OBELISK: CO-DESIGNING POST-PACKING EXECUTION FOR EFFICIENT TRANSFORMER INFERENCE UNDER FULLY HOMOMORPHIC ENCRYPTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Privacy-preserving Transformer inference under CKKS fully homomorphic encryption remains expensive in evaluation time and memory. Prior systems mainly optimize packing layouts and polynomial approximations of nonlinear operators. Packing fixes the logical dataflow, namely which values meet in which operations, but not how that dataflow is executed. Three post-packing decisions remain: the level budget, intermediate-ciphertext lifetime, and slot representation. To our knowledge, OBELISK is the first system to co-design all three for non-interactive secure Transformer inference on a commodity GPU, through budget–operator co-design, Process-in-Pieces (PiP), and Hybrid Real-Complex Representation (HRC). Budget–operator co-design shortens the nonlinear operators and lets a layer-level cost model choose the budget, cutting the layer's full-width bootstraps to two and reinvesting the freed modulus bits in cheaper key switching. PiP processes attention by head and the feed-forward network by chunk, avoiding full-width resident intermediates without a measured increase in evaluation time. HRC screens each operation through three criteria and runs it in the cheaper representation, real or complex, halving the work of the two full-width bootstraps and both ciphertext–ciphertext matmuls without adding a level. On fully encrypted 12-layer GLUE inference, OBELISK runs on a commodity 24 GiB GPU (an RTX 4090) that no evaluated baseline fits, and on the same H200 as the state of the art it reduces end-to-end evaluation time by 6.6× and peak memory by 6.0× (129.3 to 21.6 GiB), within 2.52 pp of plaintext accuracy. We report directly measured peak memory for all compared systems, which no evaluated prior system reports.

1 INTRODUCTION

Machine learning increasingly processes sensitive medical, financial, and communication data that cannot be exposed in plaintext, motivating privacy-preserving machine learning (PPML) (Al-Rubaie & Chang, 2019). Fully homomorphic encryption (FHE) such as CKKS (Cheon et al., 2017) evaluates models directly on ciphertexts. Each homomorphic multiplication consumes one level from a finite modulus chain; bootstrapping restores the chain but is the scheme's most expensive operation (Gentry, 2009; Cheon et al., 2018). FHE inference has progressed from shallow convolutional networks at fixed multiplicative depth (Gilad-Bachrach et al., 2016), through deep residual networks enabled by bootstrapping (Lee et al., 2022a;b), to encoder-based Transformers (Zhang et al., 2025; Moon et al., 2025; Park et al., 2025; Zhang et al., 2026; Gao et al., 2026).

Figure 1 illustrates why Transformer inference under FHE remains far from real time and requires server-class memory. Softmax, GELU, and LayerNorm lack exact homomorphic forms and therefore use polynomial approximations that consume much of each layer's level budget. Exhausting this budget triggers bootstrapping several times per layer, the dominant expense. Bootstrapping, homomorphic rotations, and ciphertext–ciphertext multiplications all invoke key switching, so evaluation time depends largely on the key-switch count and the per-switch cost, which grows with the level each switch runs at. Evaluation keys and full-width resident intermediates dominate memory; with each ciphertext occupying tens of megabytes, the peak can exceed 100 GiB. Prior systems primarily optimize packing layouts, matmuls, and polynomial approximations, but not the execution decisions that remain after packing, where much of that time and memory is decided.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

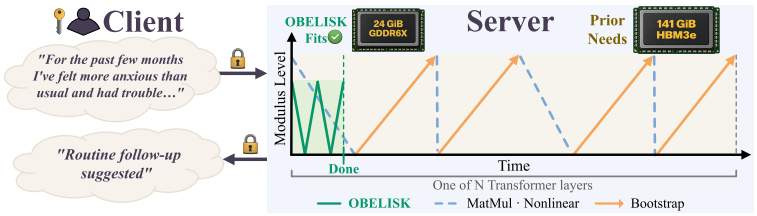


Figure 1: Privacy-preserving Transformer inference under FHE. Consuming fewer modulus levels, OBELISK (green) bootstraps less, finishes sooner, and fits a commodity 24 GiB GPU where MOAI, the state of the art, needs a server-class 141 GiB one.

After packing, three decisions remain open. These post-packing decisions govern encrypted Transformer inference's evaluation time and memory: **Budget**—how should a fixed modulus be split among compute levels, bootstrapping levels, and the special modulus, together with the nonlinear operators that spend it? **Lifetime**—how long and at what level should each intermediate ciphertext stay resident? **Representation**—should computation run in real or complex slots, and at what cost in operations? Prior systems set them in isolation, inheriting library budgets, retaining full-width intermediates, and fixing a real- or complex-native representation.

None of the three can be set independently, since each interacts with the operators and the execution schedule. The level budget is capped by 128-bit classical security (Albrecht et al., 2015), and allocating modulus bits between compute levels and the special modulus trades operator depth against key-switching cost. Reducing the compute levels therefore requires shallower GELU and LayerNorm approximations that remain stable across the full stack. Ciphertext lifetime determines peak memory, since residuals wait at a low level while attention and feed-forward intermediates stay full-width and high-level until consumed. A real-native representation leaves the imaginary lane idle, whereas a complex-native one uses both but must return to real before every nonlinear operator.

Our contributions can be summarized as follows.

- We introduce budget–operator co-design, which shortens the nonlinear operators and selects the budget with a layer-level cost model, cutting full-width bootstraps from four to two and reallocating the freed modulus bits to cheaper key switching.
- We introduce Process-in-Pieces (PiP), which evaluates attention by head and the FFN by chunk so that no full-width, high-level intermediate remains resident. At $K{=}128$, PiP fits every evaluated GPU (24–141 GiB) without per-device retuning, while the untiled 65.8 GiB intermediate fits only the evaluated server-class cards.
- We introduce Hybrid Real-Complex Representation (HRC), which screens each operation with three criteria and runs it in the cheaper representation, real or complex, halving the two full-width bootstraps and both ciphertext–ciphertext matmuls at no level cost, a measured $1.6\times$ end-to-end speedup. The pairing itself is standard (Bossuat et al., 2021); what is new is the screen that decides where it pays.

2 Related Work

Secure Transformer inference. Secure Transformer inference is either interactive or non-interactive. Interactive systems use secure multi-party computation, alone or with homomorphic encryption (Hao et al., 2022; Pang et al., 2024; Lu et al., 2025; Dong et al., 2025), or offload hard-to-approximate operations to the client, as in THE-X (Chen et al., 2022). Communication is their primary overhead. Non-interactive systems evaluate the entire model under HE without client participation, shifting the cost to server computation. Converting every nonlinearity to polynomial form first enabled non-interactive HE Transformer inference (Zimerman et al., 2024), followed by NEXUS (Zhang et al., 2025), THOR (Moon et al., 2025), Powerformer (Park et al., 2025), MOAI (Zhang et al., 2026), Euston (Gao et al., 2026), and EncryptedLLM (De Castro et al., 2025) on GPT-2, all over fixed packing layouts. OBELISK adopts this setting but optimizes post-packing execution rather than the packing layout.

Post-packing decisions. For the level budget, THOR and Powerformer use Liberate.FHE (DESILO, 2023), while MOAI and NEXUS both build on Microsoft SEAL (Microsoft Research, 2023) with bootstrapping from FHE-MP-CNN (Lee et al., 2022a) and Phantom (Yang et al., 2024) on the GPU; each fixes the budget before optimizing nonlinear operators. AutoFHE (Ao & Boddeti, 2024) jointly tunes polynomial degrees and bootstrap placement for convolutional networks, as do scale- and bootstrapping-management compilers (Lee et al., 2023b; Cheon et al., 2024; 2025; Liu et al., 2025), but within a fixed budget. For ciphertext lifetime, prior HE Transformer systems retain full-width intermediates throughout each consuming block and do not report peak memory. Concurrent GPU work instead distributes the working set across GPUs (Gong et al., 2026; Jayashankar et al., 2025) or shrinks the switching-key set under a 32-bit-friendly modulus (Seo & Min, 2026); hierarchical rotation keys shrink the resident keys (Lee et al., 2023a). For the representation, THOR is complex-native, whereas MOAI and NEXUS are real-native. Prior complex-pairing techniques share a bootstrap (Bossuat et al., 2021; EPFL-LDS, Tune Insight SA, 2024), double plaintext–ciphertext throughput (Boemer et al., 2019), or reduce conversion payload at hybrid-protocol boundaries (Zhu et al., 2026). OBELISK addresses all three (Sections 3.1–3.3).

Classification and generation. For generation, Cachemir (Yu et al., 2026) decodes with an encrypted KV cache, and a concurrent multi-GPU implementation extends encrypted inference to LLaMA-3-8B prefill and decoding (Park et al., 2026), both at high per-token cost. OBELISK targets classification with BERT-base, and the three post-packing decisions carry over to a generation layout rather than compete with it, leaving decode-time generation outside this evaluation.

3 Post-Packing Execution Co-Design

We instantiate the three decisions on a column-packed layout, one ciphertext per hidden column (Zhang et al., 2026). Each makes its own demand of that layout (Appendix J). Levels are allocated to fit the tightened nonlinear operators (**budget**, Section 3.1), attention runs head by head and the FFN chunk by chunk (**lifetime**, Section 3.2), and the operations marked C run complex (**representation**, Section 3.3). The three constrain one another. The unsplittable FFN chunk raises the depth floor the budget must clear, the budget's zero-slack chain admits only level-free pairing, and that pairing shrinks the attention working set the lifetime decision minimizes.

3.1 Co-designing the Level Budget and Operators

The co-design problem. OBELISK designs the level budget and the nonlinear operators as one system, since neither can be fixed before the other. The co-design shrinks the layer's total operator depth D by redesigning the nonlinear operators and selects (B, L, α) for the smaller workload. A layer's ciphertext modulus Q is the chain of L levels a ciphertext descends between bootstraps. It holds one 51-bit base prime, $L-1$ compute primes of 46 bits, and 14 bootstrapping primes of 51 bits. The special modulus P holds α primes of 58 bits for key switching. For 128-bit classical security at $N = 2^{16}$, the library's security table caps the total modulus bits at 1777 (Yang et al., 2024; Albrecht et al., 2018):

$$51 + 46\,(L-1) + 51 \cdot 14 + 58\,\alpha \;\le\; 1777. \tag{1}$$

The prime bit-widths and the 1777-bit security cap come from the underlying library (Yang et al., 2024), the bootstrap depth from its bootstrapping routine (Lee et al., 2022a), and a different library merely re-instantiates them. Equation (1) constrains L and α but not the full-width bootstrap count B, which is instead constrained by coverage. Each bootstrap restores $L-1$ levels, and together the B bootstraps must cover the layer's consumption D, the sum of its operators' depths, so $(L-1)\,B \ge D$. Coverage and security therefore leave a one-parameter family of budgets indexed by the bootstrap count B, since each B fixes the smallest covering chain L and the largest α the remaining bits allow.

Schedulability. Two further requirements bind the schedule. The first is a floor on the chain length. A block a bootstrap cannot split, a single nonlinear operator or the width-expanded FFN, must fit a segment's $L-1$ levels, so $L-1 \ge \delta$ with δ the depth of the deepest such block. The second is a condition on held operands, values kept for later use such as a value projection or a residual. A held operand can only drop levels, so it must be generated at or above the level where it is used ($L_{\text{gen}} \ge L_{\text{use}}$), a condition the scheduler meets through operand placement rather than the chain length.

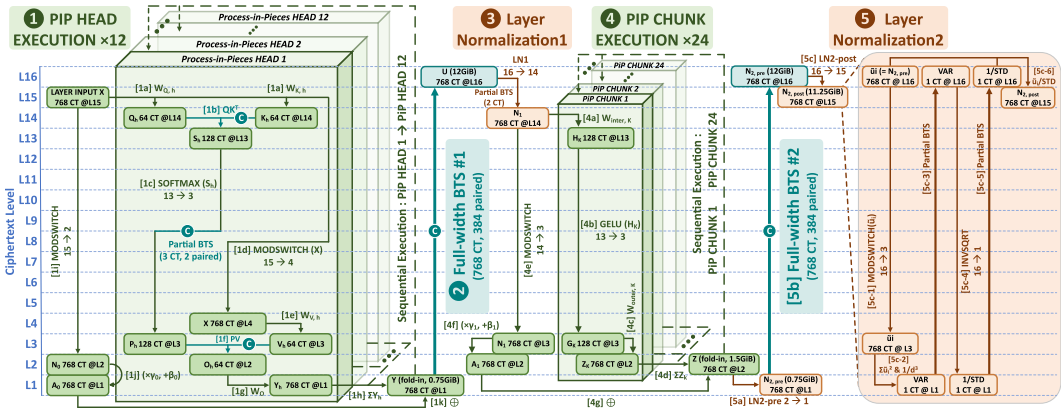


Figure 2: OBELISK's post-packing execution of one BERT-base encoder layer. Drawn at the deployed budget $(B, L, \alpha) = (2, 16, 5)$. Each box sits at the level of the ciphertexts it holds, and the operations marked C run in the complex representation. The five numbered stages run in order, and the lettered steps within each stage run alphabetically.

