \approx \varepsilon_{\text{CKKS}}$).

7.4 Trade-offs and limitations

The current implementation is limited to moderate chain lengths and matrix sizes due to memory constraints in existing FHE libraries and the overhead of materializing large ciphertext objects. Rank r is the primary knob controlling approximation quality: in our tested datasets, $r = 4$ provides a favorable accuracy–performance trade-off. Increasing r (e.g., to 6–8) reduces the tail ratio τ but increases the cost of internal $r \times r$ bridge products and associated ciphertext operations. Similarly, tighter targets (e.g., $\delta = 10^{-4}$) require larger CKKS parameters, increasing ciphertext size and runtime; nevertheless, LRF retains substantial speedups relative to direct multiplication in our experiments.

Finally, our sweeps encrypt all input matrices, representing a conservative worst case. In many applications some matrices are available in plaintext, enabling further gains by reducing the number of ciphertext encryptions and ciphertext–ciphertext multiplications.

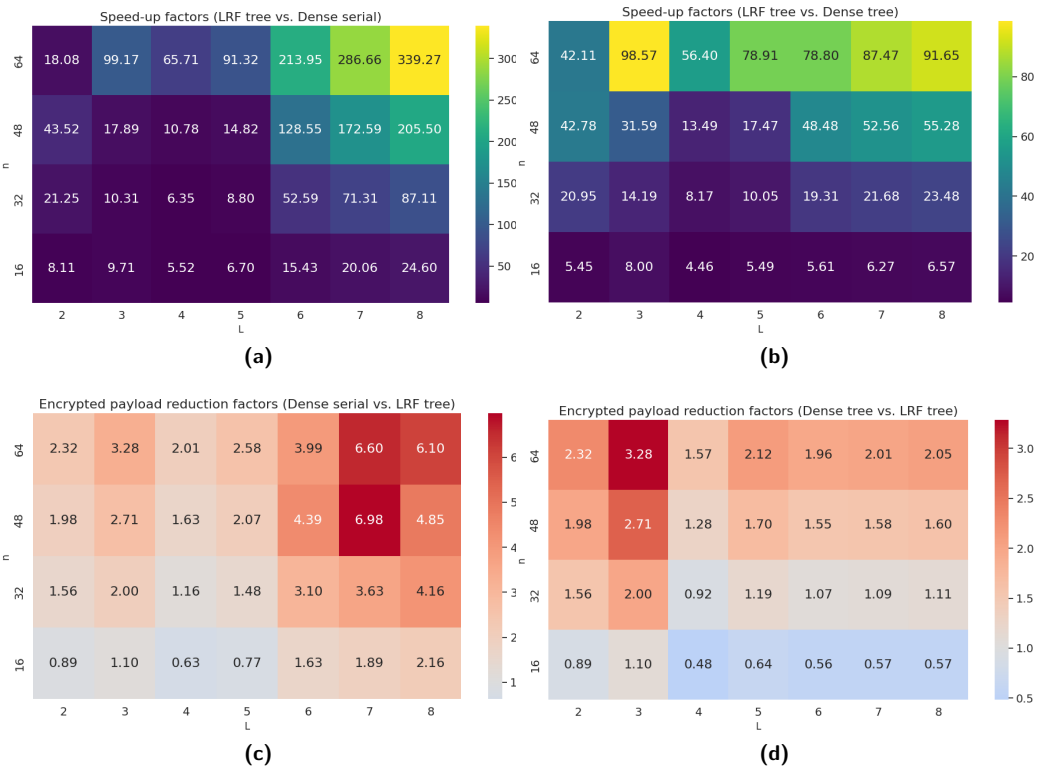


Figure 7: Runtime and ciphertext size heatmaps for varying number of matrices $L = [2, 8]$ and dimensions $n = [16, 32, 48, 64]$ (rank 4).

8 Conclusions

This work introduced cFHE, a co-design framework that integrates low-rank matrix factorization into CKKS parameter selection. The key contributions presented are (i) an error-propagation model for low-rank matrix chains that yields explicit stable-depth bounds under spectral amplification; (ii) a precision-balancing rule that maps measured factorization quality and a target tolerance to concrete CKKS parameters (scale, modulus chain, and N); and (iii) a binary-tree contraction strategy that reduces effective multiplicative depth to logarithmic in the chain length. Experiments show that cFHE substantially accelerates encrypted matrix-chain multiplication, reduces ciphertext footprint for moderate-to-large chains, and meets prescribed accuracy targets.

Future work includes multi-round extensions without bootstrapping via repeated balancing, mixed plaintext–ciphertext chains, and broader (including variable-rank) sweeps to characterize the accuracy–performance trade-off.

