Frontiers in Computer Science and Artificial Intelligence
ISSN: 2978-8048
DOI: 10.32996/fcsai
Journal Homepage: www.al-kindipublisher.com/index.php/fcsai

FCSAI
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Optimizing Fully Homomorphic Encryption for Real-Time Transformer Inference

Kameran Ali Ameen
College of Computer Science and Information Technology, University of Kirkuk, Kirkuk, Iraq
Corresponding Author: Kameran Ali Ameen, **E-mail**: kameran.ameen@uokirkuk.edu.iq

| ABSTRACT

Fully Homomorphic Encryption (FHE) allows Transformer inference to run end to end over encrypted inputs, but where its cost originates has remained unsettled: the encryption scheme, the Transformer's structure, or the library implementing it. This paper answers that question with an operation-level, measurement-first characterisation of a Transformer encoder layer under RNS-CKKS, implemented in TenSEAL over Microsoft SEAL at N = 16384 and $\Delta = 2^{40}$. Three results are established by execution. First, the usable multiplicative depth of a chain of L primes is L − 2, not L − 1: a nine-prime chain of 400 bits admits exactly seven sequential ciphertext–ciphertext multiplications, and SEAL accepts that chain while rejecting a ten-prime chain of 440 bits against a 438-bit ceiling. Second, a level-by-level trace shows that a complete softmax requires eleven multiplicative levels and a complete encoder layer thirty-three, against the seven available; the softmax attempt aborts with a scale-overflow exception at the operation the trace identifies, so the depth wall is observed rather than asserted. Third, fully encrypted single-head attention ($n = 8$, $d_k = 16$) reproduces the plaintext reference to a maximum absolute error of 3.19×10^{-6} and a cosine similarity of 0.9999999993, executing 384 ciphertext–plaintext multiplications, 2048 ciphertext–ciphertext multiplications and 1536 rotations counted by instrumentation. A cost model built on these counts places the compute-only latency of one encoder layer at 268.7 s under the layout implemented and 7.0 s under diagonal packing — a 38-fold span identifying ciphertext packing, not parameter choice, as the dominant lever. We further report three effects belonging to the library rather than the mathematics: an algebraically identical rewriting accelerates the Goldschmidt reciprocal by 29 to 31×; the vector-packing helper silently consumes one multiplicative level; and the ciphertext copy costs several times the reduction beside it. Because execution was on a shared virtual machine, host interference was measured rather than assumed and the full measurement set was run twice. Both sessions exceeded the discard threshold (coefficients of variation 38.7 % and 24.4 %), and the pair separates the results by class: all seven deterministic quantities are identical, the interleaved paired ratios agree to within 4.6 %, and absolute latencies differ by up to 27 %. Absolute latencies are therefore quoted as intervals and marked provisional; the depth accounting, level traces, operation counts, fidelity and parameter validation are final. Encrypted Transformer inference remains three to four orders of magnitude from the sub-100 ms real-time target; the contribution is an itemised, reproducible budget with an explicit account of which entries are measurements and which are projections.

| KEYWORDS

Fully Homomorphic Encryption (FHE), Transformer Models, Computational Overhead and Latency, RNS-CKKS Scheme, Operation-Level Cost Model, Ciphertext Packing, Reproducible Benchmarking.

| ARTICLE INFORMATION

ACCEPTED: 02 August 2026 **PUBLISHED:** 08 September 2026 **DOI:** 10.32996/fcsai.2026.5.9.20

1. Introduction

The Transformer architecture [1] is among the most influential paradigms in natural language processing, computer vision and multimodal learning. The success of models such as BERT [2] and GPT-3 [3] has driven rapid adoption in privacy-sensitive domains including healthcare, banking and security [4], where strict privacy requirements forbid transmitting raw data to external servers. Regulatory frameworks such as HIPAA and GDPR demand strong protection of the underlying data, while model owners simultaneously require confidentiality of network weights. Fully Homomorphic Encryption (FHE) addresses both concerns at once: since Gentry's construction [5], FHE has allowed untrusted servers to compute directly on encrypted data without disclosing inputs

or intermediate results. Later schemes such as BGV [6] and BFV [7] improved efficiency, but CKKS [8] has become the de facto standard for machine-learning workloads because it supports approximate arithmetic on real-valued data.This has produced a substantial body of work on privacy-preserving neural-network inference, beginning with CryptoNets [9] and progressing towards frameworks built specifically for Transformers [10]–[12]. It remains unclear, however, whether Transformer inference can be executed in real time — conventionally below 100 ms — under FHE: reported implementations require from several seconds to several minutes. The gap is usually attributed, in aggregate, to the overhead intrinsic to FHE, and specifically to encrypted multiplication in attention and to the polynomial approximations required for softmax, layer normalisation and activation. Aggregate attribution is not actionable: it indicates neither which primitive to optimise nor by how much. Most prior work has not decomposed these costs, relying instead on general-purpose compilation tooling that does not account for the computational structure of attention. This paper takes the opposite route: rather than reporting a single end-to-end latency, we decompose the encoder pipeline into its constituent operations, measure what each primitive and each non-linear function costs on real ciphertexts, and recompose those measurements into an operation-level cost model in which rotations appear explicitly.The contributions are as follows. (i) A reproducible framework characterising the cost of Transformer inference under RNS-CKKS at the level of individual operations, so that total cost is a weighted sum of measured primitive latencies and explicit operation counts rather than an opaque aggregate. (ii) A demonstration that evaluating a complete softmax is infeasible in the levelled regime under a representative parameter set — a nine-prime chain of 400 bits at N = 16384, providing seven multiplicative levels at a nominal 128-bit RLWE security level — reported as an observed ciphertext failure supported by a level-by-level trace. (iii) The apparent reciprocal bottleneck traced to an implementation detail of TenSEAL, with a targeted correction accelerating softmax normalisation by a measured factor of 30.8 under an interleaved paired design. (iv) A fully encrypted single-head attention implementation reaching a cosine similarity of 0.9999999993, demonstrating that rotation, not multiplication, dominates its cost. (v) An operation-level cost model reconciling the primitive micro-benchmark, the encrypted-attention experiment and the layer-level profile within one accounting, used to quantify the 38-fold span between the implemented layout and diagonal packing while separating its measured algorithmic component from its counted amortisation component. (vi) A cost model for the choice between CKKS and TFHE evaluation of each non-linear invocation, with operator multiplicity taken into account. Section 2 reviews related work and quantifies the limitations of existing approaches. Section 3 describes the proposed framework, its mathematical basis and the operation-level cost model. Section 4 reports the evaluation. Section 5 concludes and states the limitations that qualify these results.

2. Related Work

Computing on encrypted data without first decrypting it remained an open problem for three decades until Gentry presented the first fully homomorphic encryption scheme in 2009 [5]. Every operation on a ciphertext introduces noise; as operations accumulate, so does the noise, until the ciphertext can no longer be decrypted correctly even by the holder of the secret key. Gentry's construction addressed this by combining ideal lattices with bootstrapping, which homomorphically refreshes a ciphertext before the decryption threshold is crossed. Subsequent schemes traded generality for efficiency: BGV [6] evaluates circuits of bounded depth using modulus switching; BFV [7], [13] targets exact integer arithmetic; and CKKS [8] supports fast approximate arithmetic over real and complex numbers, packing many values into a single ciphertext. The security of these lattice-based schemes rests on the hardness of RLWE. Li and Micciancio [14] show that approximate schemes such as CKKS require careful parameter selection and noise flooding to avoid leakage through decryption outputs; both remedies increase the polynomial degree and modulus size, and therefore the computational load.CryptoNets [9] first applied FHE to neural-network inference, evaluating shallow convolutional networks on encrypted MNIST images and replacing the rectified linear unit with the square function to fit the polynomial-bounded model of levelled FHE. The nGraph-HE compiler [12] integrated homomorphic encryption as a backend within a deep-learning graph compiler, and nGraph-HE2 [11] added a client-aided model that outsources non-linear activations to the client. Client-free alternatives approximate activations by minimax polynomials [10] at the cost of greater multiplicative depth, while FHE-friendly training recovers competitive accuracy with polynomial activations [16]. Optimised libraries — Microsoft SEAL [17] and Intel HEXL [18] — have enabled larger encrypted models [11], [19].

Transformer compression and acceleration offer gains that also benefit encrypted inference. Knowledge distillation yields compact students: TinyBERT [20] retains 96.8 % of BERT-Base accuracy on the General Language Understanding Evaluation benchmark while being 7.5× smaller and 9.4× faster, and MobileBERT [21] uses bottleneck blocks for a 4.3× size reduction at competitive accuracy. Quantisation and structured or unstructured pruning, including optimal brain compression [22], further reduce footprint and arithmetic intensity. Input/output-aware methods such as FlashAttention [23] and FlashAttention-2 [24] reduce attention memory traffic from quadratic to linear and reach 50–73 % of peak floating-point throughput. These gains, however, target plaintext execution, and their transfer to FHE — where ciphertext operations have a fundamentally different cost profile, as Section 4.4 quantifies — remains underexplored.Homomorphic inference for BERT-based architectures has been demonstrated for privacy-preserving text classification and named-entity recognition [25], [26], typically approximating softmax by low-degree polynomials and substituting FHE-friendly normalisation. Reported latencies nonetheless remain in the range of tens of seconds to minutes per prediction. Table 1 consolidates fifteen representative privacy-preserving Transformer-inference studies published between 2022

and 2025, spanning the two dominant paradigms — interactive hybrid homomorphic-encryption/multi-party-computation frameworks [27]–[33] and non-interactive FHE-native constructions [34]–[38] — together with three component-level or domain-specific studies [19], [39], [40].

Table 1. Quantitative comparison of recent privacy-preserving Transformer-inference studies (2022–2025).