Cost-based budget selection. These requirements fix which budgets are feasible but not which is fastest. A layer's cost splits into its bootstraps and the work between them. Between bootstraps, the ciphertext–ciphertext matmuls and the polynomial nonlinearities are dominated by their key-switches. By contrast, the plaintext–ciphertext multiplications carry no key-switches, nor do the remaining products, accumulations, and rescales. Both prices follow from the operand's limbs. A ciphertext at level ℓ carries ℓ residue limbs, one per modulus prime it still holds, and each limb an operation touches costs one number-theoretic transform of $N \log N$ at ring dimension N. A key-switch-free operation touches each limb once, so it costs ℓ transforms. A key-switch instead decomposes the ciphertext into $\mathrm{dnum} = \lceil \ell/\alpha \rceil$ digits of α limbs, raises them to the extended $(\ell+\alpha)$-limb modulus, and reduces back. These $\mathrm{dnum} + 2$ passes over $\ell + \alpha$ limbs are the standard cost of hybrid key-switching (Han & Ki, 2020; Bossuat et al., 2021), so

$$t_{\mathrm{KS}}(\ell) \;\propto\; (\mathrm{dnum} + 2)(\ell + \alpha)\, N \log N. \tag{2}$$

A full-width bootstrap refreshes each ciphertext through a fixed circuit spanning the 14 limbs above the chain, and pricing its schedule of $n_{\mathrm{boot}}(\ell)$ key-switches and $m_{\mathrm{boot}}(\ell)$ key-switch-free operations the same way gives the cost of one bootstrap,

$$t_{\mathrm{BTS}} \;\propto\; \sum_{\ell} \left[\, n_{\mathrm{boot}}(\ell)\, t_{\mathrm{KS}}(\ell) \;+\; m_{\mathrm{boot}}(\ell)\, \ell \,\right]. \tag{3}$$

The second term of Eq. (3) is necessary because the key-switch share of one bootstrap falls as α grows, from a measured 87% at α=1 to 66% at α=5, so pricing a bootstrap by its key-switches alone would overstate the special-modulus penalty. With n_{ct} the ciphertexts a full-width bootstrap refreshes and n_{PBTS} those refreshed by partial bootstraps, the bootstraps contribute $(B\, n_{\mathrm{ct}} + n_{\mathrm{PBTS}})\, t_{\mathrm{BTS}}$, with column packing fixing n_{ct} at the hidden dimension H. Across the layer, $n(\ell)$ and $m(\ell)$ count the non-bootstrap key-switches and the key-switch-free operations at level ℓ, and the layer cost is then

$$T \;\approx\; (B\, n_{\mathrm{ct}} + n_{\mathrm{PBTS}})\, t_{\mathrm{BTS}} \;+\; \sum_{\ell} \left[\, n(\ell)\, t_{\mathrm{KS}}(\ell) + m(\ell)\, \ell \,\right]. \tag{4}$$

Prior cost models amortize one multiplication per level down a generic chain (Jung et al., 2021; Kim et al., 2022); Eq. (4) lifts the per-switch cost to a layer-level objective weighted by the executed schedule, and Appendix B.2 gives the per-level census behind t_{BTS} and its fit across two libraries. The key-switches of both terms of Eq. (4) grow with dnum, so a shorter chain cheapens every key-switch. A larger α lowers $\mathrm{dnum} = \lceil \ell/\alpha \rceil$ only in steps and between them merely widens the $(\ell+\alpha)$ modulus. It therefore cheapens the bootstrap at every one of its 14 limbs, measured 0.651 to 0.321 s in Appendix B.1, but not every key-switch along the chain. The same shift shrinks the evaluation keys, which store dnum digits over the extended modulus. Past a point, however, both moves force an added bootstrap, the shorter chain by restoring fewer levels and the larger α by taking modulus

bits the chain could hold. That bootstrap is worth roughly a thousand key-switches by Eq. (3) and outweighs the per-switch saving. Among the (B, L, α) satisfying the preceding constraints, we price each candidate by T and take the cheapest, $(B^\star, L^\star, \alpha^\star)$, at the largest feasible α for each chain length. The bootstraps that α cheapens carry 11.81 of the layer's 21.54 s (Table 2).

Operator redesign. OBELISK first inherits its nonlinear operators from prior work, GELU and softmax from THOR (Moon et al., 2025) at 11 and 10 levels and LayerNorm from MOAI (Zhang et al., 2026) at 20. A layer then consumes $D = 67$ levels, and LayerNorm sets $\delta = 20$. Over these operators, the constraints alone pin the budget at $(4, 21, 1)$, since the floor forces $L \geq 21$, security then admits only $\alpha = 1$, and covering $D = 67$ takes $B \geq 4$. OBELISK instead redesigns the two deepest operators. Newton iterations placed around partial bootstraps hold LayerNorm to 2 levels, and GELU's depth drops from 11 to 10 levels, as the level trajectories in Figure 2 show; Appendix B.3 gives each operator's update, its fitted domain, and the polynomial degrees behind these depths. The width-expanded FFN, whose matmuls bracket the shortened GELU across 12 levels, becomes the deepest block so far at $\delta = 12$, and the consumption falls to $D = 30$. The redesigned operators keep the full encrypted model from diverging, and Table 4 prices their unencrypted cost.

The resulting budget. The redesigned floor and the security cap now admit chains of 13 to 21 levels, and pricing the candidates across this range selects $(2, 16, 5)$, the cheapest of them, as Figure 3 traces. Section 3.2 tightens the range once the streams are fixed. The bootstrap term of each point is measured at its candidate budget, and the key-switch terms are priced by Eq. (2) over the measured deployed schedule, re-leveled to that candidate's chain. The operators affect the selection only through δ and D, so redesigning them changes the feasible set, not the cost objective. The budget therefore follows from a model's operators before any encrypted run, with Table 1 quantifying its evaluation-time and key-memory consequences.

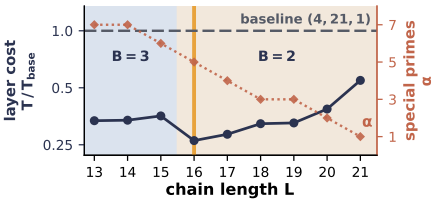


Figure 3: Modeled layer cost versus chain length, each point at the largest α its chain admits. T of Eq. (4) is normalized to the $(4, 21, 1)$ baseline. Shading marks the two- and three-bootstrap regions, and the vertical line marks the deployed L=16.

3.2 Process-in-Pieces Execution

The lifetime problem. The level budget shrinks the keys and ciphertexts, but the intermediates a layer holds at once keep its peak near 65.8 GiB, well above a commodity accelerator's capacity. Two working sets dominate that peak, the attention that materializes all twelve heads and the feed-forward network that widens the hidden state from 768 to 3072 ciphertexts. A ciphertext's memory scales with its level, so an operator chain enters large and exits small. Process-in-Pieces exploits this gap, evaluating attention one head and the feed-forward network one chunk of columns at a time, as in Figure 2. Each piece enters at its high level, leaves small, and folds into a running accumulator, so the peak is the 768-ciphertext input, one piece's working set, and the accumulator.

The peak model. A level-ℓ ciphertext occupies $|c_\ell| = 2\ell N$ coefficients, two polynomials of ℓ limbs with N coefficients each, which at $N = 2^{16}$ and eight bytes a coefficient is exactly ℓ MiB. Let d_h be the head dimension, T_{seq} the sequence length, and F the granularity-independent floor set by evaluation keys, resident input, and context. At the $QK^\top$ step, where the attention working set peaks, a resident head holds d_h paired Q, K ciphertexts at level ℓ_A and T_{seq} score ciphertexts at level ℓ_s (Table 2), so its working set is $w_A = d_h\,|c_{\ell_A}| + T_{\text{seq}}\,|c_{\ell_s}|$. A feed-forward column holds a single intermediate ciphertext at level ℓ_F, so $w_F = |c_{\ell_F}|$. Holding M heads and K columns at their high levels then gives the layer peak

$$\text{Peak}(M, K) \approx F + \max\big(M\,w_A,\ K\,w_F\big), \tag{5}$$

where attention and the FFN are largest at different points, so the larger working set rather than their sum fixes the peak. PiP takes both to their smallest feasible values, M=1 and K=128, dropping the peak from its untiled $(12, 3072)$ value to just above F (Appendix C).

The descent schedule. A full-width bootstrap cannot split a PiP stream. Refreshing a piece midstream would lift it back to the top, so the accumulator would hold high-level ciphertexts and the peak would return. The number of ciphertexts to refresh also grows along the stream, from d_h paired Q, K before $QK^\top$ to T_{seq} scores after it. The attention stream spends 14 levels, descending from 15 to 1 as each head folds from the Q, K, and V projections through its W_O slice. Each FFN chunk traverses the up-projection, GELU, and down-projection as one 12-level block from 14 to 2, so the deeper attention stream updates δ to 14. The deployed chain still clears this floor ($L{-}1 = 15 \geq 14$), so the selected $(2, 16, 5)$ stands, while the $B{=}3$ candidate of Figure 3 clears it exactly ($L{-}1 = 14$) and is excluded on cost, since covering $D{=}30$ from a 14-level segment forces a third full-width bootstrap. Each LayerNorm's affine, the elementwise scale γ and shift β, is folded into the matmul it feeds, so both streams carry only the normalized tensor. The residual, however, must add back the affine output. Each stream derives it from the normalized tensor only after its last piece folds in, as Figure 2 traces. No full-width high-level copy therefore ever transiently doubles the peak. Attention alone defers its value projection past softmax, keeping V off the peak that forms as a head's scores appear. The two full-width bootstraps follow the same discipline, refreshing their 768 ciphertexts as a stream. Only the in-flight piece holds bootstrap temporaries, so the refresh phase stays below the attention peak (Table 2).

3.3 Hybrid Real-Complex Representation

The representation problem. The faster CKKS representation, real or complex, varies by operation, and where complex wins, the operation runs 1.87 to 1.97× faster (Appendix D). HRC therefore chooses per operation, with real as the default. An operation selected for complex packs its real inputs, executes, and unpacks them back. A complex pairing must meet three criteria. The operation must offer two ciphertexts to pair (**P**). Both the packing into complex and the return to real must be level-free, since the budget's chain has no level to spare (Section 3.1). The two channels must also come back separated (**S**), kept apart by the operation or projected by a conjugation. The operation must be costly enough that halving it outweighs the pairing overhead, a net gain (**G**). Across the layer only the two ciphertext–ciphertext matmuls ($QK^\top$, SM·V) together with the full-width and softmax partial bootstraps meet all three, so they run complex (Table 2). The plaintext–ciphertext matmuls and the nonlinear operators fail separability, since their return to real cannot split the paired lanes. The LayerNorm partial bootstrap fails pairability, having collapsed to a single ciphertext.

Paired execution. At the two full-width bootstraps we pack two real ciphertexts into one complex, refresh both together, and split them back, halving their 768 bootstraps to 384 (Bossuat et al., 2021; EPFL-LDS, Tune Insight SA, 2024). The split-back conjugation also projects each output onto its real lane, discarding the imaginary noise a real-native bootstrap would seed into the cascade. The paired magnitude $\sqrt{a^2 + b^2}$ is up to $\sqrt{2}$ times the larger operand, which the first bootstrap absorbs and the second clears by a level-free rescaling (Appendix D). Both matmuls use standard complex packing (Cheon et al., 2017; Moon et al., 2025; Boemer et al., 2019; Park et al., 2025) along the head dimension, pairing adjacent columns. $QK^\top$ pairs both operands with the query conjugated, while SM·V pairs only the value. Each halves its rotations and ciphertext–ciphertext multiplications to $d_h/2$, adds no level, and recovers the real result exactly. Pairing also halves what stays resident, since the $2d_h$ real query and key ciphertexts a head would hold become the d_h paired ones of w_A. The representation decision thus does not incur a budget overhead while lowering the peak the lifetime decision manages (Sections 3.1 and 3.2).

4 Experiments

4.1 Experimental Setup

OBELISK runs on Phantom v2.0 (Yang et al., 2024) with bootstrapping ported from FHE-MP-CNN (Lee et al., 2022a), at ring dimension $N = 2^{16}$ and the deployed $(2, 16, 5)$ configuration. Measurements use a single NVIDIA H200 (141 GiB) and an A100 SXM4 (80 GiB), matching the baselines' reported hardware, both under CUDA 13.2 and Ubuntu 24.04. Table 2 spans four GPUs, down to a commodity RTX 4090, and Appendix E lists every host. Every measurement evaluates BERT-base, packing and scoring 256 sequences of length 128 in each ciphertext's 2^{15} slots. Evaluation time covers the server's forward pass alone, excluding client key generation, encryption, and decryption.

Table 1: Prior HE Transformers versus OBELISK on BERT-base. Every time and count is per input for the 12-layer encoder; the memory columns are absolute. B' counts single-ciphertext bootstraps, full-width and partial, derived for the baselines from each paper and its public code. n/r denotes not reported. †Sum of MOAI's per-module benchmarks, as its working set does not fit the A100.