References

[ACJ24] Rashmi Agrawal, Anantha Chandrakasan, and Ajay Joshi. HEAP: A Fully Homomorphic Encryption Accelerator with Parallelized Bootstrapping. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 756–769. IEEE, jun 29 2024. URL: http://dx.doi.org/10.1109/ISCA59077.2024.00060, doi:10.1109/isca59077.2024.00060.

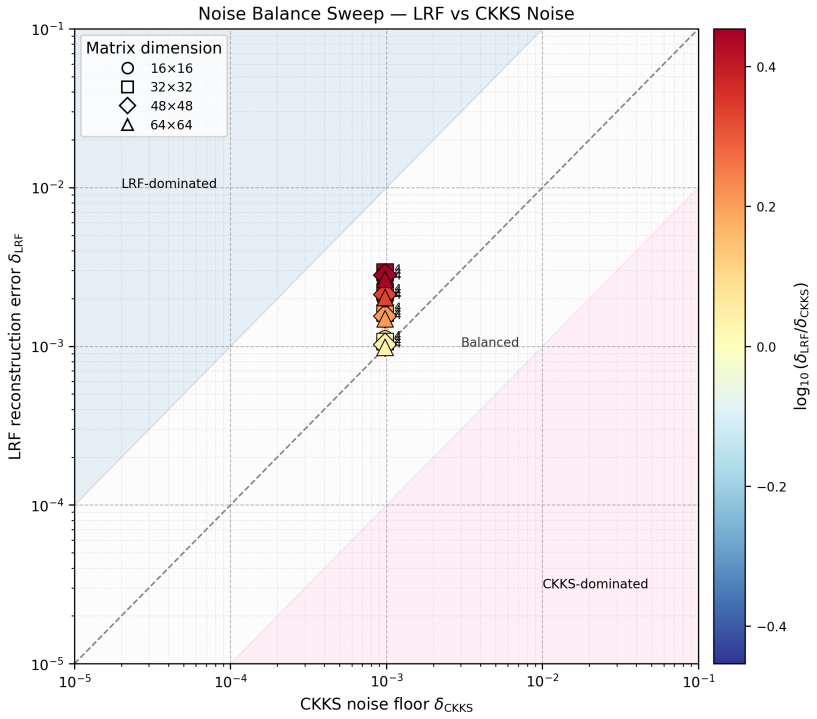


Figure 8: Noise-balance sweep toward $\varepsilon_{\mathrm{LRF}} \approx \varepsilon_{\mathrm{CKKS}}$.

[ADCJ+23] Rashmi Agrawal, Leo De Castro, Chiraag Juvekar, Anantha Chandrakasan, Vinod Vaikuntanathan, and Ajay Joshi. MAD: Memory-Aware Design Techniques for Accelerating Fully Homomorphic Encryption. In *56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 685–697. ACM, oct 28 2023. URL: http://dx.doi.org/10.1145/3613424.3614302, doi:10.1145/3613424.3614302.

[AdCY+22] Rashmi Agrawal, Leo de Castro, Guowei Yang, Chiraag Juvekar, Rabia Yazicigil, Anantha Chandrakasan, Vinod Vaikuntanathan, and Ajay Joshi. Fab: An fpga-based accelerator for bootstrappable fully homomorphic encryption, 2022. URL: https://arxiv.org/abs/2207.11872, arXiv:2207.11872.

[ADVLG+25] Carlos Agulló-Domingo, Óscar Vera-López, Seyda Guzelhan, Lohit Daksha, Aymane El Jerari, Kaustubh Shivdikar, Rashmi Agrawal, David Kaeli, Ajay Joshi, and José L. Abellán. Fideslib: A Fully-Fledged Open-Source FHE Library for Efficient CKKS on GPUs. In *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 1–3. IEEE, may 11 2025. URL: http://dx.doi.org/10.1109/ISPASS64960.2025.00045, doi:10.1109/ispass64960.2025.00045.

[AKP25] Andreea Alexandru, Andrey Kim, and Yuriy Polyakov. General functional bootstrapping using CKKS. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 304–337, Cham, 2025. Springer Nature Switzerland.

[ASD25] George Alexakis, Dimitrios Schoinianakis, and Giorgos Dimitrakopoulos. High-performance pipelined NTT accelerators with homogeneous digit-serial

modulo arithmetic. In *2025 28th Euromicro Conference on Digital System Design (DSD)*, pages 394–401, 2025. doi:10.1109/DSD67783.2025.00061.

[BCCW19] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. ngraph-HE2. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 45–56. ACM, nov 11 2019. URL: http://dx.doi.org/10.1145/3338469.3358944, doi:10.1145/3338469.3358944.

[BDPea21] Ahmed Benaissa, Morten Dahl, Bogdan C. Popa, and et al. TenSEAL: A library for encrypted tensor operations using homomorphic encryption. *arXiv preprint arXiv:2104.03152*, 2021. URL: https://arxiv.org/abs/2104.03152.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology – ASIACRYPT 2017*, volume 10624 of *LNCS*, pages 409–437. Springer, 2017. doi:10.1007/978-3-319-70694-8_15.