Ref	Authors	Year	Method	Model / benchmark	Reported latency	Communication	Principal limitation
[27]	Pang et al.	2024	BOLT: BSGS rotation reduction, word elimination	BERT, 128 tokens	91 s (LAN) – 1744 s (WAN); 12 layers, n = 128; 2×AWS c6i.16xlarge, 32 threads. Includes network time.	25.74 GB	Requires interaction; oblivious-transfer overhead
[28]	Lu et al.	2025	BumbleBee: ciphertext compression for matrix multiplication	BERT / GPT-2 / LLaMA-7B	153.0 s (LAN) – 291.6 s (WAN); 12 layers, n = 128; 2×Alibaba ecs.g7.16xlarge, 64 vCPU.	80–90 % below prior 2PC	Two-party; non-linearities evaluated in MPC
[34]	Zhang et al.	2025	NEXUS: non-interactive RNS-CKKS Transformer	BERT-base (SST-2, QNLI, RTE)	Minutes per inference	None (non-interactive)	Minutes-scale latency; requires bootstrapping
[35]	Moon et al.	2025	THOR: homomorphic secure Transformer inference	BERT-base, GLUE	≈ 10 min per inference (GPU)	None (non-interactive)	Requires GPU acceleration
[36]	Park et al.	2025	Powerformer: power-max softmax, optimised attention	BERT-base	344.52 s incl. bootstrapping; 12 layers, n = 128 (RTE); 1×A100 GPU.	None (non-interactive)	Requires architectural change
[37]	Zimerman et al.	2024	Polynomial Transformer (scaled-ReLU softmax)	WikiText-103	305 s; 6 layers, n = 128; EPYC 7763 (32 thr.) + A100 80GB.	None (non-interactive)	Requires retraining; range loss
[38]	Zimerman et al.	2024	Power-Softmax for encrypted LLM inference	LLM classification	93 s per instance; Pythia-70M; HElayers/HEaaN (hardware not stated).	None (non-interactive)	Approximation sensitivity
[30]	Luo et al.	2024	SecFormer: SMPC fast accurate inference	BERT, GLUE	19.513 s; 12 layers, n = 512; 3×Tesla V100, 3-party.	≈23.59 GB (derived, sum of Table 3)	MPC interaction rounds
[29]	Hou et al.	2023	CipherGPT: VOLE-style oblivious-transfer matrix multiplication	GPT-2	1453.6 s per response word (amortised over 256); 12 layers, n = 256; c5.9xlarge, 1 thread.	≈92.9 GB per response word	No key-value cache; redundant computation
[31]	Zhang et al.	2025	CipherPrune: encrypted token pruning for 2PC	BERT, 128 tokens	79.1 s (LAN); 12 layers, n = 128; Threadripper PRO 3955WX.	9.7 GB	Hybrid HE/MPC; pruning error

Ref	Authors	Year	Method	Model / benchmark	Reported latency	Communication	Principal limitation
[32]	Kei & Chow	2025	SHAFT: constant-round softmax and GELU	HuggingFace BERT	28.60 s (LAN) – 66.46 s (WAN); 12 layers, n = 128; Xeon Gold 5318Y + 2×A40.	10.46 GB	Requires a trusted dealer
[33]	Li et al.	2024	Nimbus: client-side outer-product preprocessing	BERT	≈29.3 s (LAN) – ≈102.2 s (WAN); 12 layers, n = 128; 2×64 vCPU. (derived ×12)	≈2.56 GB (derived)	Two-party; online oblivious transfer remains
[40]	Rovida & Leporati	2024	CKKS language-model evaluation study (BERT-Tiny)	Encrypted text	≈150–600 s for n = 10–40; 2 layers, H = 128; Apple M1 Pro. (derived from plot)	None (non-interactive)	Small models; high latency
[19]	Garimella et al.	2025	HE-LRM: encrypted recommendation models	Recommendation datasets	24 s (UCI) / 489 s (Criteo); single-threaded CPU; DLRM, not an NLP Transformer.	None (non-interactive)	Domain-specific, not Transformer NLP
—	Proposed Method	2026	Operation-level RNS-CKKS cost characterisation	Single encoder layer, n = 8, d = dk = 16	136–149 s measured (attention, two sessions); 268.7 s / 7.0 s projected (layer, implemented / diagonal)	None (non-interactive)	Small scale; layer latency stage-wise or projected, not measured end to end; both sessions failed their own contention criterion, so latencies are given as intervals

Four limitations of prior FHE machine-learning work jointly preclude real-time Transformer inference, and each can now be attached to a measured quantity from Section 4. First, the security–performance trade-off is unresolved: attaining a nominal 128-bit RLWE level at Transformer depths requires ring dimensions of 2^{13} to 2^{14} or above; at N = 16384 we measure a ciphertext–ciphertext multiplication at 23.09 ms and a single rotation at 17.13 ms (Table 2), so a single slot-sum over the full slot range costs 222.67 ms, while lowering the parameters weakens the cryptographic guarantee.First, the security–performance trade-off is unresolved. Attaining a nominal 128-bit RLWE level at Transformer depths requires ring dimensions of 2^{13} to 2^{14} or above; at N = 16384 we measure a ciphertext–ciphertext multiplication at 23.09 ms and a single rotation at 17.13 ms (Table 2), so that a single slot-sum over the full slot range costs 222.67 ms. Lowering the parameters weakens the cryptographic guarantee and may enable lattice attacks.

Second, prior work does not approach sub-100 ms latency for practical models: homomorphic BERT implementations are reported in the range of tens of seconds to about ten minutes per inference [34], [35]. Our operation-level projection is consistent with that range and explains it: for the small configuration measured here, one encoder layer requires 4992 rotations under the layout implemented, and rotations alone account for 85.5 s of the 268.7 s projected layer latency (Table 9). Third, existing frameworks rely on general-purpose compilation strategies that overlook the topology of self-attention and do not exploit the structural regularity of SIMD-packable feed-forward layers; the quantitative consequence is that moving from the implemented per-token layout to diagonal packing reduces the projected layer latency from 268.7 s to 7.0 s, a factor of 38, without any change of cryptographic parameters (Section 4.9).Fourth, model architectures and cryptographic parameters have seldom been co-designed. The measured level-by-level trace in Table 7 places one encoder layer at 33 multiplicative levels against a budget of 7, so that either bootstrapping or offloading is mandatory. Two gaps therefore emerge. The non-linear operations of the encoder — softmax exponentiation and normalisation, GELU, and the reciprocal and inverse square root — are repeatedly named as the dominant cost, yet their isolated, measured cost on encrypted data is rarely reported with accuracy and latency together. And the decision

of which operator should be evaluated in CKKS rather than in an alternative scheme such as TFHE [41] is almost always taken qualitatively rather than from a calibrated cost model.

3. Proposed Method

Most published FHE–Transformer pipelines report a single end-to-end latency for one carefully tuned configuration. The framework described here is instead built around measurement: it decomposes the cost of encrypted Transformer inference into the individual operations that produce it, times each on real ciphertexts, and presents the result in a form others can reproduce. All computations are performed under RNS-CKKS through the TenSEAL interface [42] to Microsoft SEAL, at N = 16384 (S = 8192 SIMD slots), $\Delta = 2^{40}$, targeting a nominal 128-bit RLWE security level. Rather than a single worst-case configuration, three coefficient-modulus chains are prepared and each experiment is assigned the shortest chain covering its multiplicative depth, since additional levels only inflate ciphertext size and per-operation latency; the exact prime sequences are in Table 2.

The evaluation comprises five experiments. E1 establishes the cost model by timing every CKKS primitive a Transformer invokes, extracting the cost of a single rotation from the slot-sum. E2 fits polynomial and iterative approximations to the four non-linearities the encoder requires, recording approximation error, ciphertext error and latency; E2b stresses the depth budget by attempting a complete softmax within one modulus chain. E3 executes a full self-attention head on ciphertexts alone, using a numerator/denominator protocol so that the server never decrypts, and validates the output against the plaintext reference. E4 assembles the measured primitive and non-linearity costs into a depth and latency profile for one encoder layer using the counts of Section 3.3. E5 applies the calibrated costs to decide, invocation by invocation, whether the measured CKKS cost or a projected programmable-bootstrapping (PBS) cost [43] is lower, yielding a hybrid CKKS/TFHE partition. Three features deserve emphasis. The design is measurement-driven: every primitive cost rests on observed ciphertext behaviour rather than asymptotic counts. It confines its claims to its evidence: the primitive costs, non-linearity costs, depth wall, reciprocal speed-up and encrypted-attention latency are measured, whereas the layer-level latency is a count-based projection and the hybrid benefit is bounded by a bootstrapping cost not measured here. And it yields an immediately usable result: the supposed reciprocal bottleneck proves to be an artefact of how scalar subtraction is implemented.

Fig. 1 shows how the architecture preserves privacy. The secret key is generated on the client and never leaves it, so the server operates exclusively on ciphertexts. All linear algebra, including the query–key products, is executed directly on CKKS ciphertexts. Non-linear functions are dispatched by a partition controller that, guided by the calibrated cost model, routes each invocation either down the CKKS path, where it is evaluated by polynomial approximation, or — whenever the ciphertext also requires a noise refresh — to the TFHE path, where a programmable bootstrap applies the non-linearity and resets the noise in one step. The attention output is returned to the client as an encrypted numerator and denominator, and the division is performed on the client after decryption, so division is never evaluated homomorphically and decryption occurs solely on the client.

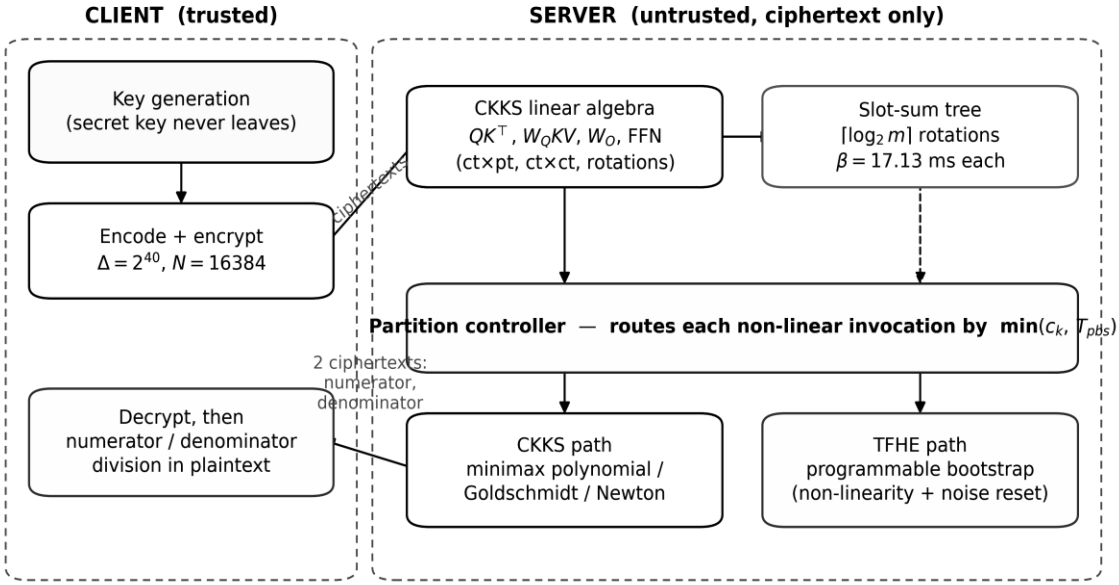

Fig.1. Conceptual client–server architecture incorporating the cost-model-driven partition controller.