System	Batch	Budget	Op count (per input)			Amortized server time (s/input)			Memory (GiB)	
		(B, L, α)	B'	HRot	KeySw	BTS (A100)	Total (A100)	Total (H200)	Peak	Keys
THOR	1	$(4, 14, 4)$	307	22,224	22,488	337.9	599.6	n/r	62.7	38.5
Powerformer	1	$(3, 12, 4)$	120	16,740	16,884	103.7	344.3	n/r	72.1	52.6
MOAI	256	$(4, 21, 1)$	145	9,648	10,008	227.8	283.3†	141.3	129.3	39.4
OBELISK	256	$(2, 16, 5)$	**38**	**4,824**	**5,004**	**20.5**	**32.5**	**21.5**	**21.63**	**6.4**
throughput gain	vs THOR		**8.1×**	**4.6×**	**4.5×**	**16.5×**	**18.4×**		**2.9×**	**6.0×**
	vs Powerformer		**3.2×**	**3.5×**	**3.4×**	**5.1×**	**10.6×**		**3.3×**	**8.2×**
matched reduction	vs MOAI		**3.8×**	**2.0×**	**2.0×**	**11.1×**	**8.7×**	**6.6×**	**6.0×**	**6.2×**

4.2 End-to-End Comparison

Main result. Against the prior homomorphic Transformers of Table 1, on the same H200 OBELISK is $6.6\times$ faster than MOAI at $6.0\times$ lower peak memory. The level schedule rather than the device sets the 21.63 GiB peak, so the same encrypted forward pass runs unassisted from the H200 down to a commodity RTX 4090, whereas MOAI overflows every card below the H200. On an A100 OBELISK runs end to end and is $8.7\times$ faster than MOAI's per-module lower bound, and against the batch-1 THOR and Powerformer the per-input gap widens to $18.4\times$ and $10.6\times$. The outcome traces back to the operation counts. OBELISK executes $3.8\times$ fewer bootstraps than MOAI and half its rotations, the bootstrap time falls $11.1\times$, and the evaluation keys shrink to 6.4 GiB against the baselines' 38.5 to 52.6 GiB. OBELISK also stores no encoded weights, folding each scalar weight into the multiplication itself, so the same model needs 660 MiB where THOR and Powerformer hold 109 and 93.8 GiB of resident CKKS plaintexts (Appendix G).

Metrics. Evaluation time is the batch wall-clock divided by the batch size. THOR and Powerformer run batch 1, so their per-input columns in Table 1 are single-input costs against the batch-256 amortization of MOAI and OBELISK. Their reductions are therefore throughput ratios, while the MOAI reductions give the matched evaluation-time comparison. HRot and KeySw are the matmuls' per-input algorithmic rotation and key-switch counts, as published for the baselines and derived from OBELISK's schedule, with neither side hoisting its rotations (Appendix B.2). Baseline times are as published, while the memory columns are our own measurement of their public code. MOAI encodes its weights during execution and excludes that cost from its reported evaluation time (Appendix F). NEXUS and Euston are named in the related work but do not enter the table (Appendix F). In absolute terms one deployed forward pass is $5{,}514$ s of H200 wall-clock for all 256 inputs, so the slot-filling batch buys throughput rather than interactive latency.

4.3 Time, Memory, and Granularity

Evaluation-time breakdown. Table 2 itemizes one end-to-end encrypted forward pass, each operation timed in place inside the run. The rows sum back to the measured total within 1%, the remainder being glue such as residual additions and modswitches. The value projection, deferred past softmax by PiP (Section 3.2), is timed inside the SM·V row, and the Repr. column marks the operations that HRC pairs into the complex lane (Section 3.3). The two full-width bootstraps dominate at 11.81 of the 21.54 s on the H200, with GELU's polynomial the largest operator at 4.6 s. Every operation holds only its own piece, so no full-width high-level intermediate is ever resident.

Peak memory decomposition. Table 3 decomposes the measured 21.63 GiB peak into a granularity-independent fixed floor of 18.6 GiB (layer input, evaluation keys, context, and a driver-runtime overhead) with an additional working set that scales with the attention granularity M, the two terms of Eq. (5) measured directly. The untiled $M{=}12$ pays 29.8 GiB and peaks at 48.4 GiB at this step, whereas PiP holds one head ($M{=}1$, 3.0 GiB) and drives the peak down to 21.63 GiB just above the floor. The $QK^\top$ time column stays at 1.12 s at both granularities, so the 26.8 GiB working-set reduction does not come at a latency penalty. Eq. (5) puts w_A at 2.5 GiB a head, matching the untiled 29.8 GiB to 1%, while the deployed head adds 0.5 GiB of transient workspace that twelve heads amortize away (Appendix C).

Table 2: Per-operation breakdown of one end-to-end encrypted forward pass. The run is at the deployed $(B, L, \alpha)=(2, 16, 5)$ on four GPUs. Repr. marks the HRC verdict, real (R) or complex (C). Peak is measured on the H200.

Component	Operation	Op type	Depth in → out	# ct	Repr.	Evaluation time (s/input, 12L)				Peak
						H200	A100	RTX PRO	RTX 4090	(GiB)
Attention (PiP ×12)	Q/K$_h$ proj	Pt×Ct	15→14	2×64	**R**	0.44	0.76	0.32	0.37	20.90
	$QK_h^\top$	Ct×Ct	14→13	128	**C**	1.12	2.01	1.04	1.40	**21.63**
	softmax$_h$	nonlinear	13→3	128	**R**	0.90	1.01	0.68	1.01	21.18
	partial BTS	boot	–	3	**C**	0.57	0.81	0.47	0.72	20.72
	SM·V$_h$	Ct×Ct	3→2	64	**C**	0.68	0.99	0.56	0.55	20.40
	W$_{O,h}$	Pt×Ct	2→1	64	**R**	0.06	0.09	0.06	0.05	19.22
BTS	BTS#1	boot	**1→16**	768	**C**	5.90	9.78	5.20	7.84	19.86
LayerNorm	LN1	nonlinear	16→14	768	**R**	0.09	0.13	0.08	0.11	19.25
	partial BTS	boot	–	2	**R**	0.03	0.05	0.03	0.04	18.76
FFN (PiP ×24)	FC1$_k$	Pt×Ct	14→13	128	**R**	0.76	1.33	0.59	0.62	20.83
	GELU$_k$	nonlinear	13→3	128	**R**	4.56	5.19	3.51	5.03	21.26
	FC2$_k$ + accum	Pt×Ct	3→2	128	**R**	0.23	0.37	0.19	0.18	19.49
LayerNorm	LN2-pre	nonlinear	2→1	768	**R**	0.01	0.01	0.01	0.02	8.58
	BTS#2	boot	**1→16**	768	**C**	5.91	9.76	5.20	7.97	19.87
	LN2-post	nonlinear	16→15	768	**R**	0.10	0.12	0.10	0.14	19.99
	partial BTS	boot	–	2	**R**	0.03	0.05	0.03	0.04	19.48
Total						**21.54**	**32.50**	**18.10**	**26.18**	**21.63**

Table 3: Measured memory components and $QK^\top$ time at the untiled (M=12) and deployed (M=1) attention granularities, both under HRC pairing. The layer input is 768 ciphertexts, the evaluation keys 29 Galois and 1 relinearization key, the CKKS context the moduli and NTT twiddles, and the working set the Q, K pairs and scores of Eq. (5). †Varies slightly by CUDA version and GPU.

Granularity	Fixed floor (GiB)				Working set	Peak	$QK^\top$ time
	Layer input	Eval. keys	CKKS context	CUDA runtime†	(GiB)	(GiB)	(s)
Untiled (M=12)	11.25	6.40	0.21	0.76	**29.8**	**48.4**	1.12
Deployed (M=1)					**3.0**	**21.63**	1.12

Granularity sweep. We sweep PiP granularity because smaller pieces reduce the working set but can introduce per-piece latency overhead. In the FFN, the piece size is the chunk width K. Making K too small increases the number of chunks and raises evaluation time by up to 51% as $K \to 1$, whereas making K too large gives no further time reduction and only increases peak memory beyond $K \approx 256$ (Figure 4). The lower edge is a time wall of repeated launch and accumulation overhead, the upper a memory wall of more resident high-level ciphertexts. Attention is swept separately at head granularity M; a single head already amortizes the overhead, so M=1 gives the smallest working set without increasing evaluation time. The deployed setting, M=1 and K=128, therefore stays within 2% of the FFN time floor while keeping the peak at 20–22 GiB across the evaluated GPUs, below the 24 GiB target and far below the 65.8 GiB single-piece pipeline. The same tiling would slow plaintext execution, so this no-latency tiling is specific to encrypted execution (Appendix C).

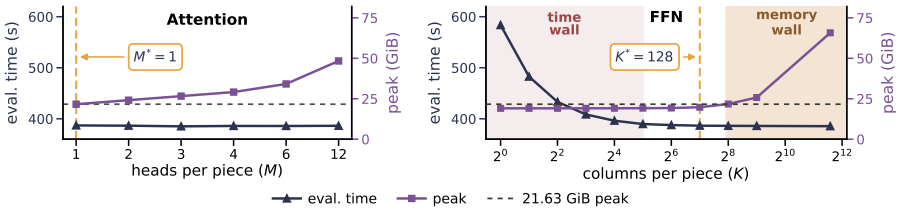
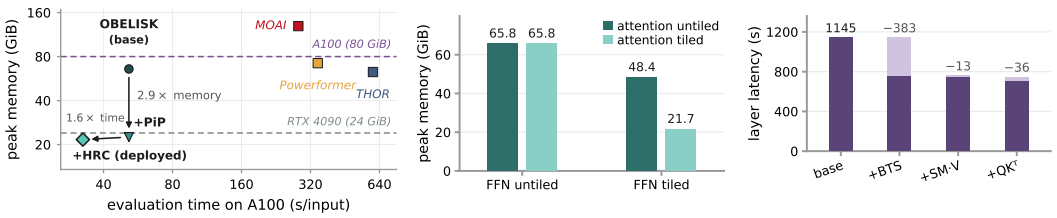


Figure 4: PiP granularity sweep. Evaluation time and peak memory against attention granularity M (left) and FFN piece size K (right). The dashed line is the deployed 21.63 GiB peak. Both axes are one encoder layer at batch 256 on the RTX PRO 6000, so the 386 s time floor is Table 2's 18.10 s per input over twelve layers.

4.4 Ablation Studies

OBELISK (base) includes budget co-design and three lossless execution-level optimizations (Appendix G), so both levers sit on the same base. Budget co-design is not ablated separately end to end.



(a) systems (b) inside PiP (c) inside HRC

Figure 5: (a) Evaluation time on the A100 and H200 peak memory of different FHE BERT systems. (b) Peak memory of one encoder layer at batch 256 over the two PiP axes, untiled being OBELISK (base) + HRC. (c) Latency of the same layer as HRC is applied one group at a time, each pale cap the time that step removes; base is OBELISK (base) + PiP, and the bars are wall-clock, not Table 2's per-input amortization.

Reverting it restores the 20-level LayerNorm it removes, which is a different operator set rather than the same system at a different budget. Appendix B.1 prices the budget candidates over the deployed schedule instead. Appendix G separates the two levers with a 2×2 factorial and prices the fused kernel, the stride key, and the cached bootstrap encoding. Figure 5 plots the three cumulative stages. PiP cuts peak memory $2.9\times$ without changing evaluation time or operation counts, and HRC then halves the single-ciphertext bootstraps and the two ciphertext–ciphertext matmuls, taking evaluation time down $1.6\times$ while trimming a further 0.9 GiB off the peak as the paired attention operands halve. Tiling attention alone leaves the peak at 65.8 GiB, since the feed-forward binds either way. Tiling the feed-forward alone only reaches 48.4 GiB, and only both together reach 21.7 GiB. Inside HRC the two bootstraps carry most of the gain, 1145 to 762 s, and only $QK^\top$ moves the peak.

4.5 Accuracy

Table 4 traces the gap to plaintext, the nonlinear operator approximations fitted to the tightened depth and the homomorphic noise, on the full validation sets of four GLUE tasks (Wang et al., 2019), each with its own public fine-tuned checkpoint. The polynomials themselves cost at most one sample per task unencrypted, so the remainder is CKKS noise that the shallower 16-level schedule has less headroom to absorb. Across the four tasks the encrypted model stays within 2.52 pp of its plaintext baseline, with the largest gap on RTE.

Table 4: OBELISK's per-task GLUE accuracy. G, G-LN, and G-LN-S cumulatively substitute the polynomial GELU, LayerNorm, and softmax into the unencrypted model, and the Encrypted column runs the full 12-layer cascade.