[CKKS18] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. A full RNS variant of approximate homomorphic encryption. In *Selected Areas in Cryptography – SAC 2018*, volume 11349 of *LNCS*, pages 347–368. Springer, 2018. doi:10.1007/978-3-030-10970-7_17.

[DFH+24] Xianglong Deng, Shengyu Fan, Zhicheng Hu, Zhuoyu Tian, Zihao Yang, Jiangrui Yu, Dingyuan Cao, Dan Meng, Rui Hou, Meng Li, Qian Lou, and Mingzhe Zhang. Trinity: A General Purpose FHE Accelerator. In *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 338–351. IEEE, nov 2 2024. URL: http://dx.doi.org/10.1109/MICRO61859.2024.00033, doi:10.1109/micro61859.2024.00033.

[DMGS23] Stefano Di Matteo, Matteo Lo Gerfo, and Sergio Saponara. VLSI Design and FPGA Implementation of an NTT Hardware Accelerator for Homomorphic SEAL-Embedded Library. *IEEE Access*, 11:72498–72508, 2023. URL: http://dx.doi.org/10.1109/ACCESS.2023.3295245, doi:10.1109/access.2023.3295245.

[EY36] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. doi:10.1007/BF02288367.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, Stanford, CA, USA, 2009.

[HOMK+24] Pengzhou He, Samira Carolina Oliva Madrigal, Çetin Kaya Koç, Tianyou Bao, and Jiafeng Xie. CASA: A Compact and Scalable Accelerator for Approximate Homomorphic Encryption. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(2):451–480, mar 12 2024. URL: http://dx.doi.org/10.46586/tches.v2024.i2.451-480, doi:10.46586/tches.v2024.i2.451-480.

[JKA+21] Wonkyung Jung, Sangpyo Kim, Jung Ho Ahn, Jung Hee Cheon, and Younho Lee. Over 100x Faster Bootstrapping in Fully Homomorphic Encryption through Memory-centric Optimization with GPUs. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 114–148, aug 11 2021. URL: http://dx.doi.org/10.46586/tches.v2021.i4.114-148, doi:10.46586/tches.v2021.i4.114-148.

[KPP20] Andrey Kim, Antonis Papadimitriou, and Y. Polyakov. Approximate homomorphic encryption with reduced approximation error. *IACR Cryptology ePrint Archive*, 2020.

[LED+23] Haohao Liao, Mahmoud A. Elmohr, Xuan Dong, Yanjun Qian, Wenzhe Yang, Zhiwei Shang, and Yin Tan. Turbohe: Accelerating Fully Homomorphic Encryption Using FPGA Clusters. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 788–797. IEEE, 5 2023. URL: http://dx.doi.org/10.1109/IPDPS54959.2023.00084, doi:10.1109/ipdps54959.2023.00084.

[LLK+20] Yongwoo Lee, Joon-Woo Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and HyungChul Kang. High-precision bootstrapping for approximate homomorphic encryption by error variance minimization. Cryptology ePrint Archive, Paper 2020/1549, 2020. URL: https://eprint.iacr.org/2020/1549.

[LLK+22] Yongwoo Lee, Joon-Woo Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and HyungChul Kang. High-precision bootstrapping for approximate homomorphic encryption by error variance minimization. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 551–580, Cham, 2022. Springer International Publishing.

[LLSL24] Chulwoo Lee, Hanyoung Lee, Ardianto Satriawan, and Hanho Lee. Configurable Arithmetic Core Architecture for RNS-CKKS Homomorphic Encryption. *IEEE Access*, 12:147220–147234, 2024. URL: http://dx.doi.org/10.1109/ACCESS.2024.3473903, doi:10.1109/access.2024.3473903.

[NKL23] Tuy Tan Nguyen, Jisu Kim, and Hanho Lee. CKKS-Based homomorphic encryption architecture using parallel NTT multiplier. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2023. doi:10.1109/ISCAS46773.2023.10181714.

[PPK24] Prasetiyo, Adiwena Putra, and Joo-Young Kim. Morphling: A throughput-maximized TFHE-based accelerator using transform-domain reuse. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 249–262, 2024. doi:10.1109/HPCA57654.2024.00028.

[SEA23] Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, January 2023. Microsoft Research, Redmond, WA.

[YYZ+23] Zepeng Yang, Chen Yang, Fahong Zhang, Jianfei Wang, Jia Hou, and Yang Su. An Acceleration Architecture of Polynomial Multiplication based on NTT for CKKS Homomorphic Evaluation. In *2023 8th International Conference on Integrated Circuits and Microsystems (ICICM)*, pages 229–233. IEEE, oct 20 2023. URL: http://dx.doi.org/10.1109/ICICM59499.2023.10365797, doi:10.1109/icicm59499.2023.10365797.