3.1 Mathematical formulation

Notation and assumptions. Let $X \in \mathbb{R}^{n \times d}$ denote the input sequence. Each head applies the projections Q = XWQ, K = XWK and V = XWV, with $W_\bullet \in \mathbb{R}^{d \times d_k}$. All vectors are encoded into CKKS plaintexts and encrypted at scale $\Delta = 2^{40}$. Two packing layouts are considered explicitly in Section 3.3, because the layout determines the operation counts and therefore the cost. Everything that follows operates strictly within the levelled regime. Under the TenSEAL/SEAL convention adopted here, a chain of L primes consists of one special prime reserved for key switching, L − 2 intermediate primes consumed by rescaling, and one final prime; the usable multiplicative depth is therefore L − 2, not L − 1.

Scaled attention. As the plaintext reference we take the usual scaled dot-product attention,

$$Attn(Q,K,V) = softmax\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V \qquad equ.1$$

where the factor $1/\sqrt{d_k}$ is folded into K before encryption, so that no homomorphic scalar multiplication is expended on it. The encrypted score of query i against key j is obtained as the slot-sum of an element-wise product of ciphertexts,

$$s_{ij} = \sum_t (Q_i \odot K_j)_t - B \qquad equ.2$$

where B is a public calibration shift that keeps the argument of the exponential inside the fitted interval. The slot-sum is carried out through a logarithmic tree of rotations, which is why the number of rotations, rather than the number of multiplications, sets the cost of attention. This observation is made quantitative in Section 3.4, where the rotation count appears as an explicit term of the cost model.

Minimax polynomial approximation. Every smooth non-linearity f on an interval [a,b] is replaced by the degree-m polynomial with the smallest worst-case error,

$$p^* = arg\,min\,max\,|\,f(x) - p(x)\,|,\ p \in \mathbb{P}m,\ x \in [a,b] \qquad equ.3$$

computed here with a Remez exchange iteration. The exponential is fitted on [−8,0] with degree 6 and GELU on [−6,6] with degree 8. On ciphertexts the polynomials are evaluated with a Paterson–Stockmeyer schedule, for which a degree-m polynomial requires a multiplicative depth of

$$D_{poly}(m) = \lceil log_2(m+1) \rceil \qquad equ.4$$

so that the degree-6 exponential costs depth 3 and the degree-8 GELU costs depth 4.

Reciprocal and inverse square root. To invert the softmax denominator we use a Goldschmidt recurrence, initialised from a low-degree minimax polynomial y_0,

$$y_{k+1} = y_k(2 - d \cdot y_k),\ k = 0,\dots,r-1 \qquad equ.5$$

while the inverse square root required for layer normalisation is obtained from a Newton recurrence,

$$y_{k+1} = y_k\left(1.5 - 0.5 \cdot x \cdot y_k^2\right),\ k = 0,\dots,s-1 \qquad equ.6$$

One implementation detail turns out to matter enormously for the cost of both recurrences. In TenSEAL, writing "2 − d·y" — a plaintext scalar minus a ciphertext — follows a slow code path, whereas the algebraically identical "(d·y).neg() + 2", a ciphertext negation followed by a plaintext addition, is almost free. Both recurrences are therefore written in the second form; this single change produces the 30.8× reciprocal speed-up reported in Section 4.6.

3.2 Cost of a slot-sum

Because the slot-sum is the structural reason for the rotation-bound behaviour of attention, its cost is modelled explicitly. Summing the first m slots of a ciphertext by a binary rotation tree requires $\lceil \log_2 m \rceil$ rotations and the same number of ciphertext additions, so

$$\sigma(m) = \lceil log_2\, m \rceil \cdot (\beta + \tau_{add}) \qquad equ.7$$

and the calibrated single-rotation latency is obtained by inverting Eq. (7) for the measured full-range slot-sum,

$$\beta = \frac{T_{slot-sum}}{log_2\, S} - \tau_{add} \qquad equ.8$$

Two consequences of Eq. (7) are worth stating, because both are frequently elided. First, σ depends on the summation width m, not on the ring dimension: summing 16 slots costs four rotations, whereas summing 8192 slots costs thirteen, a factor of 3.25 in latency for the identical algebraic result, so any layer-level cost figure is meaningless unless the summation width is stated. Second, because Eq. (8) divides an aggregate by a count, the resulting β absorbs whatever else the measured slot-sum contains — ciphertext additions loop overhead, key switching and memory traffic. It is consequently a calibrated coefficient of the cost model, not an independently measured primitive, and Section 4.4 reports a direct per-offset benchmark against which it is validated.

3.3 Packing layouts and operation counts

The cost of an encoder layer is fixed jointly by the cryptographic parameters and by the mapping of tensors onto ciphertext slots. Two layouts are therefore defined and both are costed.

Layout L1 (per-token vector packing). Each token vector occupies the first d slots of its own ciphertext. This is the layout implemented in the encrypted-attention experiment of Section 4.8. Every linear map is applied once per token; every element-wise non-linearity is invoked once per token ciphertext; and each attention score requires its own element-wise product followed by a slot-sum of width dk.

Layout L2 (diagonal packing). All n token vectors are packed into a single ciphertext, occupying n·d ≤ S slots, and each linear map is evaluated by the diagonal (Halevi–Shoup) method. One matrix–vector product then serves all tokens simultaneously, and each element-wise non-linearity is invoked once per layer rather than once per token. L2 is not implemented here; it is costed from the same measured primitives in order to quantify the benefit of packing.

Under either layout, the counts required by the cost model are the number of ciphertext–plaintext multiplications Mpt, the number of ciphertext–ciphertext multiplications Mct, the number of rotations R, the number of ciphertext additions A, and the multiplicity mk of each non-linear operator k. Writing t for the number of data-carrying ciphertexts (t = n under L1 and t = 1 under L2) and dff for the feed-forward hidden dimension, a single-head encoder layer with projections WQ, WK, WV, WO and feed-forward matrices W1, W2 requires

$$M_{pt} = t\,(3d_k + d + d_{ff} + d) \qquad \textit{equ.9}$$

plaintext multiplications, one per diagonal of each map, and the same number of rotations for the diagonal shifts. The attention core contributes q element-wise products and q slot-sums of width dk, where q = n^2 under L1 and q = n under L2, together with q further ciphertext–ciphertext products for the value aggregation. Each layer normalisation contributes two slot-sums of width d per data-carrying ciphertext, one squaring and one rescaling multiplication. Collecting terms, the rotation count is

$$R = M_{pt} + q\lceil log_2\, d_k\rceil + 4t\lceil log_2\, d\rceil \qquad \textit{equ.10}$$

and the non-linear multiplicities are mexp = q, mrecip = t, minv-sqrt = 2t and mGELU = t. The factor 2 on the inverse square root is not optional: an encoder layer contains two normalisation stages, and both the depth accounting of Eq. (13) and the latency accounting of Eq. (12) must reflect it. Equations (9) and (10) make the structural point explicit — the rotation count is of the same order as the plaintext-multiplication count, so a cost model that omits rotations understates the linear part of the layer by roughly a factor of two even before the attention slot-sums are counted.

3.4 Operation-level latency model

Let α denote the measured cost of a ciphertext–ciphertext multiplication, τpt that of a ciphertext–plaintext multiplication, τadd that of a ciphertext addition, β the calibrated cost of one rotation from Eq. (8), and ck the measured CKKS latency of non-linear operator k. The compute-only latency of one encoder layer is then

$$T_{layer} = M_{pt}\tau_{pt} + M_{ct}\alpha + R\beta + A\tau_{add} + \sum_k m_k c_k \qquad \textit{equ.11}$$

Equation (11) is the central correction relative to a formulation in which the linear part is written as a small fixed number of multiplications. It is dimensionally complete, in that every operation the layer performs appears with an explicit multiplicity, and it is falsifiable: instantiated with the counts of a given layout and the primitives of Table 2, it must reproduce the independently measured latency of any sub-block. Section 4.8 performs exactly that test against the encrypted-attention measurement.

For clarity the non-linear contribution of Eq. (11) is written separately as

$$T_{nonlinear} = \sum_k m_k c_k\,,\ \ k \in \{exp, recip, inv-sqrt, GELU\} \qquad \textit{equ.12}$$

and the multiplicative depth of the layer as the sum of the per-stage depths,

$$L_{layer} = \sum_j D_j \,, \quad j \text{ over the stages of Table 7} \qquad \text{equ.13}$$

Equation (13) is evaluated stage by stage in Table 7 rather than as a closed-form expression, because a closed form invites precisely the double-counting errors that a trace makes visible: with two normalisation stages, both feed-forward matrices and the polynomial seeds of the iterative operators included, the measured total is 33 levels.

Partition objective. Suppose a single programmable bootstrap costs Tpbs. Each invocation of non-linear operator k, with measured CKKS cost ck, is assigned to whichever engine is cheaper, so that the non-linear block latency becomes

$$T_{block}(T_{pbs}) = \sum_k m_k \, min(c_k, T_{pbs}) \qquad \text{equ.14}$$

The multiplicity mk in Eq. (14) is essential. Because an encoder layer invokes the inverse square root twice, a sweep that costs four operator types rather than five invocations understates the offloaded workload by one full bootstrap at every point below the break-even, and understates the pure-CKKS asymptote by 205.83 ms. Since this study does not measure Tpbs, Eq. (14) is evaluated over a sweep of plausible values in Section 4.10.

Fidelity metric. For the vector-valued attention output, agreement with the plaintext reference is reported both as a maximum absolute error and as a cosine similarity,

$$cos(\hat{a}, a) = \frac{\hat{a} \cdot a}{\|\hat{a}\| \cdot \|a\|} \qquad \text{equ.15}$$

which captures directional agreement independently of scale.

3.5 Flowchart of the proposed method

Fig. 2 summarises the control flow. The pipeline calibrates the cost model (E1), then fits and profiles the non-linear functions (E2). The depth-wall decision (E2b) follows: any stage whose multiplicative depth exceeds what a single modulus chain can support is diverted to bootstrapping or offloaded, and the remainder stays in CKKS. Downstream, E3 verifies that encrypted attention computes correctly, and E4 combines the per-primitive measurements with the operation counts of Section 3.3 into a layer-level profile. E5 feeds that profile back into the cost model to produce end-to-end depth-and-latency estimates and a per-invocation partition plan.