Task	Metric	Unencrypted				Encrypted
		Baseline	G	G-LN	G-LN-S	
SST-2	Acc	92.43	92.32	92.32	92.32	**91.63**
QNLI	Acc	91.54	91.63	91.62	91.62	**90.04**
RTE	Acc	72.56	72.20	72.20	72.20	**70.04**
MRPC	Acc	87.75	87.75	87.75	87.75	**85.54**
	F1	91.35	91.35	91.35	91.35	**90.18**

5 Conclusion and Limitations

Packing fixes the logical dataflow but not its execution, and encrypted inference's evaluation time and memory are set by the level budget, the intermediate-ciphertext lifetime, and the slot representation. Choosing the three jointly through budget–operator co-design, PiP, and HRC cuts end-to-end evaluation time $6.6\times$ and peak memory $6.0\times$ against MOAI on the same H200, widening to $8.7\times$ against MOAI's per-module lower bound on an A100 it cannot fit and fitting the forward pass onto a commodity RTX 4090. We probed one LLaMA-3-8B prefill decoder layer with grouped-query attention, which runs at cosine similarity 0.997 of plaintext with no diverged head and a 92 GiB single-GPU peak (Appendix I).

Limitations and future work. Carrying that layer to a whole model is a redesign rather than an extension. Approximation error compounds with depth, so which polynomial each nonlinearity needs, and therefore how much depth each layer consumes, becomes a per-layer decision across 32 layers, and the budget and the packing layout are in turn chosen for that depth. Generation adds a second obstacle, a growing KV cache that needs a packing beyond today's batch-filled traces. Both sit upstream of post-packing execution, so we leave them to future work.

REPRODUCIBILITY STATEMENT

The deployed parameters are stated in the main text: ring dimension $N = 2^{16}$, the budget $(B, L, \alpha) = (2, 16, 5)$ with the prime bit-widths of Section 3.1, the PiP granularities $M{=}1$ and $K{=}128$ (Section 3.2), and the HRC verdict of every operation (Table 2). Section 3.1 gives the cost model and the constraints that select the budget, and Section 4.5 the GLUE validation sets used for accuracy, each scored with its own public fine-tuned checkpoint. The hardware, CUDA version, operating system, and CKKS library versions are listed in the experimental setup. The appendix carries the evidence behind each of these: Appendix A the CKKS parameter set and its lattice-estimator security check, Appendix B.1 the budget-selection procedure and the measured sweep it minimizes over, Appendix B.3 the operator constructions and their fitted windows, Appendix E the measurement hosts and the timing and peak-memory instruments, Appendix F the provenance of every baseline cell, and Appendix H the calibration protocol. Source code, the per-operation measurement scripts, the polynomial coefficients of the redesigned GELU, LayerNorm, and softmax, and the checkpoint identifier for every task are provided as anonymous supplementary material.

AI USE STATEMENT

In this work we used generative AI tools in four ways: as coding assistants while implementing and instrumenting the system; for retrieval and discovery, to locate related work and to check the claims we attribute to it against primary sources; to draft passages of the manuscript; and to edit the manuscript's text for clarity and readability. Every AI-assisted contribution was reviewed by the authors: code was read, executed, and validated against the fidelity gates and the measurements this paper reports; every citation and every quantity attributed to prior work was confirmed against the cited paper; and drafted or edited text was checked against the underlying results. No measurement reported in this paper was produced by a generative model; quantities derived arithmetically from the cost model of Section 3.1 were computed with AI assistance and are reproducible from the stated equations. We did not use generative AI tools to develop the theoretical model, formulate the mathematical claims, design the experiments, or interpret the results. We have reviewed all AI-assisted work, and we take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Mohammad Al-Rubaie and J. Morris Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2):49–58, 2019.

Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, 2018.

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

Wei Ao and Vishnu Naresh Boddeti. AutoFHE: Automated adaption of CNNs for efficient evaluation over FHE. In *33rd USENIX Security Symposium (USENIX Security 24)*, 2024.

Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. nGraph-HE2: A high-throughput framework for neural network inference on encrypted data. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC)*, 2019.

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

Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. THE-X: Privacy-preserving transformer inference with homomorphic encryption. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 3510–3520, 2022.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology — ASIACRYPT 2017*, pp. 409–437. Springer, 2017.

Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *EUROCRYPT*, 2018.

Seonyoung Cheon, Yongwoo Lee, Dongkwan Kim, Ju Min Lee, Sunchul Jung, Taekyung Kim, Dongyoon Lee, and Hanjun Kim. DaCapo: Automatic bootstrapping management for efficient fully homomorphic encryption. In *33rd USENIX Security Symposium (USENIX Security 24)*, 2024.

Seonyoung Cheon, Yongwoo Lee, Hoyun Youm, Dongkwan Kim, Sungwoo Yun, Kunmo Jeong, Dongyoon Lee, and Hanjun Kim. HALO: Loop-aware bootstrapping management for fully homomorphic encryption. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2025.

Leo De Castro, Daniel Escudero, Adya Agrawal, Antigoni Polychroniadou, and Manuela Veloso. EncryptedLLM: Privacy-preserving large language model inference via GPU-accelerated fully homomorphic encryption. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

DESILO. Liberate.FHE: An open-source FHE library for bridging the gap between theory and practice. `https://github.com/Desilo/liberate-fhe`, 2023.

Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, Wenguang Chen, and Jianying Zhou. PUMA: Secure inference of LLaMA-7B in five minutes. *Security and Safety*, 4:2025014, 2025.

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

Xinwen Gao, Shaojing Fu, Lin Liu, Zhuotao Liu, Yuchuan Luo, and Yongjun Wang. Euston: Efficient and user-friendly secure transformer inference with non-interactivity. In *2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 201–210, 2016.

Zhaoting Gong, Ran Ran, Fan Yao, and Wujie Wen. Scaling long-sequence homomorphic encrypted transformer inference via hybrid parallelism on multi-gpu systems. In *Proceedings of the 40th ACM International Conference on Supercomputing (ICS)*, pp. 1206–1219, 2026.

Kyoohyung Han and Dohyeong Ki. Better bootstrapping for approximate homomorphic encryption. In *Topics in Cryptology – CT-RSA 2020*, 2020.

Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Siddharth Jayashankar, Joshua Kim, Michael B. Sullivan, Wenting Zheng, and Dimitrios Skarlatos. A scalable multi-GPU framework for encrypted large-model inference. arXiv:2512.11269, 2025.

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*, 2021.

Sangpyo Kim, Jongmin Kim, Michael Jaemin Kim, Wonkyung Jung, John Kim, Minsoo Rhu, and Jung Ho Ahn. BTS: An accelerator for bootstrappable fully homomorphic encryption. In *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)*, 2022.

Eunsang Lee, Joon-Woo Lee, Junghyun Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and Woosuk Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022a.

Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, and Jong-Seon No. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 10: 30039–30054, 2022b.

Joon-Woo Lee, Eunsang Lee, Young-Sik Kim, and Jong-Seon No. Rotation key reduction for client-server systems of deep neural network on fully homomorphic encryption. In *Advances in Cryptology – ASIACRYPT 2023*, 2023a.

Yongwoo Lee, Seonyoung Cheon, Dongkwan Kim, Dongyoon Lee, and Hanjun Kim. ELASM: Error-latency-aware scale management for fully homomorphic encryption. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023b.

Yan Liu, Jianxin Lai, Long Li, Tianxiang Sui, Linjie Xiao, Peng Yuan, Xiaojing Zhang, Qing Zhu, Wenguang Chen, and Jingling Xue. ReSBM: Region-based scale and minimal-level bootstrapping management for FHE via min-cut. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2025.

Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. BumbleBee: Secure two-party inference framework for large transformers. In *Proceedings of the 32nd Annual Network and Distributed System Security Symposium (NDSS)*, 2025.

Microsoft Research. Microsoft SEAL (release 4.1). `https://github.com/microsoft/SEAL`, January 2023. Redmond, WA. NEXUS uses a bootstrapping-friendly variant (SEAL-4.1-bs).

Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. THOR: Secure transformer inference with homomorphic encryption. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2025.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. BOLT: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

Dongjin Park, Eunsang Lee, and Joon-Woo Lee. Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Jaiyoung Park, Sejin Park, Jai Hyun Park, Jung Ho Ahn, Jung Hee Cheon, Guillaume Hanrot, Jung Woo Kim, Minje Park, and Damien Stehlé. Scaling up FHE-based privacy-preserving ML: Higher throughput, longer inputs for LLaMA-3-8B. arXiv:2601.18511v2, 2026.

Sieun Seo and Chohong Min. Zephyr: GPU-efficient homomorphic encryption for privacy-preserving transformer inference. Cryptology ePrint Archive, Paper 2026/932, 2026.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2019.

Hao Yang, Shiyu Shen, Wangchen Dai, Lu Zhou, Zhe Liu, and Yunlei Zhao. Phantom: A CUDA-accelerated word-wise homomorphic encryption library. *IEEE Transactions on Dependable and Secure Computing*, 2024.

Ye Yu, Yifan Zhou, Yi Chen, Pedro Soto, Wenjie Xiong, and Meng Li. Cachemir: Fully homomorphic encrypted inference of generative large language model with KV cache. arXiv:2602.11470, 2026.

Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. In *Proceedings of the 32nd Annual Network and Distributed System Security Symposium (NDSS)*, 2025.

Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok-Yan Lam. MOAI: Module-optimizing architecture for non-interactive secure transformer inference. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Yufan Zhu, Chao Jin, Khin Mi Mi Aung, and Xiaokui Xiao. EncFormer: Secure and efficient transformer inference over encrypted data. *IEEE Transactions on Dependable and Secure Computing*, 23(5):11582–11597, 2026.

Itamar Zimerman, Moran Baruch, Nir Drucker, Gilad Ezov, Omri Soceanu, and Lior Wolf. Converting transformers to polynomial form for secure inference over homomorphic encryption. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

APPENDIX

Appendix A gives the deployed parameters and their security analysis, Appendices B–D provide more detail for each of the three method subsections, Sections 3.1–3.3, and Appendices E–H the experiments, with Appendix E defining the instruments behind every reported number. Appendix I reports an isolated LLaMA-3-8B decoder layer as a generality probe rather than a full-model result.

A CKKS PARAMETERS AND SECURITY

This appendix provides supplementary material on Eq. (1) and the 128-bit security claim of the main text. Three of the four terms of Eq. (1) are fixed once the chain length is chosen, so at $L = 16$ the security constraint of the main text reduces to one free variable,

$$\log QP \;=\; \underbrace{51 + 46 \cdot 15 + 51 \cdot 14}_{1455,\ \text{fixed at } L=16} + 58\,\alpha \;\le\; 1777, \tag{A.1}$$

which gives 1745 bits at the deployed $\alpha = 5$ and rejects $\alpha = 6$ at 1803. The special modulus is therefore at its secure maximum for this chain, a fact the feasibility sweep of Appendix B.1 rests on. At ring dimension $N = 2^{16}$, the secret is a sparse ternary key of Hamming weight 192 under an error standard deviation $\sigma = 3.2$.

We check the classical security directly with the lattice estimator (Albrecht et al., 2015) rather than resting on the tabulated ceiling. A larger modulus is easier to attack, so security falls as $\log QP$ grows and the standard's 1777 is a ceiling. Table A.1 estimates the attack cost in bits at the deployed point and at that ceiling, under a uniform ternary secret and under the deployed sparse one, and the parameters meet or exceed the HomomorphicEncryption.org 128-bit classical parameterization with about 2.6 bits to spare. Two notes qualify this table. First, the absolute scale is the conservative core-SVP bound, on which the standard's own accepted 128-bit points read 92.9 to 99.9 bits across ring dimensions. The 128-bit label therefore sits well above the core-SVP bound for every accepted CKKS parameterization, and only the relative comparison carries information, which is how we use it. Second, the estimator ran the dual-hybrid attack that specifically targets sparse secrets, the dominant attack on the ceiling row, so the 0.3- to 0.6-bit sparse penalty is measured against its actual threat rather than assumed.

Table A.1: Lattice-estimator attack cost per secret distribution (higher is harder).

		Attack cost (bits)	
$\log QP$	Configuration	uniform key	sparse key
1745	deployed $\alpha = 5$	94.0	**93.7**
1777	128-bit ceiling	91.7	91.1

The comparison of Appendix F is not confounded by security level. MOAI states 1743 bits in its paper, OBELISK deploys 1745, and THOR's chain extracted from its released code totals ∼1759, so the three actually used budgets sit within 16 bits of one another at the same ring dimension. Powerformer runs the same Liberate 128-bit bootstrapping preset as THOR. MOAI's figure is author-stated, THOR's code-derived rather than measured.

Classical security is what the main text claims. Under a more conservative quantum core-SVP budget the deployed $\alpha = 5$ falls short, and Eq. (A.1) recovers the margin through its one free term: $\alpha = 3$ gives $\log QP = 1455 + 58 \cdot 3 = 1629$ bits, inside that budget, at a measured 9.6% more evaluation time and 3.7 GiB more peak memory. The level chain L is unchanged, so the operator schedule and every depth of Appendix B.3 carry to the post-quantum point unmodified.

B Budget and Operator Co-design

This appendix backs Section 3.1. Figure B.1 shows three development configurations, each sound where it was validated, that diverged only in the full encrypted cascade. Their resolutions are, in order, the wide basin of Table B.5, calibration under encryption (Appendix H), and the paired bootstrap (Appendix D).