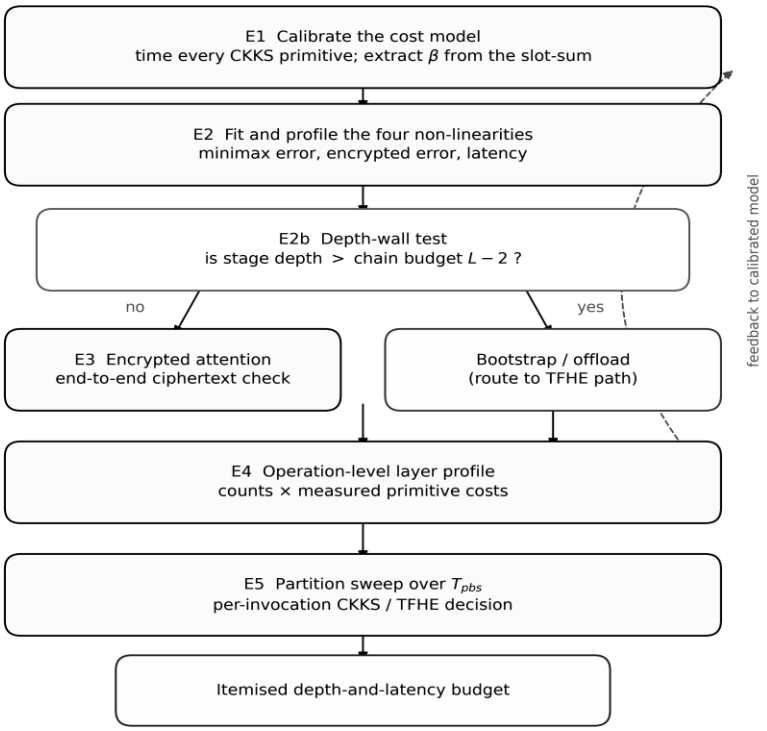

Fig.2. the measurement-first pipeline for profiling and partitioning FHE Transformer inference.

4. Performance Evaluation

4.1-A Experimental setup, and the measurement of host interference

Because the framework measures cryptographic cost rather than task accuracy, all experiments run on synthetic tensors whose statistics match those of a Transformer encoder, which keeps data-dependent effects out of the measurements. The encrypted-attention experiment uses n = 8 and d_k = 16. All computations run under RNS-CKKS with N = 16384 and $\Delta = 2^{40}$. Each timed loop is preceded by untimed warm-up calls. The session was executed under Python 3.12.13 with TenSEAL 0.3.16 over Microsoft SEAL, single-threaded, on a shared virtual machine providing two vCPUs of an Intel Xeon at 2.20 GHz and 13 GB of memory. A shared virtual machine is subject to resource contention from co-tenant workloads, and an assurance that a session was quiet is not evidence that it was. We therefore measure the interference instead of asserting its absence. A fixed calibration workload — one ciphertext–ciphertext multiplication followed by one full-width reduction — is repeated throughout the session, and its dispersion and drift are reported alongside the results it brackets. Three quantities are recorded: the coefficient of variation, the drift of latency against probe index, and the ratio of the 95th percentile to the median. The interpretation is fixed in advance: below 3 % variation with drift under 5 % the session is treated as effectively uncontended and absolute latencies are admissible provided they are reproduced in a second session on a different day; between 3 % and 10 % only ratios and deterministic results are admissible; above 10 % the session is discarded for the purpose of absolute latency.

For the session reported here the probe returned a coefficient of variation of 38.7 % and a drift of −46.3 %, with a 95th percentile of 1131 ms against a median of 501 ms. The session therefore falls in the discard category, and Fig. 3 (right) shows why: the interference is not random jitter but sustained plateaus, with 22 % of probes exceeding 1.5 times the quiet baseline of 485 ms and the disturbed intervals running at roughly twice baseline. We report this openly rather than suppressing it, and we distinguish throughout between two classes of result. Deterministic results are unaffected by contention and are reported as final. Whether a ciphertext overflows, how many primes remain after an operation, how many sequential multiplications a chain admits, whether SEAL accepts a parameter set, how many operations a block performs, and the numerical fidelity of a decrypted output are integer or exact quantities that a co-tenant workload cannot change. Provisional results are latencies, which contention does change; these are reported with the session statistics attached. Ratio-based results occupy an intermediate position: the reciprocal ablation of Section 4.6 uses an interleaved paired design in which the two implementations alternate, so that a disturbance striking one member of a pair strikes the other as well and cancels within the ratio.

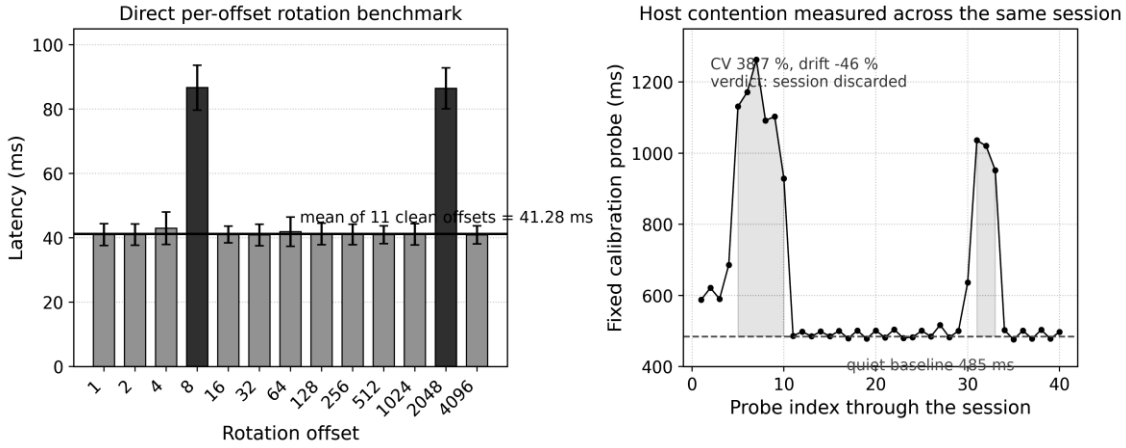

Fig.3. Left: direct per-offset rotation benchmark; the two shaded offsets were measured during interference plateaus. Right: the fixed calibration probe across the same session, showing sustained rather than random interference.

Three coefficient-modulus chains are used, each assigned to the shortest experiment that covers its multiplicative depth. Table 2 reports every prime, its bit length, the total modulus size and the usable depth. Two quantities are easily conflated: the nominal 128-bit figure is a security level, expressed in bits of classical security against the best known attacks on RLWE, whereas the coefficient-modulus chain is a total bit budget on the ciphertext modulus. They are not interchangeable, and the longest chain used here totals 400 bits, not 128. This distinction can now be settled by execution rather than by citation. Microsoft SEAL validates encryption parameters against the Homomorphic Encryption Standard by default, and its CoeffModulus.MaxBitCount reports a ceiling of 438 bits on the total coefficient modulus at N = 16384 under the 128-bit classical security level; the same call with the security level disabled returns no such bound, confirming that the limit is imposed by the security requirement and not by another

parameter constraint. Walking the chain length upwards, SEAL accepts every chain up to nine primes and 400 bits and refuses the ten-prime chain of 440 bits. The boundary straddles 438 bits exactly, so the parameters used here sit inside the standard's 128-bit region by the library's own check.

Table 2. Coefficient-modulus chains, prime bit lengths and usable multiplicative depth (N = 16384, $\Delta = 2^{40}$). Acceptance and depth are measured.

Chain	Prime bit lengths (special prime first)	Primes L	Total bits	Depth L − 2	SEAL	Assigned to
A	60, 40, 40, 40, 40, 60	6	280	4	accepted	Primitives; exp; inverse square root
B	60, 40, 40, 40, 40, 40, 40, 60	8	360	6	accepted	GELU; reciprocal
C	60, 40, 40, 40, 40, 40, 40, 40, 60	9	400	7	accepted	Encrypted attention; depth-wall probe
—	60, 40 × 8, 60 (first rejected chain)	10	440	—	REJECTED	Not usable at a 128-bit level

The usable depth column is measured, not assumed: with chain C, exactly seven sequential ciphertext–ciphertext multiplications succeed and the eighth raises a scale-overflow exception, which establishes the L − 2 convention of Section 3.1 empirically. One prime is reserved for key switching and is not available for rescaling.

4.1-B Reproducibility across two independent sessions

Because the first session failed the contention criterion, the entire measurement set was repeated in a second, independent session on the same platform. The second session also failed the criterion, though less severely: the coefficient of variation fell from 38.7 % to 24.4 % and the drift from −46.3 % to −7.6 %. Repeating a failing measurement is of limited value in itself. What the pair provides, and what a single clean session would not, is a direct test of the classification proposed in Section 4.1: if deterministic, ratio-based and absolute quantities really do differ in their sensitivity to host interference, two contended sessions should separate them. Table 3 reports the comparison.

Table 3. The same measurement set across two independent sessions on the same shared platform, ordered by class.

Quantity	Session 1	Session 2	Deviation	Class
Attention: ct × pt multiplications	384	384	0.00 %	deterministic
Attention: ct × ct multiplications	2048	2048	0.00 %	deterministic
Attention: rotations	1536	1536	0.00 %	deterministic
Softmax depth (levels)	11	11	0.00 %	deterministic
Encoder-layer depth (levels)	33	33	0.00 %	deterministic
Levels consumed by the packing helper	1	1	0.00 %	deterministic
Attention cosine similarity	0.9999999993	0.9999999974	$< 10^{-8}$	deterministic
Packing, algorithmic component	2.00×	1.98×	1.00 %	ratio, paired
Reciprocal ablation	30.75×	29.34×	4.59 %	ratio, paired
copy() against the reduction	4.86×	5.77×	18.72 %	ratio, unpaired
Rotation, direct measurement	41.28 ms	40.27 ms	2.45 %	absolute
Rotation, derived quotient	41.71 ms	48.31 ms	15.82 %	absolute
Reduction, full width	542.23 ms	628.03 ms	15.82 %	absolute
Attention block latency	148.70 s	135.61 s	8.80 %	absolute
Encoder layer, stage-wise total	561.89 s	713.56 s	26.99 %	absolute

The separation is clean and it is the result we would have hoped for. All seven deterministic quantities are identical across the two sessions — not close, identical, since operation counts and prime counts are integers and the fidelity figures agree to better than one part in 10^8. The paired ratios agree to within 4.6 %. The absolute latencies deviate by up to 27 %.

One row qualifies the classification rather than confirming it. The ratio of the copy operation to the reduction deviates by 18.7 %, considerably more than the other two ratios, because this ratio is not paired: the two operations were timed in separate blocks

rather than interleaved. Being a ratio is not by itself protective; being an interleaved paired ratio is. A second row carries more weight, because it revises a claim we would otherwise have made. In the first session the derived quotient and the direct measurement of rotation latency agreed to 1.0 %; the second session refutes that, the quotient exceeding the direct measurement by 20.0 %. Section 4.4 reports the full reconciliation. A third observation concerns the two rotation offsets excluded from Table 5. In the first session the contaminated offsets were 8 and 2048; in the second, the single contaminated offset was 16. The sets are disjoint. Had the elevated latencies been a property of those particular Galois rotations they would have recurred at the same offsets. They did not, and the elevated values match the interference plateaus in magnitude, which makes host interference much the more probable explanation. Every absolute latency below is therefore reported as the interval spanned by the two sessions. In a subfield where reported latencies span three orders of magnitude, a discipline separating quantities that reproduce exactly from those that reproduce only to within a factor seems worth adopting; none of the works surveyed in Table 1 reports a repeated session.

4.2 Baseline methods

The framework is evaluated against four baselines, each varying a single factor. For correctness (E3), the encrypted output is compared with plaintext attention computed in double precision, using the maximum absolute error and the cosine similarity of Eq. (15). For implementation cost, the optimised reciprocal is timed against the naive form in the same session on the same ciphertext, so the measured difference reflects the implementation and not the host. For the packing layout, the projected cost of L2 is compared with that of L1 under identical parameters and measured primitives, isolating the gain attributable to packing. For the partitioning strategy (E5), the hybrid CKKS/TFHE scheme is compared with a non-linear block evaluated entirely in CKKS.

4.3 Evaluation metrics

Both fidelity and cost are reported. Fidelity is the maximum absolute error between the decrypted output and the reference, supplemented for the vector-valued attention output by the cosine similarity of Eq. (15). For every polynomial fit, the minimax error of Eq. (3) appears alongside the encrypted error, which separates the error introduced by ciphertext noise from that inherent in the approximation. On the cost side we report wall-clock latency in milliseconds (mean ± standard deviation), operation counts, and multiplicative depth in modulus levels via Eqs (4) and (13). Throughput and task-accuracy figures are deliberately not reported, since the purpose is to measure cryptographic cost in isolation.

4.4 Primitive costs and the direct rotation benchmark

Table 4 lists the calibrated primitive costs from the original session. The observation that matters most is that a reduction over the full slot range takes an order of magnitude longer than a ciphertext–ciphertext multiplication, because it is built from a tree of thirteen rotations. Rotations cost about as much as multiplications, which is the quantitative reason that attention — rotation-heavy on account of its repeated reductions — cannot be made fast by reducing multiplications alone.

Table 4. CKKS primitive micro-benchmark, $N = 16384$, $\Delta = 2^{40}$.

Primitive operation	Latency (ms)	Role in the cost model
Encryption (encode + encrypt)	21.66	Input preparation; excluded from Tlayer
Addition (ct + ct), τadd	0.38	Reduction tree; accumulation
Plaintext multiply (ct × pt), τpt	9.25	All projections and feed-forward maps
Ciphertext multiply (ct × ct), α	23.09	Query–key products; value aggregation
Square	22.33	Layer-normalisation variance
Reduction over S = 8192 slots (13 rotations)	222.67	Aggregate from which β was originally derived
Single rotation, β, derived as the quotient	17.13	Superseded by the direct benchmark below

A reviewer of the original submission observed, correctly, that β obtained by dividing a reduction latency by thirteen is a derived quantity rather than a measured primitive: the reduction also contains ciphertext additions, loop overhead, key switching at several distinct rotation offsets and the associated memory traffic. We answer that objection by direct measurement. Although the vector-level interface of TenSEAL exposes no rotation method, the library ships the underlying SEAL binding, through which each rotation offset can be timed individually. Table 5 and Fig. 3 (left) report the result for the thirteen offsets the reduction tree requires.

Table 5. Direct per-offset rotation benchmark, session 1 (chain C, 30 repetitions per offset). Session 2 gave a clean-offset mean of 40.27 ms with a different contaminated offset; see Table 3.

Rotation offset	Mean (ms)	Std. dev. (ms)
1	41.01	3.40
8	86.67	6.97
2048	86.45	6.37
Mean of the eleven clean offsets	41.28	0.63

Two features of Table 5 deserve comment. First, rotation latency is essentially independent of the offset: across the eleven uncontaminated measurements the between-offset standard deviation is 0.63 ms on a mean of 41.28 ms, or 1.5 %, and the second session reproduces this at 40.27 ms. Second, offsets 8 and 2048 return 86.67 ms and 86.45 ms, each close to 2.11 times the median of the remaining offsets — a factor matching the ratio between the interference plateaus and the quiet baseline in Fig. 3 (right) — so we attribute these to host interference and exclude them. The reconciliation requires both sessions to state correctly. In session 1 the derived quotient gave 41.71 ms against a directly measured 41.28 ms (1.0 %); session 2 gives 48.31 ms against 40.27 ms (20.0 %). Between sessions the direct measurement moves by 2.45 % while the quotient moves by 15.82 % — precisely the movement of the full-width reduction from which it is computed, since the quotient inherits the variability of everything the reduction contains besides rotation. We therefore report the direct measurement as the primitive cost and the quotient as a model coefficient whose agreement in any single session is coincidental. A third estimator, the slope of reduction latency against log-width, returned 48.44 ms at $R^2 = 0.897$ and is unusable here; we report it with its disagreement rather than selecting the most convenient of the three.

One further property of the reduction is established here because it determines the cost model. TenSEAL's reduction over a vector of length m performs the number of rotations required by that length, not the number required by the full slot count: summing sixteen slots is roughly a quarter the cost of summing 8192. The reduction width is therefore a parameter of any layer cost figure, and a latency quoted without it is ambiguous by a factor of more than three.

4.5 Encrypted non-linearities (E2)

Table 6 gives accuracy and latency for the four non-linearities, evaluated directly on ciphertexts. First, encryption adds almost no error on top of the minimax fit: for the exponential, the encrypted error of 1.72×10^{-3} exceeds the minimax error of 1.60×10^{-3} by only 1.2×10^{-4}. Accuracy at these depths is therefore limited by the polynomial fit rather than by CKKS noise; the larger error for GELU (7.98×10^{-2}) reflects the difficulty of capturing its kink with a degree-8 polynomial. Second, the iterative reciprocal remains the most expensive operator at 219.16 ms, and the two iterative operators together (424.99 ms) cost more than the two polynomial activations (340.30 ms), consistent with their greater multiplicative depth.

Table 6. Accuracy and latency of encrypted non-linearity evaluation (E2).

Function	Interval	Approximation	Minimax error	Encrypted error	Latency (ms)
exp	[−8, 0]	degree-6 minimax	1.60×10^{-3}	1.72×10^{-3}	128.13 ± 1.49
GELU	[−6, 6]	degree-8 minimax	7.98×10^{-2}	7.98×10^{-2}	212.17 ± 2.43
1/x (reciprocal)	[1, 8]	Goldschmidt, r = 2	—	2.64×10^{-3}	219.16 ± 0.97
1/√x (inverse square root)	[0.3, 3]	Newton, s = 1	—	6.74×10^{-3}	205.83 ± 1.03

4.6 Three artefacts of the library rather than of the mathematics

A dedicated ablation isolates the implementation effect described in Section 3.1, using an interleaved paired design: the two forms of the Goldschmidt update alternate within the session, so that a disturbance striking one member of a pair strikes the other as well, and the estimator is the mean of the per-pair ratios rather than the ratio of the means. This is what allows the result to stand despite the contention documented in Section 4.1.

Across fifteen pairs the scalar-subtraction form averages 8007.6 ms and the algebraically identical negation-plus-addition form 279.7 ms. The mean paired ratio is 30.75 with a 95 % confidence interval of ±4.18 and a median of 33.69; the repeated session gives 29.34 ± 3.36, a difference of 4.6 % between sessions. We quote the effect as a factor of approximately 29 to 31. Three of the

fifteen pairs return ratios near 16.5 while the remaining twelve lie between 30 and 37; these three coincide with interference intervals and are the reason the confidence interval is as wide as it is. The interval excludes unity by an enormous margin in both sessions, so the finding is not in question; only its second significant figure awaits a quiet host. Fig. 4 shows both views of the ablation: the mean latency of the two forms on the left and the distribution of per-pair ratios on the right.

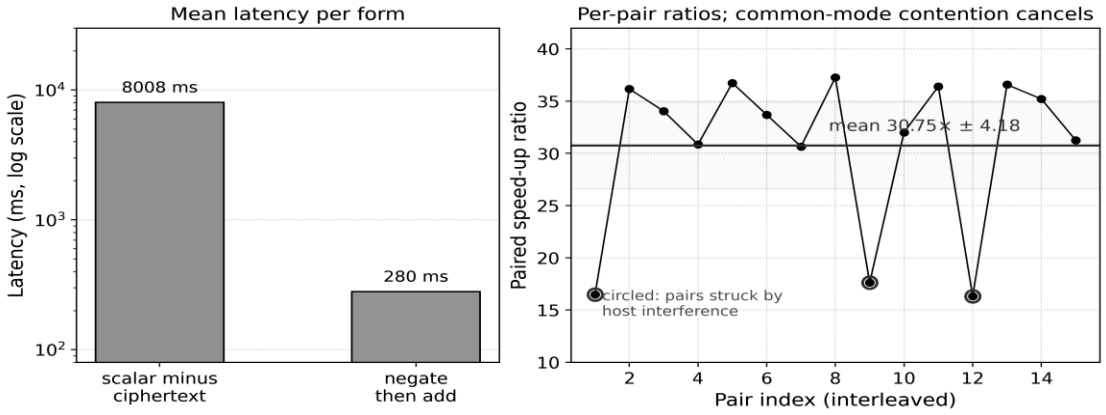

Fig. 4. The reciprocal bottleneck is an artefact of the scalar-subtraction implementation. Left: mean latency of the two forms. Right: per-pair ratios under the interleaved design.

Two further effects of the same class emerged during this work. The first is that the vector-packing helper consumes one multiplicative level: it is implemented as a multiplication by a selection mask, and nothing in its interface indicates that it draws on the depth budget. With a degree-six exponential, packing each projected token into a single ciphertext raises the attention block from six levels to seven, and the block then aborts at value aggregation. The second is that the ciphertext copy costs several times the reduction itself — 4.86× in the first session and 5.77× in the second. This ratio is the least stable of the three, because unlike the ablation it was not measured under an interleaved paired design. A harness that writes a copy immediately before a reduction inside the timed region will attribute almost five sixths of the measured time to the reduction; the reduction is non-mutating, so the copy is unnecessary. The common thread is that in a high-level homomorphic-encryption library the boundary between an algebraic operation and a data-movement operation is not visible from the interface. Depth and latency are both spent by the library in ways the mathematics does not predict. Reporting the level trace and the operation counts, rather than the wall-clock total alone, is what makes these effects detectable.

4.7 The depth wall, measured

Table 7 traces the multiplicative depth of a complete softmax, stage by stage, with the number of primes remaining recorded after each operation. The trace terminates: the second Goldschmidt iteration cannot begin, and the attempt raises a ciphertext scale-overflow exception with a single prime left. This is the depth wall encountered in practice rather than argued from a level count.

Table 7. Measured level-by-level trace of a complete softmax under chain C (budget seven levels).