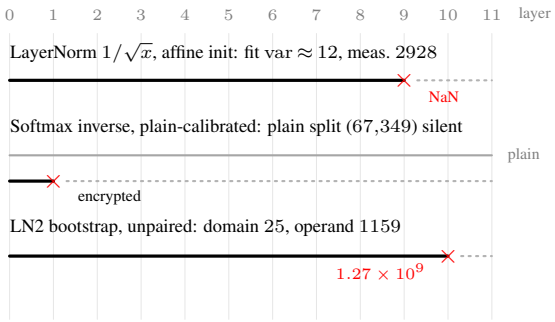


Figure B.1: Configurations that passed every local check and still diverged in the encrypted cascade.

B.1 Feasibility Sweep

The deployed budget is the cheapest chain the constraints admit, measured. The width-expanded FFN sets a provisional floor of $\delta = 12$, which the lifetime rule raises to the binding $\delta = 14$, since a bootstrap cannot split a piecewise stream and the attention stream spans 14 levels. With the security bound capping $L \le 21$ at $\alpha = 1$, the admitted range is $15 \le L \le 21$, which Algorithm B.1 enumerates and Table B.1 measures, at the largest α each L admits. That restriction is not free. Since $\lceil \ell/\alpha \rceil$ steps, a larger α cheapens a key-switch only when it lowers the digit count, and otherwise costs a further $\mathrm{dnum}+2$ for the wider modulus alone. Over an L=16 chain α=5 is dearer than α=4 at ten of the sixteen levels. It cheapens the bootstrap at all fourteen of its limbs, however, and the Per-BTS column measures that monotonicity, so the largest feasible α is the cheapest for the term that carries the layer. The L=18 candidate is priced by the L=19 row through their shared key-switch digit count. The cheapest candidate is the shortest chain covering $D = 30$ in two bootstraps, since coverage forces a third full-width bootstrap below $L = 16$ that outweighs every per-switch saving. Toggling α alone at the fixed 16-level chain shrinks evaluation keys 29.09 → 6.59 GiB (−77%), layer peak −43%, and layer time −37%. Three notes qualify Table B.1. The costs are those of unpaired single-ciphertext bootstraps, scaled by the 768 ciphertexts, so the pairing of Appendix D halves every row uniformly and Layer BTS is normalized to the deployed point. The Key column is the sweep harness's key-generation delta, which spans the bootstrap key set and its linear-transform tables and is therefore larger than the main text's evaluation-key figure. At $L \ge 17$ the library's minimax coefficients are out of their tuned range, so those rows carry representative cost but not decryptable output.

The largest feasible α is not the cheapest at every level. Equation (2) is not monotone in α at a fixed level. Table B.2 evaluates it over the deployed L=16 chain and the 14 bootstrap limbs above

Algorithm B.1 Budget selection from operator depths

Require: layer consumption D, deepest unsplittable stream depth δ, security cap $L_{\max}$
Ensure: budget $(B^\star, L^\star, \alpha^\star)$

$T^\star \leftarrow \infty$
for $L = \delta + 1$ **to** $L_{\max}$ **do**
  $B \leftarrow \lceil D/(L-1) \rceil$
  $\alpha \leftarrow \max\{a \in \mathbb{Z}_{\geq 1} : 51 + 46(L-1) + 51{\cdot}14 + 58a \leq 1777\}$
  if no such a exists **then**
    continue
  end if
  $T \leftarrow$ layer cost of (B, L, α)
  if $T < T^\star$ **then**
    $T^\star \leftarrow T$; $(B^\star, L^\star, \alpha^\star) \leftarrow (B, L, \alpha)$
  end if
end for
return $(B^\star, L^\star, \alpha^\star)$

Table B.1: Measured bootstrap cost across the admitted candidate budgets, RTX PRO 6000.

L	α	B	Per-BTS (s)	Key (GiB)	Layer BTS
21	1	2	0.651	39.44	2.03×
20	2	2	0.465	19.19	1.45×
19	3	2	0.395	12.44	1.23×
17	4	2	0.347	8.81	1.08×
16	5	2	0.321	6.59	**1.00×**
15	6	3	0.304	5.50	1.42×

it. The digit count steps, so each increment of α either lowers $\lceil \ell/\alpha \rceil$ or buys nothing and costs a further $\mathrm{dnum} + 2$ for the wider modulus. Accordingly $\alpha{=}5$ is dearer than $\alpha{=}4$ at ten of the sixteen chain levels, and the largest feasible α is not the per-level minimum at any of them. Summed over the schedule the steps still outweigh the penalty, by 2.4% along the chain and by 11.8% over the bootstrap limbs, and it is the second of these that decides the budget: the two full-width bootstraps carry 11.81 of the layer's 21.54 s. Algorithm B.1 therefore slaves α to its cap without loss on this schedule, though the guarantee is one of summed cost, not of per-level dominance.

Table B.2: $\sum_\ell t_{\mathrm{KS}}(\ell)$ of Eq. (2) at the deployed $L{=}16$, in units of $N \log N$. The chain column sums the 16 levels a ciphertext descends, the bootstrap column the 14 limbs above it.

α	chain $\ell \leq 16$	bootstrap $\ell > 16$	total
1	1936	8974	10910
2	1260	5110	6370
3	1067	3838	4905
4	980	3246	4226
5	**956**	**2864**	**3820**

B.2 Bootstrap Cost Model

Intercepting every primitive with its execution level shows a bootstrap circuit invariant across (L, α): 168 key-switches and 1,233 limb-linear operations per bootstrap in our Phantom runtime, the engine shared with MOAI, against 522 and 1,375 in Liberate.FHE, the engine behind THOR and Powerformer. Priced per level, the census shows the key-switch share of one bootstrap falling from 87% at $\alpha{=}1$ to 66% at the deployed $\alpha{=}5$, the measured basis for the main text's caution against pricing bootstraps by key-switches alone. Pricing that census with the key-switch cost $t_{\mathrm{KS}}(\ell) = (\lceil \ell/\alpha \rceil + 2)(\ell + \alpha)$ and one transform per limb for the rest gives the full model

$$t_{\mathrm{BTS}} \;=\; c \sum_{\ell} \left[\, n_{\mathrm{boot}}(\ell)\, t_{\mathrm{KS}}(\ell) + m_{\mathrm{boot}}(\ell)\, \ell \,\right] + h\, N_{\mathrm{call}}, \tag{B.1}$$

Table B.3: The full model (B.1) against each system's measured bootstrap.

System	Units (10^3)	Meas. (ms)	Model (ms)	Err.
THOR	195.4	530.8	514.0	-3.2%
Powerformer	161.9	448.3	425.7	-5.0%
MOAI	191.1	635.0	623.0	-1.9%
OBELISK	63.5	274.8	287.5	$+4.6\%$

Table B.4: Bootstrap cost model (B.2) fitted per library.

Library	A (s)	C	R^2	Mean err.
Phantom	3.3×10^{-4}	17.7	0.96	4.8%
Liberate	7.3×10^{-4}	13.7	0.95	3.0%

with n_{boot} and m_{boot} the census counts at level ℓ, c the wall-clock per limb-transform, and $h = 86\ \mu$s a per-call host dispatch. The dispatch term h is the measured wall-to-busy gap of one bootstrap spread over its $N_{\mathrm{call}} = 1{,}401$ calls, and it is charged only to the Phantom systems, whose wall clocks carry this gap. A single $c = 2.63\ \mu$s prices all four systems' warm-median bootstraps, ten fresh processes each, within 5% in Table B.3, whose Units column is the bracketed sum.

Folding the schedule gives the closed form

$$t_{\mathrm{BTS}} \;=\; A\,(\mathrm{dnum} + C)(k + \alpha), \tag{B.2}$$

with A the per-transform wall-clock, k the level of the bootstrapped operand, and C the offset image of the key-switch-free term of Eq. (B.1), fitted per library by least squares in Table B.4: $R^2 \geq 0.95$ in both libraries, MOAI's held-out $\alpha = 1$ bootstrap predicted within 2%, the THOR and Powerformer Liberate builds agreeing within 1.3% at a matched (k, α), and the offset-free form $t \propto (\mathrm{dnum} + 2)(k + \alpha)$ collapsing to $R^2 \approx 0.1$. The model that drives the co-design is a property of the hybrid key-switch algorithm, not a curve fit to one implementation.

Neither the rotation and key-switch counts of Table 1 nor Eq. (2) assumes hoisted rotations. Hoisting trades decompositions for automorphisms and leaves unchanged the key-switch count that Eq. (2) prices, and neither our runtime nor the baselines' implements it.

B.3 Operator Approximation

The redesign starts from depths measured rather than assumed, the inherited GELU and softmax depths logged by instrumenting THOR's released code at every operation boundary. GELU spends 11 levels inherited against 10 deployed, softmax 10 against 10, and LayerNorm 20 against 2, taking the layer from $D = 67$ under the inherited $(4, 21, 1)$ budget to $D = 30$ under the deployed $(2, 16, 5)$. The GELU saving is the Paterson–Stockmeyer band rule, degree ≤ 15 costing four levels where 16–31 costs five. The LayerNorm saving relocates seven Newton iterations onto the single ciphertext holding the reduction statistic, where two partial bootstraps absorb them and the width-768 tensor pays only the two multiplications that apply the statistic. The accounting closes: restoring the inherited operators adds $1 + 18 + 18 = 37$ to the deployed $D = 30$, reproducing the inherited $D = 67$ exactly.

Table B.5 gives each construction and its depth cost. The inherited configurations fail where their windows stop short, GELU fitted to $[-32, 14]$ against a measured $[-61.5, 139.6]$ and LayerNorm valid to $\mathrm{var} \approx 12$ against a deep-layer 2928. The deployed GELU is refitted to $[-64, 150]$ over that measured envelope. A controlled substitution attributes $\sim 89\%$ of the 12-layer error to the LayerNorm inverse square root. The wide initialization that closes it holds on $[0.5, 3500]$ and must be evaluated in the Chebyshev basis by Clenshaw recurrence, the monomial form canceling catastrophically near variance 2600. Both LN2 variants charge the tensor the same two levels, the wide variant's extra depth running on the single statistic ciphertext, so the per-layer assignment of Table B.6 never touches the budget. The softmax normalizer's per-task constant is fitted on training data alone (Appendix H).

Table B.5: Construction and depth cost of each nonlinear operator, each depth term labeled with the stage that pays it.

Operator	Construction	Depth cost
GELU	$x\,(p_2(p_1(x/64)) + 0.5)$, deg-15 then 28, PS	4 (p_1) + 5 (p_2) + 1 ($\times\, x$), scale folded
Softmax sm1	exp deg-15, square, rot-free row sum, $1\times$ Newton (inline BTS)	5 (exp) + 1 (square) + 2 (Newton) + 1 (normalize) + 1 (drain)
Softmax sm2	sm1, exp refit to deg-11 on layer 2's envelope	4 (exp) + 1 (square) + 1 (mask) + 2 (Newton) + 1 (normalize) + 1 (drain)
LayerNorm 1	affine init, $7\times$ Newton, $2\times$ partial BTS	2 (tensor) + 24 (statistic ct, BTS-absorbed)
LN2 narrow	affine init $r_0 = a \cdot \mathrm{var} + b$	2 (tensor)
LN2 wide	Chebyshev deg-6 by Clenshaw, $4\times$ Newton, $2\times$ Goldschmidt	2 (tensor)

Table B.6: Deployed per-layer operator variants.

Layers	Softmax	LayerNorm 1	GELU	LayerNorm 2
0, 1, 11	sm1	standard	standard	narrow
2	sm2	standard	standard	wide
3–10	sm1	standard	standard	wide

C PROCESS-IN-PIECES

This appendix backs Sections 3.2 and 4.3. At $N = 2^{16}$ and eight bytes a coefficient the ciphertext size of Section 3.2 is $|c_\ell| = \ell$ MiB exactly, so both terms of w_A read off the levels Table 2 records for the $QK^\top$ step,

$$w_A \;=\; \underbrace{d_h\,|c_{\ell_A}|}_{64\,\times\,14\ \mathrm{MiB}\,=\,0.875\ \mathrm{GiB}} \;+\; \underbrace{T_{\mathrm{seq}}\,|c_{\ell_s}|}_{128\,\times\,13\ \mathrm{MiB}\,=\,1.625\ \mathrm{GiB}} \;=\; 2.5\ \mathrm{GiB\ a\ head.} \qquad \text{(C.1)}$$

The paired operands are the d_h of Eq. (D.1) rather than the $2d_h$ a real-native head would hold, which is where the representation decision enters the peak. Twelve heads give $2.5 \times 12 = 30.0$ GiB against the 29.8 the untiled $M{=}12$ column of Table 3 measures, an agreement to 1%. The deployed $M{=}1$ column reads 3.0 rather than 2.5, and the remaining 0.5 GiB is transient workspace that twelve heads amortize and one does not. The granularity that minimizes the working set costs no measurable evaluation time. Table C.1 sweeps both decomposition axes on the RTX PRO 6000, one encoder layer at batch 256 with the other axis held at its deployed value, and time is full-layer wall-clock, so the work independent of the swept axis sits in the denominator. Above the knee, where the sweep's time penalty flattens, the piece size is set by memory rather than by time. The run's tracing probes cost about one percent of its wall-clock. The peak column is measured in the configuration of the sweep, which predates the deployed configuration whose peaks Table C.2 reports. Configurations that additionally revert levers unrelated to tiling read higher still, so the sweep isolates the granularity term alone. Both blocks of the peak column also follow the peak model of the main text, the K block flat while attention binds and then linear at 0.016 GiB per column, the M block linear throughout

Table C.1: Both decomposition axes on an RTX PRO 6000, one encoder layer at batch 256, highlighted rows deployed.