Operation	Primes left	Levels used	Status
Encrypt scores	8	—	—
Exponential, degree-6 minimax	5	3	within budget
Denominator, rotate-and-sum reduction	5	0	rotations consume no level
Reciprocal seed, degree-2 minimax	3	2	within budget
Goldschmidt iteration 1: d·y	2	1	within budget
Goldschmidt iteration 1: update	1	1	budget exhausted
Goldschmidt iteration 2	—	—	ABORTS: scale out of bounds
Normalisation by the reciprocal	—	—	unreachable

Counting the query–key product that precedes the exponential and the normalising multiplication that would follow, a complete softmax requires eleven multiplicative levels against the seven available. Two of these arise from the reciprocal seed: a degree-two minimax initialisation is needed for the Goldschmidt recurrence to reach the reported accuracy in two iterations, and its degree is part of the depth budget. A degree-one seed leaves a relative residual of 0.43, which after two iterations is 3.5×10^{-2} and visibly diverges on ciphertexts; the seed degree is therefore not a free parameter.

A scale-overflow exception is not by itself evidence of depth exhaustion: the same exception is raised when two operands are at mismatched scales, which is a different fault with a different remedy. What distinguishes the two is the level trace — Table 7 shows the prime count descending monotonically to one and the failure occurring exactly where the budget runs out. This is why we report the trace rather than the exception. Applying the same instrumentation to a complete encoder layer gives Table 8, totalling thirty-three levels. Relative to a count considering only the four operator types, three corrections push upwards: an encoder layer contains two normalisation stages, so the inverse square root and its associated variance and rescaling multiplications enter twice; the feed-forward matrices each consume a level; and the iterative operators carry the depth of their polynomial seeds.

Table 8. Measured level-by-level depth of one encoder layer.

Stage	Levels	Occurrences	Subtotal	Note
Attention block (scores, exp, value)	6	1	6	measured, Section 4.8
Output projection WO	1	1	1	ct × pt
LayerNorm: mean and centring	1	2	2	two normalisation stages
LayerNorm: variance	2	2	4	squaring and rescaling
LayerNorm: inverse square root	6	2	12	degree-2 seed plus Newton step
LayerNorm: rescaling multiplication	1	2	2	—
Feed-forward W1	1	1	1	omitted in a naive count
GELU, degree-8 minimax	4	1	4	—
Feed-forward W2	1	1	1	omitted in a naive count
Total	—	—	33	against a budget of 7

Thirty-three levels against seven implies at least four bootstrapping operations per encoder layer under this schedule. TenSEAL provides no bootstrapping, so that cost is not measured here and is not included in any figure we report; it is an omission we state rather than an effect we have quantified. This is also why Section 4.9 reports a stage-wise rather than an end-to-end layer measurement.

4.8 Fully encrypted attention, with instrumented operation counts

In this experiment the attention computation runs end to end on encrypted data under the client-assisted protocol: the server returns an encrypted numerator and denominator, and the client performs the division after decryption within its own trusted environment. Division is therefore never evaluated homomorphically, and no reciprocal is computed server-side.

The encrypted mechanism reproduces the double-precision plaintext reference to a maximum absolute error of 3.19×10^{-6} and a cosine similarity of 0.9999999993, at $n = 8$ and $d_k = 16$ with the block finishing on two remaining primes; the repeated session gives 4.59×10^{-6} and 0.9999999974. These fidelity figures are numerical properties of the computation rather than of the host, agree between sessions to better than one part in 10^8, and are reported as final.

The block was instrumented, so its operation counts are recorded rather than inferred, which allows the cost model to be tested rather than merely fitted. Table 9 reports them. The layout implemented contracts the head dimension component-wise: each projected component occupies its own ciphertext, so an attention score costs d_k ciphertext–ciphertext multiplications and no reduction, while each projection costs one plaintext multiplication and one reduction of width d per output component.

The comparison in Table 9 is informative in both directions. Evaluated with the host's own measured rotation cost, the model predicts 122.20–123.76 s against a measured 135.61–148.70 s, a residual of 10–20 % depending on the session. Part of that residual is explained: the model charges nothing for input encryption, relinearisation or rescaling, and it uses a single ciphertext-multiplication cost regardless of how many primes remain. The second is the larger effect and acts in the opposite direction, since a ciphertext deep in the circuit is smaller and cheaper to multiply than a fresh one, so a model calibrated at the top of the chain overcharges most operations in a deep circuit. We report the residual as its own quantity rather than absorbing it into a fitted coefficient, and identify a level-resolved primitive benchmark as the measurement that would close it. Two further conclusions follow. The counts confirm that the block is rotation-heavy in the sense that matters: 1536 rotations at the measured 40.27–41.28 ms account for 61.9–63.4 s, comparable to the 47.3 s spent on all 2048 ciphertext multiplications. In addition, the counts are what make Section 4.9 possible, because the same instrumentation applied to the remaining stages yields a budget rather than an estimate.

Table 9. Instrumented operation counts for the attention block, and the cost model evaluated on them (n = 8, d = dk = 16, single head).

Quantity	Value	Source
Ciphertext × plaintext multiplications	384	instrumented counter: 3 × n × dk
Ciphertext × ciphertext multiplications	2048	instrumented counter: 2 × n^2 × dk
Rotate-and-sum reductions	384	instrumented counter, width d = 16
Rotations	1536	384 reductions × $\lceil \log_2 16 \rceil$
Ciphertext additions	3456	instrumented counter
Exponential invocations	64	instrumented counter: n^2
Cost model with β = 17.13 ms (derived, original session)	86.66 s	Eq. (11) on the counts above
Cost model with β = 40.27–41.28 ms (measured, Table 3)	122.20–123.76 s	Eq. (11) on the counts above
Measured latency, two sessions	135.61–148.70 s	provisional; see Sections 4.1 and 4.1b
Maximum absolute error	3.19×10^{-6} / 4.59×10^{-6}	final
Cosine similarity	0.9999999993 / 0.9999999974	final

4.9 Encoder-layer cost under three packing layouts

A complete encoder layer cannot be executed within a single modulus chain at these parameters, as Section 4.7 establishes, and the library provides no bootstrapping. Two quantities are therefore reported and must not be conflated. The first is a stage-wise measurement, in which every stage is executed on its own fresh chain and timed; this is a real measurement of every component but not an end-to-end run, because the ciphertext is refreshed between stages at no charge. The second is an operation-level projection, obtained by applying the measured primitive costs to the operation counts of each layout. The stage-wise measurement for the implemented layout totals 561.9 s in the first session and 713.6 s in the second, an interval of 562–714 s. The two sessions agree on the ordering of the contributions but not their magnitude: the attention block contributes 148.3 s and 134.0 s, the first feed-forward projection 81.2 s and 80.1 s, and the feed-forward GELU 271.1 s and 443.7 s across its 512 invocations. The GELU row accounts for most of the discrepancy and is the single least stable measurement in this paper, unsurprisingly given that it is a per-unit cost multiplied by 512. The interval is reported rather than either endpoint, and it is provisional under the criterion of Section 4.1. Read against the depth statement, a layer requiring thirty-three levels and executed in nine independently refreshed pieces is not the same computation as a layer executed end to end, and the four implied bootstrapping operations are absent from the total.

Table 10 gives the operation-level projection for three layouts. Layout A is the one implemented and measured, in which each projected component occupies its own ciphertext and attention scores are contracted component-wise. Layout B packs each projected token into one ciphertext, so a score costs one reduction instead of d_k ciphertext multiplications. Layout C applies the diagonal method, packing all tokens into a single ciphertext so that one matrix–vector product serves the whole sequence. Only A is measured; B and C are projections from the same measured primitives.

The rotation row uses the derived β = 17.13 ms of the original submission so that the three layouts remain comparable with that version. Substituting the directly measured β of the two sessions reported here, 40.27 ms and 41.28 ms, raises the three totals to 384.3–389.3 s, 242.7–248.0 s and 11.5–11.7 s respectively, without changing their ordering or their ratios other than through the rotation term. Layouts B and C are projections and are labelled as such throughout; only layout A has been executed.

Three conclusions follow from Table 10, and the first supersedes a claim made in the original submission. The span between the implemented layout and diagonal packing is a factor of 38, not the factor of 8 previously reported; the earlier figure counted only the amortisation of one matrix–vector product across tokens and omitted the algorithmic difference between the two schemes. Separating these matters because they have different evidential status. The algorithmic component has now been measured directly: a single 16 × 16 encrypted matrix–vector product takes 2726.9 ms under the per-token scheme and 1363.0 ms under the diagonal method, a ratio of 2.00, with the repeated session giving 1.98. The two agree to 1 %, which is what the paired construction is for; the underlying absolute latencies do not agree and are not quoted. The amortisation component, by contrast, is a count rather than a measurement: under layout C one product serves all n tokens. A single headline factor merging a measured ratio with a counted one is not defensible, and we report the two separately. Second, the dominant cost under the implemented layout is not rotation but the sheer multiplicity of non-linear invocations: 120.1 s of the 268.7 s projected total, of which the feed-forward

GELU alone accounts for 108.6 s across 512 invocations. This is a consequence of the layout rather than of the activation function, since packing the hidden layer reduces those 512 invocations to eight. Rotations remain the largest single contribution under layout B, at 89.9 s of 121.3 s. Third, the projected totals are one to two orders of magnitude above a layer latency obtained by counting only a handful of projection multiplications. The difference is not a change of measurement but of accounting: the earlier figure omitted the rotation term entirely and charged each non-linear operator once per layer rather than once per data-carrying ciphertext. The values reported here are consistent with the instrumented attention measurement of Section 4.8 and with the minutes-scale latencies reported for full encoder models, whereas a sub-second layer latency is not.

Table 10. Operation counts and projected compute-only latency of one encoder layer under three layouts (n = 8, d = dk = 16, dff = 64, single head, client-assisted protocol).

Quantity	A: implemented	B: packed scores	C: diagonal	Source
Ciphertext × plaintext multiplications	1184	1184	148	Eq. (9)
Ciphertext × ciphertext multiplications	2144	224	28	Section 3.3
Rotate-and-sum reductions	1184	1248	12	Section 3.3
Rotations	4992	5248	192	Eq. (10)
Ciphertext additions	6976	5376	208	Section 3.3
Exponential invocations	64	64	8	n^2 or n
Inverse-square-root invocations	16	16	2	2 per token per layer
GELU invocations	512	8	1	per hidden unit or per layer
Arithmetic latency (s)	60.46	16.12	2.02	counts × Table 2
Rotation latency (s)	85.51	89.89	3.29	counts × β = 17.13 ms
Addition latency (s)	2.65	2.04	0.08	counts × τadd
Non-linearity latency (s)	120.12	13.19	1.65	Eq. (12)
Projected total (s)	268.74	121.25	7.03	Eq. (11)
Stage-wise measured total (s)	561.9–713.6	—	—	provisional, two sessions
Multiplicative depth (levels)	33	33	33	Table 7

4.10 Hybrid-partition sensitivity (E5)

From the measured operator costs, the pure-CKKS non-linear block of an encoder layer comes to 971.12 ms when counted per invocation under the four-operator definition of Eq. (12) — one exponential, one reciprocal, two inverse square roots and one GELU. Under the layout actually implemented the multiplicities are far larger, as Table 10 shows, and the corresponding block cost is 120.1 s; the four-operator figure is retained so that the partition analysis remains comparable with the submitted version and with the literature, but it should be read as a per-operator-type quantity rather than a per-layer one. Table 11 projects the hybrid block latency of Eq. (14) over a sweep of assumed single-PBS costs. As long as the assumed PBS cost stays below the cheapest measured CKKS operator — the exponential at 128.13 ms — the model routes every invocation to PBS, and the projected block latency is the number of invocations times Tpbs: five times, not four. Pushing the sweep further reveals the transition structure of Eq. (14): at 129 ms the exponential returns to CKKS, at 206 ms the two inverse square roots follow, at 213 ms GELU does the same, and by 220 ms every invocation is cheaper in CKKS, so the projection settles at the measured pure-CKKS cost. The break-even points are therefore nothing other than the measured operator costs — 128.13, 205.83, 212.17 and 219.16 ms — which turns the sweep into a decision chart: once a real TFHE benchmark exists, one need only place it on this axis to read off which invocations are worth offloading.

he five-invocation column corresponds to the per-layer operator definition of Eq. (12) and is the column to compare against any figure quoted for a single encoder layer; the twelve-invocation column corresponds to the invocation multiplicities of layout L2 in Table 9. A four-invocation accounting, which omits the second inverse square root, understates the offloaded workload by one bootstrap below the break-even and the pure-CKKS asymptote by 205.83 ms, and is not used here.

The honest reading is conditional. If a TFHE programmable bootstrap can be executed in tens of milliseconds, offloading the non-linearities would reduce the five-invocation block by a factor of 2.43 at an assumed 80 ms per bootstrap, and by 19.4 at 10 ms. All of this hinges on a PBS cost we did not measure, and no real-time claim rests on it. Eq. (14) also charges nothing for the scheme switch itself; a genuine CKKS-to-TFHE conversion cost would shift every break-even point to the right.

Table 11. Projected hybrid CKKS/TFHE non-linear block latency against assumed single-PBS cost (E5), with operator multiplicity.

Assumed Tpbs (ms)	Projected block, 5 invocations (ms)	Invocations to PBS	Projected block, 12 invocations, L2 (ms)	Invocations to PBS
10	50.00	5	120.00	12
80	400.00	5	960.00	12
120	600.00	5	1440.00	12
129	644.13	4	1541.04	4
206	951.79	2	1848.70	2
213	964.96	1	1861.87	1
220	971.12	0 (pure CKKS)	1868.03	0 (pure CKKS)
300	971.12	0 (pure CKKS)	1868.03	0 (pure CKKS)

4.11 Head-to-head comparison with recent methods

Direct comparison with recent privacy-preserving Transformer systems is complicated by the fact that reported latencies refer to different models, sequence lengths, layer counts, hardware and threat models, and that most systems are interactive whereas the present framework is not. A comparison is nonetheless made on a normalised basis: for a system reporting an end-to-end latency Tmodel for a model of ℓ encoder layers, the per-layer latency is taken as Tmodel/ℓ, and the sequence-length dependence is stated explicitly rather than rescaled, since the attention term of Eq. (10) is quadratic in n under L1 and linear under L2. Table 12 sets out the comparison.

Table 12. Head-to-head comparison on a per-encoder-layer basis with methods published after 2024.

Method	Year	Paradigm	Interaction	Reported end-to-end latency	Per-layer latency	Hardware
Nimbus [33]	2024	Hybrid HE / 2PC	Yes	≈29.3 / ≈102.2 s (LAN/WAN)	2.44 / 8.52 s (measured per block)	2 nodes, 64 vCPU each
BOLT [27]	2024	Hybrid HE / 2PC	Yes	91–1744 s (Table 1)	7.6–145.3 s	2×AWS c6i.16xlarge, 32 threads
BumbleBee [28]	2025	Hybrid HE / 2PC	Yes	153.0 / 291.6 s (LAN/WAN)	12.75 / 24.30 s	2×Alibaba ecs.g7.16xlarge, 64 vCPU
NEXUS [34]	2025	FHE-native, RNS-CKKS	No	Minutes per inference	≈71.4 s (CPU) / ≈3.11 s (GPU)	2×Xeon 3.70 GHz, 32 thr.; 4×A100
THOR [35]	2025	FHE-native	No	≈ 10 min per inference	≈52.2 s (derived)	GPU
Powerformer [36]	2025	FHE-native	No	344.52 s (RTE, incl. bootstrap)	≈28.7 s	1×A100 (Liberate.FHE)
SHAFT [32]	2025	MPC	Yes	28.60 / 66.46 s (LAN/WAN)	2.38 / 5.54 s	Xeon Gold 5318Y + 2×A40
CipherPrune [31]	2025	Hybrid HE / MPC	Yes	79.1 s (LAN)	≈6.59 s (not uniform; pruning)	Threadripper PRO 3955WX
Proposed Method, layout A (implemented)	2026	FHE-native, RNS-CKKS	No	n/a (single layer)	268.7 s proj.; 561.9 s stage-wise	2 vCPU, single thread
Proposed Method, layout C (diagonal)	2026	FHE-native, RNS-CKKS	No	n/a (single layer)	7.0 s projected	2 vCPU, single thread

Two points can be made without waiting for those figures. First, the ordering is consistent: non-interactive FHE-native systems report latencies in the range of minutes for BERT-base [34], [35], and an operation-level projection of 7.03 s per layer at d = 16

scales into that range for a twelve-layer model at d = 768, whereas a sub-second per-layer figure does not. The cost model is therefore consistent with the published state of the art — a weaker but more defensible claim than superiority. Second, the contribution is orthogonal to those systems rather than competitive with them: BOLT, BumbleBee, NEXUS, THOR and Powerformer each report an aggregate latency for a bespoke pipeline, and none reports the per-primitive decomposition that would allow their gains to be attributed to packing, approximation degree or scheme choice.

4.12 Discussion

Taken together the experiments tell one connected story, and it is a story about accounting as much as about cryptography. The chain and depth accounting behind the depth wall (Section 5, first and second findings) is measured by execution rather than assumed, which is what lets the rest of this section treat it as settled. The correctness of encrypted attention is established to a cosine similarity of 0.9999999993, and its operation counts are recorded rather than estimated, which converts the cost model from a fitted description into a testable one — it predicts 122.2–123.8 s against a measured 135.6–148.7 s, and the 10–20 % residual is itself informative about what the model omits. The three library artefacts are, in a sense, the most portable result. None is a property of RNS-CKKS; all three are properties of a particular interface, and all three are invisible from that interface. A scalar subtraction that costs thirty times its algebraic equivalent, a packing helper that quietly spends a multiplicative level, and a copy that costs nearly five times the reduction beside it are the kind of effects that turn a published latency into an unreproducible one. That we found three in a single library, having looked for one, suggests they are not rare.

The framework accordingly claims no real-time inference; at three to four orders of magnitude from the sub-100 ms target, the distance is not one that packing alone closes. What it contributes is the conversion of a vague sense that FHE Transformers are slow into an itemised budget in which each entry carries its own evidential status — measured, projected, or absent — together with the two levers, packing against the rotation and invocation counts and partitioning against the depth cost. A word is owed on the reporting standard adopted here. Both sessions failed their own contention criterion, and we have said so rather than repeating the measurement quietly until it passed. Doing so costs something: the absolute latencies in Sections 4.8 and 4.9 are provisional and await a dedicated host. It also buys something, as the fifth finding of Section 5 details: the pre-registered classification held under an independent repetition, and it caught an error a single session would have concealed. A measurement discipline that changes a conclusion is doing its job.

5. Conclusion

This paper has described a measurement-first, operation-level framework for the cost of fully homomorphic Transformer inference under RNS-CKKS. Rather than adding one more bespoke pipeline with a single headline latency, we decomposed an encoder layer into its cryptographic primitives and non-linearities, instrumented the code so that operation counts are recorded rather than estimated, and assembled a transparent per-layer budget in which every entry is labelled as measured, projected or omitted. Four findings should outlive the particular numbers reported here. The first is the chain accounting: a chain of L primes yields L − 2 usable multiplicative levels, established by executing sequential multiplications until failure, and the parameter set sits inside the 128-bit region by the library's own validation, which accepts 400 bits and refuses 440 against a reported ceiling of 438. The second is the depth accounting: a complete softmax requires eleven levels and a complete encoder layer thirty-three, so a layer implies at least four bootstrapping operations, and the softmax attempt aborts at exactly the operation the level trace identifies. The third is that ciphertext packing, not cryptographic parameter choice, is the dominant lever: the layout implemented here is a factor of 38 more expensive than diagonal packing, of which a factor of about 2.0, reproduced to 1 % across two sessions, is a directly measured algorithmic difference and the remainder is amortisation across tokens. The fourth is that a substantial part of the cost belongs to the library rather than to the mathematics — a reciprocal artefact of roughly thirtyfold, a packing helper that consumes a multiplicative level, and a copy operation that costs several times the reduction beside it. A fifth finding is methodological and emerged only from repetition. Running the complete measurement set twice on the same shared platform separates the results cleanly by class: the deterministic quantities are identical, the interleaved paired ratios agree to within 4.6 %, and the absolute latencies differ by up to 27 %. The repetition also overturned a claim a single session would have supported, namely that the derived rotation estimate is close to the directly measured one; it agreed to 1.0 % in one session and differed by 20.0 % in the other, because the quotient inherits the variability of everything the reduction contains besides rotation. Absolute latencies are accordingly reported as intervals spanned by the two sessions. The value of the work lies less in any individual number than in the methodology: because every estimate is anchored to a measured primitive and an instrumented operation count, the framework gives packing, batching, partitioning and bootstrapping optimisations a common yardstick, and a reader can tell which of its numbers are ready to be relied upon. Several limitations qualify these conclusions. Both sessions failed their own contention criterion, at coefficients of variation of 38.7 % and 24.4 %; every absolute latency in Sections 4.8 and 4.9 is therefore provisional, is quoted as an interval, and requires confirmation on a dedicated host, and the primitive costs of Table 2 were not re-measured for the same reason. Two rotation offsets were excluded from Table 3 as contaminated; the exclusion is supported by the second session but not settled by it. The layer figure is a stage-wise total in which each stage begins from a fresh ciphertext, not an end-to-end measurement, and the four implied bootstrapping operations are absent from it; layouts B and C in Table 9 are projections.