	Pieces	Time (s)	vs. floor	Peak (GiB)
Feed-forward, piece width K				
1	3072	583.6	+51.4%	24.6
2	1536	482.6	+25.2%	24.6
4	768	433.1	+12.4%	24.6
8	384	408.5	+6.0%	24.6
16	192	396.3	+2.8%	24.6
32	96	389.7	+1.1%	24.6
64	48	387.7	+0.6%	24.6
128	24	386.5	+0.3%	24.6
256	12	386.2	+0.2%	26.2
512	6	386.0	+0.1%	30.4
3072	1	385.5	0	72.3
Attention, heads per piece M				
1	12	387.0	+0.5%	24.6
2	6	386.5	+0.3%	29.7
3	4	385.2	0	34.5
4	3	386.0	+0.2%	39.3
6	2	385.9	+0.2%	49.3
12	1	386.3	+0.3%	74.9

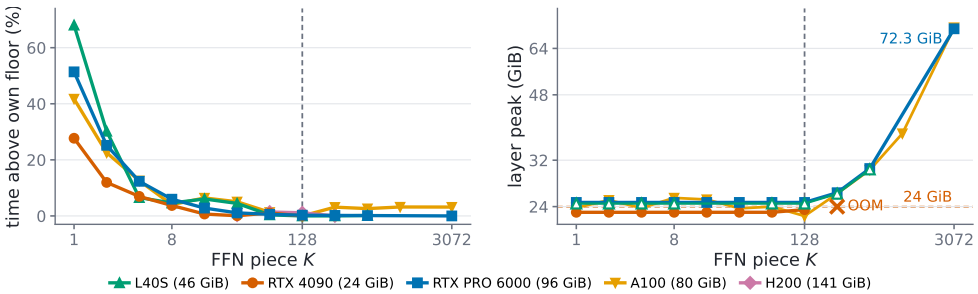


Figure C.1: The feed-forward sweep on five cards, time above each card's own floor (left) and full-layer live peak on the four cards that measure it (right).

at about 4.5 GiB per head. The deployed $M{=}1$ sits at the top of its band, so the finest attention granularity is free rather than fastest, and it is deployed because its working set is the smallest.

One piece size fits every card. Figure C.1 repeats the feed-forward sweep on five cards spanning 24 to 141 GiB, the L40S of Appendix E filling the span between the evaluated GPUs. Each curve is normalized to its own floor, since the cards run sweep-specific configurations and their absolute times are not comparable. The H200 sweep logs time only, the A100 curve is additionally normalized for that host's bimodal timing bands (Appendix E), and the two L40S peak points its log flags as sampler outliers are excluded. The shape is invariant. At $K{=}128$ every card sits within 1.2% of its own floor, finer pieces cost up to +68%, and coarser pieces buy nothing outside noise while the peak leaves its floor on every card that measures it and overflows the 24 GiB card at $K{=}256$. The deployed $K{=}128$ is therefore the largest piece that is simultaneously on the time floor and the memory floor of every device.

The cost inversion. Figure C.2 sweeps the tiling penalty against the granularity itself, with both settings measured on the same card, plaintext circles at batch 256 and batch 1, encrypted sweeps squares, and dashed verticals at the deployed $M{=}1$ and $K{=}128$. In plaintext (PyTorch at half precision, median of 50 iterations after 10 warmups, batch-matched at the 32,768 slots one ciphertext holds), the deployed tiles cost 1.83× and 2.68× at batch 256 and 11.5× and 13.7× at batch 1. The encrypted curves are the measured sweeps of Table C.1, with ratios within 0.6% of one across the entire range, since one homomorphic operation already saturates the card. The mechanisms

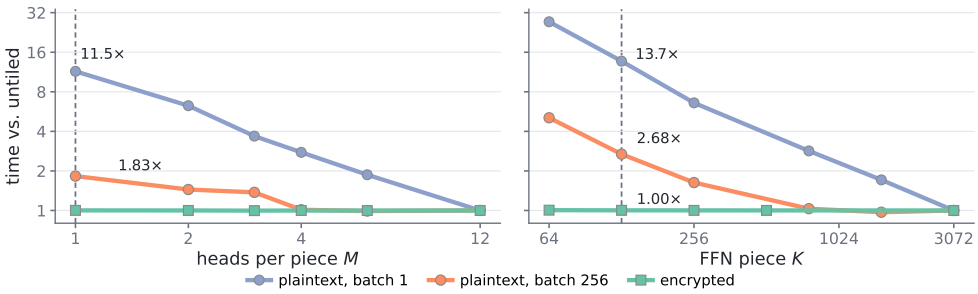


Figure C.2: Tiling penalty against granularity, plaintext versus encrypted, on the RTX PRO 6000.

Table C.2: The deployed layer peak across the four evaluated GPUs, per-process live metric of Appendix E.

Device	VRAM (GiB)	Layer peak (GiB)
H200	141	21.63
RTX PRO 6000	96	21.95
A100	80	21.67
RTX 4090	24	21.61–21.67

likewise do not carry over to CKKS. The online softmax at the heart of FlashAttention-style kernels recomputes a running maximum and a division per tile, both expensive polynomial approximations under CKKS. The decomposition axes were derived from the encrypted cost model instead, head boundaries from the packing and chunk boundaries from the elementwise nonlinearity.

Exactness and peak stability. Neither granularity trades accuracy for memory, since both are reassociations rather than approximations. Chunking the feed-forward network reorders a column sum, and the layer's cosine similarity against its plaintext reference and its output magnitude are identical at $K{=}64$, 128, 256, and 512. Grouping heads is likewise exact, with the output magnitude identical at $M{=}1$, 2, 6, and 12. Table C.2 closes the memory side. The deployed layer peak is stable within 0.34 GiB across the four evaluated GPUs, the RTX 4090 range spanning its twelve cascade layers, and binds at $QK^\top$ on every one. The first layer of a fresh process reads about 1.5 GiB higher, the memory pool's first touch charged to it, so the steady-state layer is the comparable figure.

D Hybrid Real-Complex Representation

This appendix backs Section 3.3.

The pairing map. Two real ciphertexts are carried in one complex ciphertext as $z = a + i\,b$, and the lanes are recovered through the conjugation automorphism $\kappa\colon z \mapsto \bar{z}$,

$$a \;=\; \tfrac{1}{2}\big(z + \kappa(z)\big), \qquad b \;=\; \tfrac{1}{2i}\big(z - \kappa(z)\big). \tag{D.1}$$

κ is a Galois automorphism of the ring, so it costs one key-switch and consumes no level, and the factor $\frac{1}{2}$ is absorbed into the scale rather than paid for with a multiplication. Recovery is therefore exact and level-free, which is what the budget of Section 3.1 requires of any pairing it admits.

Why the query is conjugated. Pair adjacent head columns as $q = q_1 + i\,q_2$ and $k = k_1 + i\,k_2$. The product with the query conjugated is

$$\kappa(q)\cdot k \;=\; (q_1 - i\,q_2)(k_1 + i\,k_2) \;=\; \underbrace{(q_1k_1 + q_2k_2)}_{\text{two score terms}} + i\,(q_1k_2 - q_2k_1), \tag{D.2}$$

so the real lane of a single complex multiplication already holds two terms of the inner product $\sum_c q_c k_c$ that the attention score accumulates. Summing Eq. (D.2) over $d_h/2$ paired columns gives the score exactly, at half the ciphertext–ciphertext multiplications and half the rotations. Without

the conjugation the real lane would read $q_1k_1 - q_2k_2$, which is not a partial inner product, and the pairing would buy nothing.

SM·V pairs only the value, because the score is already real. For $v = v_1 + i\,v_2$,

$$s \cdot (v_1 + i\,v_2) \;=\; s\,v_1 \;+\; i\,s\,v_2, \tag{D.3}$$

and the two products land in separate lanes for Eq. (D.1) to split, again at one multiplication for two.

The separability criterion. Eq. (D.1) recovers both lanes whenever the operation leaves them unmixed, which an $\mathbb{R}$-linear map applied lane-wise does and a polynomial does not, since $p(a + i\,b) \neq p(a) + i\,p(b)$ for any p of degree above one. That is the criterion Table D.1 applies, and it is why the nonlinear operators are excluded by construction rather than by measurement.

The three criteria, pairability, separability, and gain, are checked per operation rather than assumed. Table D.1 gives the verdict and the binding criterion for each, and every criterion binds somewhere, so none is redundant.

Table D.1: Pairing verdict for every operation, with the deciding criterion (P pairability, S separability, G gain).

Operation	P	S	G	Repr.
$QK^\top$, SM·V	✓	✓	✓	complex
Full-width bootstraps	✓	✓	✓	complex
Softmax partial bootstrap	✓	✓	✓	complex
Pt×Ct matmuls	✓	×	–	real
Polynomial nonlinearities	✓	×	–	real
LayerNorm partial bootstrap	×	–	–	real
Residual adds, rotations	✓	✓	×	real

The measured gain. Table D.2 puts each paired operation against its unpaired execution on the same card, the matmuls as isolated blocks at the deployed batch and the bootstraps in place in the layer. Each pair runs under the configuration of its own measurement run, so the ratio rather than the seconds is the comparable quantity. An isolated comparison of 768 real against 384 paired bootstraps gives $2.00\times$ with a 0.29% pairing overhead. The unpaired remainder of the layer reads the same with pairing on or off, 192 against 197 s, within that run's noise. The softmax partial bootstrap shares the same paired refresh without a separately isolated measurement.

Table D.2: Paired against unpaired execution per operation, RTX PRO 6000.

Operation	Unpaired (s)	Paired (s)	Gain
$QK^\top$	41.7	22.3	1.87×
SM·V	16.1	8.3	1.94×
BTS#1	247.8	125.5	1.97×
BTS#2	249.9	127.4	1.96×

The bootstrap magnitude condition. Packing two reals a and b as $a + bi$ hands the bootstrap an operand of magnitude $\sqrt{a^2 + b^2}$, up to $\sqrt{2}$ times the larger of the two. That can push the operand past the range where the bootstrap's polynomial approximation of modular reduction stays valid, so we check the bound at each paired site. At the first full-width bootstrap the operand is already held well inside the domain and the paired magnitude stays inside it. The second does not clear the bound on its own, so its operand is divided by a fixed scale before the bootstrap and multiplied back afterwards. That brings the paired magnitude into the domain and consumes no level. The scale is a client-server shared value among the build flags of the Code and Data Supplement (Appendix E).

Validation. The two ciphertext–ciphertext matmuls return results byte-for-byte identical to those of their real-native execution, since the conjugation recovers each lane exactly and the pairing introduces no approximation. The bootstraps cannot be byte-identical, because pairing changes the circuit the refresh executes, so we validate them on their output, where the paired and unpaired refreshes agree to a cosine of 1.0 at the precision we measure.

Relation to prior work. The mechanism the paired bootstraps use, packing two real ciphertexts into one complex, refreshing them together, and splitting them back by conjugation without spending a level, is established, and the main text cites it at the point of use (Bossuat et al., 2021; EPFL-LDS, Tune Insight SA, 2024). What the representation decision contributes is the procedure around that mechanism and the substrate it runs on. Bossuat et al. (2021) merge the two halves of a single ciphertext inside one bootstrap, rather than pairing two independent ciphertexts of a pipeline. Lattigo's routine does pair two independent reals, but in the conjugate-invariant ring and as a general-purpose utility with no notion of where in a network the pairing repays its cost. THOR (Moon et al., 2025) pairs at its bootstrap as well, but it is complex-native throughout, so its imaginary lane already carries a second matmul diagonal and the pairing is a byproduct of a cost it pays at every operation. OBELISK is real-native, so the imaginary lane is idle wherever the criteria admit a pairing, and identifying those sites is what the criteria do.

E Measurement Methodology

This appendix backs Section 4.1. The two comparison tables count differently. The # ct column of Table 2 is the logical ciphertexts one layer's operation processes, which HRC pairing halves before they reach the hardware, while B' of Table 1 counts the physical bootstraps one input costs across all twelve layers.