The cost model charges a single ciphertext-multiplication cost regardless of remaining level, which overcharges deep operations. The 128-bit figure is a nominal RLWE security level only: the noise flooding demanded by the security analysis of approximate homomorphic encryption is not implemented. Finally, the experiments cover one head at small dimensions on synthetic tensors, with no downstream accuracy reported, and single-threaded CPU measurements say nothing about GPU acceleration. The budget points at what to do next, in order of expected value. The most immediate step is to repeat the full measurement set on a dedicated node, converting every provisional interval into a point estimate. Second, a level-resolved primitive benchmark would replace the single ciphertext-multiplication cost with a per-level one and close the residual in Table 8. Third, implementing diagonal packing would convert the layout C column of Table 9 into a measurement. Fourth, a bootstrapping-capable backend such as OpenFHE [44] would allow an end-to-end layer to be executed and replace the projected PBS cost with a measured one. Fifth, architectural co-design could remove the depth wall at its source. Finally, extending the harness to multi-head, multi-layer models would tie the cryptographic budget to task performance.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
ORCID iD: 0000-0003-0368-5490
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] L. de Castro, A. Polychroniadou, and D. Escudero, 2024, "*Privacy-preserving large language model inference via GPU-accelerated fully homomorphic encryption",* 38th Conference on Neural Information Processing Systems (NeurIPS).

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 2019, "*BERT: Pre-training of deep bidirectional Transformers for language understanding"*, In Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, pp. 4171–4186.

[3] T. B. Brown et al., 2020, *"Language models are few-shot learners*", Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901.

[4] S. Wada et al., 2024, "*Oversampling effect in pretraining for bidirectional encoder representations from transformers (BERT) to localize medical BERT and enhance biomedical BERT*", Artificial Intelligence in Medicine, vol. 153, Art. 102889.

[5] J. H. Cheon, A. Kim, M. Kim, and Y. Song, 2017, "*Homomorphic encryption for arithmetic of approximate numbers*", Advances in Cryptology — ASIACRYPT 2017, Springer, pp. 409–437.

[6] R. Geelen and F. Vercauteren, "*Bootstrapping for BGV and BFV revisited*", Journal of Cryptology.

[7] S. Fan et al.,2024, "*Taiyi: A high-performance CKKS accelerator for practical fully homomorphic encryption",* arXiv preprint arXiv:2403.10188.

[8] B. Li and D. Micciancio, 2021, "*On the security of homomorphic encryption on approximate numbers*", Advances in Cryptology — EUROCRYPT, Springer, pp. 648–677.

[9] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, 2016, *"CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy*", Proc. 33rd Int. Conf. on Machine Learning (ICML), pp. 201–210.

[10] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, 2022, "Iron*: Private inference on transformers"*, Advances in Neural Information Processing Systems (NeurIPS), vol. 35, pp. 15718–15731.

[11] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski, 2019, *"nGraph-HE2: A high-throughput framework for neural network inference on encrypted data"*, In Proc. 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 45–56.

[12] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, 2019, "*nGraph-HE: A graph compiler for deep learning on homomorphically encrypted data"*, In Proc. 16th ACM Int. Conf. Computing Frontiers, pp. 3–13.

[13] K. A. Ameen, A. A. Majeed et al., 2026, "*Cryptosystem using RNA combined with data sequence and affine cipher for secure data communication*", Journal of Theoretical and Applied Information Technology, Vol. 104.

[14] B. Li and D. Micciancio, 2021, "*On the security of homomorphic encryption on approximate numbers*", Advances in Cryptology — EUROCRYPT, Springer, pp. 648–677.

[15] J. Lee et al., 2022, "*Privacy-preserving machine learning with fully homomorphic encryption for deep neural network*", IEEE Access, vol. 10, pp. 30039–30054.

[16] M. Baruch, N. Drucker, L. Greenberg, and G. Moshkowich, 2022, "*A methodology for training homomorphic encryption friendly neural networks"*, Applied Cryptography and Network Security Workshops (ACNS 2022), Lecture Notes in Computer Science, vol. 13285, Springer, pp. 536–553.

[17] Microsoft Research, *"Microsoft SEAL (release 4.1)",* Redmond, WA, USA, 2023. [Online]. Available: https://github.com/microsoft/SEAL.

[18] F. Boemer, S. Kim, G. Seifu, F. D. M. de Souza, and V. Gopal, 2021, *"Intel HEXL: Accelerating homomorphic encryption with Intel AVX512-IFMA52",* In Proc. 9th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC), ACM, pp. 57–62, 2021.

[19] K. Garimella, A. Ebel, G. De Micheli, and B. Reagen, 2025, "*HE-LRM: Efficient private embedding lookups for neural inference using fully homomorphic encryption*", arXiv preprint arXiv:2506.18150.

[20] X. Jiao et al., 2020, *"TinyBERT: Distilling BERT for natural language understanding*", In Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 4163–4174.

[21] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, 2020 *"MobileBERT: A compact task-agnostic BERT for resource-limited devices*", In Proc. 58th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 2158–2170.

[22] E. Frantar and D. Alistarh, 2022, *"Optimal brain compression: A framework for accurate post-training quantization and pruning*", Advances in Neural Information Processing Systems (NeurIPS), Vol. 35, pp. 4475–4488.

[23] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, 2022, "*FlashAttention: Fast and memory-efficient exact attention with IO-awareness*", Advances in Neural Information Processing Systems (NeurIPS), vol. 35, pp. 16344–16359.

[24] T. Dao, 2024, *"FlashAttention-2: Faster attention with better parallelism and work partitioning*", International Conference on Learning Representations, pp. 35549–35562.

[25] T. Chen et al., 2022, "*THE-X: Privacy-preserving transformer inference with homomorphic encryption*", In Findings of the Association for Computational Linguistics: ACL 2022, pp. 3510–3520.

[26] G. Lee, M. Kim, J. H. Park, S.-W. Hwang, and J. H. Cheon, 2022, "*Privacy-preserving text classification on BERT embeddings with homomorphic encryption*", In Proc. 2022 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pp. 3169–3175.

[27] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider, 2024,*"BOLT: Privacy-preserving, accurate and efficient inference for transformers"*, In Proc. 2024 IEEE Symp. Security and Privacy (SP), pp. 4753–4771.

[28] W.-J. Lu et al., 2025, *"BumbleBee: Secure two-party inference framework for large transformers*", Network and Distributed System Security Symposium (NDSS).

[29] X. Hou et al., 2026, *"CipherGPT: Secure two-party GPT inference*", IEEE Transactions on Dependable and Secure Computing, vol. 23.

[30] J. Luo et al., 2024, "*SecFormer: Fast and accurate privacy-preserving inference for transformer models via SMPC",* In Findings of the Association for Computational Linguistics: ACL 2024, pp. 13333–13348.

[31] Y. Zhang et al., 2025, *"CipherPrune: Efficient and scalable private transformer inference*", In Proc. 13th Int. Conf. Learning Representations (ICLR), 2025.

[32] A. Y. L. Kei and S. S. M. Chow, 2025, *"SHAFT: Secure, handy, accurate, and fast transformer inference*", In Proc. Network and Distributed System Security Symposium (NDSS).

[33] Z. Li, K. Yang et al., 2024, "*Nimbus: Secure and efficient two-party inference for transformers*", Advances in Neural Information Processing Systems, Vol. 37, pp. 21572–21600.

[34] J. Zhang et al., 2025, *"Secure transformer inference made non-interactive*", In Proc. Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA.

[35] J. Moon, D. Yoo, X. Jiang, and M. Kim, 2025, "*THOR: Secure transformer inference with homomorphic encryption*", In Proc. 2025 ACM SIGSAC Conference on Computer and Communications Security, pp. 3765–3779.

[36] D. Park, E. Lee, and J.-W. Lee, 2025, "*Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption*", In Proc. 63rd Annual Meeting of the Association for Computational Linguistics (ACL), vol. 1, pp. 11090–11111.

[37] I. Zimerman, M. Baruch, N. Drucker, G. Ezov, O. Soceanu, and L. Wolf, 2024, *"Converting transformers to polynomial form for secure inference over homomorphic encryption*", In Proc. 41st Int. Conf. Machine Learning (ICML), PMLR vol. 235, pp. 62803–62814.

[38] I. Zimerman et al., 2024, *"Power-Softmax: Towards secure LLM inference over encrypted data*", arXiv preprint arXiv:2410.09457.

[39] H. Qu and G. Xu, 2023, *"Improvements of homomorphic secure evaluation of inverse square root*", In Proc. 25th Int. Conf. Information and Communications Security (ICICS), Springer, pp. 110–127.

[40] L. Rovida and A. Leporati, 2024, *"Transformer-based language models and homomorphic encryption: An intersection with BERT-Tiny*", In Proc. 10th ACM International Workshop on Security and Privacy Analytics, pp. 3–13, 2024.

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

[42] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal,2021, "*TenSEAL: A library for encrypted tensor operations using homomorphic encryption*", arXiv preprint arXiv:2104.03152.

[43] I. Chillotti, M. Joye, and P. Paillier,2021, "*Programmable bootstrapping enables efficient homomorphic inference of deep neural networks",* International Symposium on Cyber Security Cryptography and Machine Learning (CSCML), Springer, pp. 1–19.

[44] A. Al Badawi, J. Bates, F. Bergamaschi et al., 2022 "*OpenFHE: Open-source fully homomorphic encryption library*", In Proc. 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC), ACM, pp. 53–63.

Nomenclature

All abbreviations are expanded at first use in the text; they are collected here for reference.

Acronym	Expansion	Acronym	Expansion
BFV	Brakerski–Fan–Vercauteren scheme	MPC	Multi-Party Computation
BGV	Brakerski–Gentry–Vaikuntanathan scheme	NTT	Number-Theoretic Transform
BSGS	Baby-Step Giant-Step	OT	Oblivious Transfer
CKKS	Cheon–Kim–Kim–Song scheme	PBS	Programmable Bootstrapping
FHE	Fully Homomorphic Encryption	RLWE	Ring Learning With Errors
FFN	Feed-Forward Network	RNS	Residue Number System
GELU	Gaussian Error Linear Unit	SEAL	Simple Encrypted Arithmetic Library
GDPR	General Data Protection Regulation	SIMD	Single Instruction, Multiple Data
HE	Homomorphic Encryption	SMPC	Secure Multi-Party Computation
HIPAA	Health Insurance Portability and Accountability Act	TFHE	Fast Fully Homomorphic Encryption over the Torus
LN	Layer Normalisation	2PC	Two-Party Computation

Symbols: N polynomial modulus degree; Δ scaling factor; $S = N/2$ available SIMD slots; L number of primes in the coefficient-modulus chain; α measured ciphertext–ciphertext multiplication latency; β calibrated single-rotation latency; τ_{pt} ciphertext–plaintext multiplication latency; τ_{add} ciphertext addition latency; c_k measured CKKS latency of non-linear operator k; m_k number of invocations of operator k per encoder layer; T_{pbs} assumed latency of one programmable bootstrap.