Measurement hosts. Every number the main text reports was measured on one of the first four GPUs of Table E.1. The cumulative ladder of Table G.3 was measured on the fifth, which first repeated the deployed configuration and reproduced the other RTX PRO 6000's rotation and key-switch counts to the digit and its steady-state peak to 0.01 GiB. The fifth host has also completed the full 12-layer cascade in one process at 294 s per layer, flat across the layers to $\pm 0.6\%$, every layer binding at $QK^\top$ at a 21.85 GiB live peak inside the band of Table C.2. The main text's RTX PRO 6000 column is measured on the power-capped Max-Q variant of the same card and is therefore conservative on evaluation time. Peak memory follows the level schedule and the ciphertext count rather than the device (Table C.2), and Table E.2 extends the agreement operation by operation: every row agrees to within 0.06 GiB with $QK^\top$ binding on all three devices. Evaluation time is device-dependent and is measured separately on each card. Under an earlier 26-key configuration the same code also ran the full four-task cascade on an RTX 3090 (24 GiB), with all 48 divergence gates passing and layer 0 byte-identical across two machines with that GPU. It carried the full per-operation profile on an L40S (46 GiB) at a 22.80 GiB live peak. Both are figures of that configuration rather than of the deployed one, whose peak is 21.63 GiB. The deployed configuration has since been run on 24 GiB cards directly, an RTX 3090 Ti (sm_86) at a 21.62 GiB live peak and a stock RTX 4090 at 21.68. Each ran from a fresh unpack of the Code and Data Supplement with its own provisioning and key generation, and each left over 2.3 GiB of headroom with no divergence.

Table E.1: The five measurement hosts, the RTX PRO 6000 WS also sm_120.

GPU	GiB	Host CPU	CUDA / OS
H200 (sm_90)	141	Xeon Platinum	13.2 / 24.04
A100 (sm_80)	80	EPYC 7763	13.2 / 24.04
RTX PRO 6000 (sm_120)	96	TR PRO 9965WX	12.8 / 24.04
RTX 4090 (sm_89)	24	Xeon Silver 4310	12.4 / 22.04
RTX PRO 6000 WS	96	i9-14900K	13.2 / 24.04

Why the batch is 256. The batch is the slot count the security parameters fix, not an amortization choice: at $N = 2^{16}$ a ciphertext carries 2^{15} slots, and 256 sequences of 128 tokens fill one exactly. A batch-1 run allocates the same ciphertexts, executes the same operations, and takes the same wall-clock while leaving 255 of every 256 slots empty. The setting this suits is one data owner scoring many records under one key, since separate owners cannot share a key without sharing the ability to decrypt. Choosing a different batch is a repacking rather than a redesign. Any power-of-two batch b with $T_{\text{seq}} \cdot b = 2^{15}$ fills the same ciphertexts. Only the interleaving stride and the stride-b rotation key of Appendix G follow it, and both are regenerated at key generation. The budget, the operators, and the peak are untouched, since the 768 hidden-dimension ciphertexts set them rather than the

Table E.2: Per-operation live peak (GiB), A100 and RTX 4090 layer-0 profiles against the main text's H200 column.

Operation	H200	A100	RTX 4090
Q/K proj	20.90	20.94	20.88
$QK^\top$	21.63	21.67	21.61
softmax	21.18	21.21	21.16
SM·V	20.40	20.44	20.38
W_O	19.22	19.26	19.20
BTS#1	19.86	19.90	19.85
LN1	19.25	19.25	19.19
FC1	20.83	20.87	20.82
GELU	21.26	21.30	21.24
FC2+accum	19.49	19.53	19.47
LN2-pre	8.58	8.62	8.58
BTS#2	19.87	19.90	19.87
LN2-post	19.99	19.97	19.93
Layer peak	21.63	21.67	21.61

Table E.3: Determinism and dispersion checks behind the reporting rules.

Axis	Protocol	Result
Machines	per-layer cosine and correct counts, six hosts	6 digits, counts exact
Keys	full cascade under 8 key sets, narrowest-margin batch	same $235/256$, no divergence
Dispersion	65 layer times, 6 runs (RTX PRO 6000)	447.9 ± 2.8 s (0.63%)
Probes	profiling overhead across the configuration span	12–21 s constant over 448–2292 s
Clock	A100 layer times, fixed workload	two bands $\sim$660 / 723 s
Boots	3 warmup + 12 timed, median	fit re-run in 10 fresh processes

batch. Wall-clock and operation counts are likewise b-independent, so the per-input cost scales as $1/b$. A batch below the slot-filling 256 saves no time and only idles slots, and a larger one runs as additional ciphertext sets at the same per-input cost. A longer sequence trades against the batch on the same slots, T_{seq}=256 halving b to 128 and doubling the per-input cost.

Instruments and reporting. Every time reported is a device-synced host interval, not a CUDA event trace: each timed region is bracketed by a device synchronization and measured with the host steady clock, so it carries the region's full cost. Peak memory is the live per-process footprint, the NVML non-pool allocation plus the CUDA pool high-water mark, with reserved-but-reclaimable pool memory recorded separately and excluded, since the allocator returns it on demand and does not bound the device a run fits on. In the main text's per-operation table each row is timed in place while the total is a separate end-to-end measurement, the sub-1% remainder being the glue operations, residual additions and modswitches, that the main text names. Accuracy and peak memory are deterministic, since the bounded CKKS noise never changes an argmax. Each accuracy cell is therefore one full-validation pass, each peak one measurement, and no seed control is needed, with the checks behind this and behind the timing protocol in Table E.3. Evaluation time is the only quantity that varies, through the sustained GPU clock rather than the algorithm. Every reported time is therefore the uncontended floor, the fastest device-synced pass with no other compute process on the device, and the printed 32.50 s per input is such a completed pass. A slightly faster figure near 30.9 s, reconstructable only by splicing timings across the A100's two timing bands, is left unprinted.

Software and artifacts. OBELISK builds on Phantom v2.0 with bootstrapping ported from FHE-MP-CNN (pinned by commit), needing only a CUDA 12.4+ toolkit and system NTL and GMP, and every pure memory or latency optimization is on by default in one code path. The source code, the weight-folding and fixture-generation scripts, the deployed per-task calibration tables, and the run manifest with the entry point and toggles behind each table are in the accompanying Code and Data Supplement, to be released under a permissive research license on publication. The folded weights and every fixture regenerate from the public checkpoints with one command. The supplement ships the same bootstrapping port every number in this appendix was measured with, so a reproduced layer matches the reported times rather than approximating them.

F Baseline Protocol

This appendix backs Section 4.2 and Table 1. The BTS column of Table 1 is the bootstrap time within the A100 evaluation time of the same row, not an additional cost. Every memory figure the main text reports for a competing system is our own measurement of that system's released code, and every competitor evaluation time is the authors' reported value. Table F.2 gives the split per quantity, "ours" meaning measured or counted by us in the system's released code and "rep." the authors' reported value, no quantity mixing the two within a row. We take the authors' timings because secure-inference latency moves substantially with the accelerator and the thread count, so quoting each system at its authors' configuration avoids penalizing a baseline for a host we selected. Memory and key footprints lack that sensitivity and are measured directly and uniformly. NEXUS, named in the main text's related work, does not enter the comparison: its inconsistent packing formats require costly conversions that its reported performance omits, as observed by Moon et al. (2025) and Zhang et al. (2026). Euston is excluded on threat model rather than on performance. Its user draws a random mask, decomposes it by SVD, and transmits the two orthogonal factors U and H to the server in the clear, encrypting only the vector of singular values (Gao et al., 2026). The server therefore holds in plaintext a part of the input that stays encrypted here. It also reports system-wide speedups over NEXUS rather than an absolute evaluation time, and no peak memory, so no cell of Table 1 could be filled from it. EncryptedLLM (De Castro et al., 2025) likewise falls outside the table, benchmarking an encrypted GPT-2 forward pass on its own GPU-accelerated FHE implementation. That is a decoder model rather than the BERT-base encoder every row of Table 1 reports, so no cell corresponds. No baseline reports accuracy on a checkpoint matched to ours, so Table 1 compares cost alone and not an accuracy-matched frontier. Table F.1 states each exclusion against what the system reports, so the reason can be checked without reading the prose above.

Table F.1: The three non-interactive systems outside Table 1, with the ground each exclusion rests on.

System	Model	Reports	Excluded on
NEXUS (Zhang et al., 2025)	BERT-base	evaluation time	measurement scope
Euston (Gao et al., 2026)	BERT-base	speedup over NEXUS	threat model
EncryptedLLM (De Castro et al., 2025)	GPT-2	time over a CPU	model

Table F.2: Provenance of every cell of the main text's comparison.

Quantity	OBELISK	THOR	Powerformer	MOAI
Peak memory	ours	ours	ours	ours
Evaluation keys	ours	ours	ours	ours
Bootstrap count B'	ours	ours	ours	ours
HRot, KeySw	ours	rep.	rep.	rep.
Evaluation time	ours	rep.	rep.	rep.

Every mixed choice resolves against OBELISK. Where a prior work's reported value and our measurement of its published code disagree, the main text carries the figure more favorable to the prior work. Table F.3 lists them. The rotation row reads the same way: OBELISK's printed count is the algorithmic one its schedule implies, above what its instrumentation physically records, set

against MOAI's published algorithmic count rather than the physical rotations we count in MOAI's own code.

Table F.3: Quantities where the reported and the measured value disagree.

			OBELISK adv.	
Quantity	Reported	Measured	rep.	meas.
MOAI time, H200 (s)	141.3	195.1	6.6×	9.1×
Powerf. time, A100 (s)	344.3	521.3	10.6×	16.0×
Rotations vs. MOAI	$\frac{4,824}{9,648}$	$\frac{4,572}{12,762}$	2.0×	2.8×

Why the two MOAI timings differ. The gap between 141.3 and 195.1 is a measurement-scope difference. On the same run at MOAI's optimal thread count, the sum of its per-stage timers gives 3,115 s per layer while a device-synchronized end-to-end interval gives 4,175.7 s. The 25.4% gap localizes entirely to the two operations timed through an output parameter, whose measured wall-clock times are 849 and 231 s against reported 35 and 6.9. Nearly all of the untimed span is on-the-fly plaintext weight encoding, which sits outside those stage timers and cannot be precomputed at this packing, since the encoded weights would occupy tens of terabytes per layer. A block-drain probe after every stage read 0.000 s, ruling out asynchronous kernel tails. MOAI's published 141.3 sits on its own instrument, which our rerun of its code reads at 146.0 s per input, and the printed 195.1 is the device-synchronized end-to-end interval of the structured reproduction pass, 4,161.7 s per layer. The OBELISK evaluation times reported in the main text are measured the same way.

Peak memory instruments. The three peak instruments differ and the difference is bounded. OBELISK uses the per-process live footprint of Appendix E; MOAI a 10 ms device-wide sampler; Powerformer and THOR external device-wide samples with a foreign baseline subtracted, since Liberate.FHE ships its bootstrapping obfuscated. A device-wide reading exceeds a per-process one by the CUDA context, a few hundred megabytes, so the convention can overstate a competitor's peak by well under a percent, too little to move any ratio. On Powerformer we checked directly: 72.1 GiB device-wide against 70.8 per-process and 70.1 from the allocator, on the same run.

Per-system protocol. MOAI's released harness implements one encoder layer, so we ran that layer at its full packing of 256 inputs, took peak and key memory from it, and scaled only the bootstrap count to twelve layers, which is conservative, since twelve layers cannot need fewer. Evaluation keys are the resident relinearization and bootstrapping Galois allocations for MOAI (39.4 GiB), the itemized runtime key set for THOR (38.5 GiB), and the key-complete 210-key configuration for Powerformer at a measured 256.4 MiB per key (52.6 GiB). The bootstrap counts of the two Liberate systems are measured despite the obfuscated kernel. THOR executes 25 bootstraps on a baseline layer, 18 structural and 7 operator-internal, with layer 2 at 30 and layers 9 and 10 at 26, summing to the 307 per input the main text's table prints. Powerformer's 120 follows from a live level trace whose inter-bootstrap segments close with no slack.

Cross-checks. MOAI's footprint is also measured operationally. It runs out of memory on the A100 at bootstrap setup with 59.68 GiB committed, and again on the 96 GiB card at the first $QK^\top$. On the H200 it completes only at 32 OpenMP threads, one seventh of its default, peaking at 132.5 GiB, 94.4% of the card, against the 129.3 printed from the 4-thread instrumented run. A staged memory ladder on THOR, engine 5.39 GiB, keys loaded 46.46, peak 62.70 with three instruments agreeing to within 0.9, resolves its 109 GiB encoded model against its 62.7 GiB peak, the encoded weights being host-resident and streamed per layer. THOR's reported bootstrap time divided by our traced count gives 1.10 s per bootstrap, consistent with its engine. A lean-key Powerformer variant reaching a ∼24 GiB peak was measured and rejected as key-incomplete rather than reported. Our tuned build of the baseline reaches 521.3 s where Powerformer's own paper assigns 623.4 s to THOR, so the baseline was not run in a weakened configuration.

G ABLATIONS

This appendix backs Section 4.4. The two levers are separable, and Table G.1 measures them as a 2×2 factorial rather than as a cumulative ladder, so that each lever's effect is read with the other held both on and off. All four runs are one encoder layer at batch 256 on the Max-Q RTX PRO 6000 of Table E.1, on one build at the deployed 29-key configuration, with the granularities as the only difference between the tiled and untiled arms. Time is the sum of the per-operation column, peak is the per-process live footprint at the binding operation, and every run passes the layer's accuracy gate.

Table G.1: The two levers as a 2×2 factorial, one encoder layer at batch 256.

PiP	HRC	Time (s)	Peak (GiB)	HRot	KeySw
–	–	821.3	66.09	602,824	894,810
✓	–	829.3	24.32	602,824	894,810
–	✓	540.3	66.09	304,792	570,546
✓	✓	540.5	23.45	304,792	570,546

Reading the factorial. Reverting the two granularity knobs alone, with every other optimization left on, leaves every rotation, key-switch, and multiplication count identical to the digit and moves evaluation time only inside the machine's run-to-run spread. It raises the peak from 23.45 to 66.09 GiB, a factor of 2.82 from the granularity change alone. Toggling the representation lever drops rotations by $1.98\times$ and key-switches by $1.57\times$, the second smaller because relinearizations and the bootstrap circuit are counted in it and are not all paired. Evaluation time drops 1.52–$1.53\times$, one machine's figure since latency moves with the host and its sustained clock. The peaks carry the peak model's signature. Pairing trims 0.87 GiB with the granularities deployed and exactly nothing when untiled, where both arms read 66.09 GiB to the digit. The peak is a maximum of the attention and feed-forward terms, and pairing shrinks only the attention term, which binds only in the tiled regime. The HRot and KeySw columns are the instrument's per-layer primitive counts at batch 256, a different basis from the per-input algorithmic counts of Table 1, so only their ratios are comparable across the two tables.

The factorial replicates on a second architecture. The same four arms on an A100 (Ampere) reproduce every reading of Table G.1, with arm-level spreads at or below 0.17% over paired runs. The granularity toggle leaves rotation and key-switch counts identical to the digit and moves time by at most 1.1%. The representation toggle drops layer rotations $1.97\times$ and key-switches $1.56\times$ against the RTX PRO 6000's $1.98\times$ and $1.57\times$. Pairing trims the tiled peak by 0.86 GiB against 0.87 here while leaving the untiled peak untouched to the digit on both cards. Evaluation time falls 1.60–$1.61\times$ on the A100 against 1.52–$1.53\times$ on the RTX PRO 6000, the host dependence stated above. The untiled arms also expose the peak model's maximum directly, the binding operation moving from $QK^\top$ to GELU as the feed-forward term overtakes the attention term the pairing acts on.

The three base techniques. Three execution-level optimizations sit in the base of Table G.1 and of the main text's ablation, so both levers are measured against a base that already carries them. Each is exact rather than approximate, and each removes work rather than trading accuracy for it. Table G.2 collects what each removes and what it costs, and the rungs that price them are Table G.3.

Encoding-free fused plaintext–ciphertext multiply. Column packing turns a linear projection into a rotation-free accumulation,

$$Y_{h_{\text{out}}} \;=\; \sum_{h_{\text{in}}} W[h_{\text{in}}, h_{\text{out}}] \cdot X_{h_{\text{in}}}, \tag{G.1}$$

one ciphertext per hidden column and one scalar weight per term. Each term needs that scalar as a plaintext, and the plaintext is where the cost sits.

Encoding a slot vector $\mathbf{z} \in \mathbb{C}^{N/2}$ runs four stages. The canonical embedding σ evaluates a polynomial of $R = \mathbb{Z}[X]/(X^N + 1)$ at the primitive $2N$-th roots of unity, and encoding inverts it and

quantizes,

$$m \;=\; \sigma^{-1}(\mathbf{z}) \in \mathbb{R}[X]/(X^N+1), \qquad \tilde{m} \;=\; \lfloor \Delta\, m \rceil \in R, \tag{G.2}$$

with Δ the CKKS scale. The result is split over the RNS primes as $\tilde{m} \bmod q_p$, and each residue polynomial is carried to the evaluation form the multiplication consumes by an N-point number-theoretic transform. Two transforms therefore dominate an encode, the embedding at $\Theta(N \log N)$ and the forward transforms at $\Theta(\ell N \log N)$ over ℓ limbs, against $\Theta(\ell N)$ for the multiplication they feed.

Column packing makes every plaintext on this path slot-constant, because $W[h_{\text{in}}, h_{\text{out}}]$ multiplies a whole ciphertext rather than a slot range, so the vector to encode is $\mathbf{z} = w\mathbf{1}$. Both transforms then degenerate. Since σ evaluates, the constant polynomial $m = w$ has $\sigma(w) = w\mathbf{1}$ at every root, so the embedding inverts in closed form,

$$\sigma^{-1}(w\mathbf{1}) \;=\; w, \qquad \text{giving} \qquad \tilde{m} \;=\; \lfloor \Delta w \rceil \cdot X^0, \tag{G.3}$$

one integer instead of N, and exactly rather than to rounding error. A polynomial supported on X^0 alone then has

$$\widehat{\tilde{m}}\,[p][j] \;=\; \sum_{k=0}^{N-1} \tilde{m}_k\, \omega_p^{jk} \;=\; \tilde{m}_0 \;=\; \lfloor \Delta w \rceil \bmod q_p \quad \text{for every } j \in [0, N), \tag{G.4}$$

so the evaluation form is one residue per prime and nothing of size N is built, stored, or read. The multiplication reads $\widehat{\tilde{m}}$ pointwise, and a constant vector makes that a scalar scaling of each residue, which fuses with the accumulation of Eq. (G.1) into a single pass over the ciphertext.

The degeneracy belongs to the packing rather than to the weight. Under a layout that places different values in different slots $\mathbf{z}$ is not constant, $\sigma^{-1}(\mathbf{z})$ is a genuine N-point inverse, $\widehat{\tilde{m}}[p][j]$ varies with j, and both transforms and the full-size plaintext return.

OBELISK evaluates Eq. (G.1) in a single kernel that forms Eq. (G.4) per prime, scales the ciphertext's residues by it under Barrett reduction, and accumulates in registers, leaving the rescale as the only library call on the path. The result is bit-identical to the textbook recipe at the level of the RNS integer state rather than merely close, and the gate reads cosine 1.000000 at zero maximum absolute error. Per operation the two projections run $41.4\times$ and $39.9\times$ faster, taking them from 256 and 513 s a layer to 5.2 and 9.0 on the instrumented build. The count behind those seconds is one encode per term of Eq. (G.1), so FC1 alone asks the textbook route for 768×3072 of them.

A resident variant holds the folded weight in device memory, about 57 MiB a layer, and builds the residue form of Eq. (G.4) with its own kernel instead of on the host, removing a 245 MB host-to-device transfer per FC1 call. That build is deliberately not fused into the per-coefficient kernel, since the weight does not depend on the coefficient and fusing it would recompute the same value N times. Caching encoded weights instead, the usual alternative, is not available at this packing. A pre-encoded store holds ciphertext-sized plaintexts, and Powerformer's, measured, is the main text's 93.8 GiB at 7.82 GiB a layer, against the 660 MiB of folded scalars the fixture scripts regenerate for the same model.

Cumulative single-stride rotation key. Under interleaved batching the rotations $QK^\top$ needs form an arithmetic progression of step b, and rotations compose, so $\rho_{ib} = \rho_b^i$. One Galois key for stride b therefore serves every step when it is applied cumulatively, at one key-switch a rotation, for $H \cdot (T_{\text{seq}} - 1) = 768 \times 127 = 97{,}536$ key-switches across the layer. Rotating each row directly by $i\,b$ from a fresh copy of K asks instead for a step that is not a power of two, and the library answers by decomposing $i\,b$ in non-adjacent form over power-of-two component keys. That path needs 13 distinct keys and averages 2.46 key-switches a rotation, $240{,}384$ over the same 768 ciphertexts. Both compute the same map, so the comparison is exact rather than statistical, and it reads $2.46\times$ in key-switches and $2.33\times$ in $QK^\top$ time at cosine 1. The single key also replaces thirteen resident ones. The supplement keeps both paths behind an environment toggle so the comparison is reproducible.

Scale-independent encode template. Bootstrapping's linear transforms multiply by a fixed set of diagonals, about 316 of them, and re-encode each at whatever scale the ciphertext carries at that moment. An encode factors into a scale-free stage, the inverse transform of the diagonal's values,

and a scale-dependent tail of scaling, decomposition, and forward transform. Only the tail reads the scale, so the first stage is computed once per diagonal and reused at every later scale. Diagonals are keyed by a 64-bit hash of their contents, an avalanching mix rather than an additive one, because the structured double-precision patterns these diagonals carry collide under FNV-1a. At 316 diagonals the residual collision probability is about $316^2/2^{64} \approx 5 \times 10^{-15}$, and the bootstrap accuracy gate is the backstop. Caching also removes the device-to-host synchronization each encode performs to find its maximum coefficient. The refreshed output is bit-for-bit unchanged. The cost is $+0.14$ GiB of cached images and the return is -7.0% of layer time, or -15.0% on a host with a cheaper per-encode path, so the fraction is host-dependent while the mechanism is not.

Table G.2: The three execution-level optimizations in the base. All are lossless, and only the stride key changes an operation count.

Optimization	Removes	Costs	Measured
Fused Pt×Ct	the N-point encode	57 MiB a layer	$41.4\times$, $39.9\times$
Stride key	12 keys, 1.46 switches/rot	nothing	$2.46\times$, $2.33\times$
Encode template	316 scale-free stages	$+0.14$ GiB	-7.0% layer time

The cumulative ladder. Table G.3 builds that base up from a budget-only arm, one technique at a time, on two chained encoder layers at batch 256 with the deployed key set. The stride toggle is reachable only with the representation lever off, so the ladder runs unpaired with the granularities deployed, a combination the factorial shows to be count-neutral. Ratios rather than seconds are reported, since the instrumented build inflates absolute time uniformly. The peak column is a delta against the full-technique arm, the multiplication count is identical in all four arms, and every arm decrypts to cosine 0.999484 against the same reference.

Table G.3: The three base techniques added cumulatively to the budget-only arm.

Configuration	Time (rel.)	HRot	KeySw	Peak vs base
Budget-only	1.00	1.00	1.00	$+0.74$
+ enc.-free fused	0.41	1.00	1.00	$+0.74$
+ stride key	0.38	0.66	0.74	-0.14
+ diag. cache (= base)	0.32	0.66	0.74	0

Each rung moves what its mechanism predicts. The fused kernel's rung changes no operation count and takes the two plaintext–ciphertext projections from 256 and 513 seconds per layer to 5.2 and 9.0 on the instrumented build. The stride rung's entire count reduction, 316,416 rotations and as many key-switches, is exactly twice the per-layer $QK^\top$ delta, no other row's counts move, and that block alone accelerates by $2.41\times$, a host-dependent reading of the $2.33\times$ above. The diagonal cache's rung is count-neutral, costs $+0.14$ GiB of cached images, and reads -15.0% here against the -7.0% above, the same A/B on a host with a cheaper per-encode path, so the fraction is host-dependent while the mechanism is not.

H Accuracy and Calibration

This appendix backs Section 4.5 and Table 4.

Datasets. We evaluate on four GLUE tasks (Wang et al., 2019) spanning both families the encoder serves, single-sentence classification (SST-2) and sentence-pair inference (QNLI, RTE, MRPC), each scored on its full validation set with its own public fine-tuned checkpoint.

Where the gap arises. The polynomial operators account for at most one sample per task, so the encrypted gap is dominated by homomorphic noise rather than by approximation. Table H.1 restates the main text's accuracy as counts of correct predictions, the form in which a one-sample claim is checkable, the G, G-LN, and G-LN-S columns cumulatively substituting the polynomial GELU,

LayerNorm, and softmax. The unencrypted columns are deterministic, in double precision with the encrypted inverse's noise model removed, since single precision flips borderline samples. On QNLI the polynomial GELU is worth five samples more than the exact one, so the approximations are not uniformly harmful.

Table H.1: Correct predictions on the full validation set of each task.

Task	n	Baseline	G	G-LN	G-LN-S	Enc.
SST-2	872	806	805	805	805	799
QNLI	5463	5001	5006	5005	5005	4919
RTE	277	201	200	200	200	194
MRPC	408	358	358	358	358	349

Calibration. The softmax normalizer carries a per-task constant, fitted on training data alone by one of two routes. The first takes the statistic from a plaintext pass over the training set. Where that constant drives the encrypted model outside the range its inverse is valid over, the second runs a few training batches under encryption and takes the observed percentile. SST-2 alone needs the second, because the encrypted pipeline's inflation is structurally invisible to a plaintext pass and a larger plaintext calibration set does not expose it. Either route freezes a table that the full validation set is scored against with no per-batch adjustment, so no validation input or label enters the calibration. On SST-2 that yields the reported 91.63 from 799 of 872, and calibrating on 256 against 768 training examples gives the same accuracy.

Scope of the calibration. The constant is fitted per task and does not transfer. A table fitted on SST-2 pushes inputs from another task beyond the range the encrypted inverse covers, so each task of Table H.1 carries its own frozen table. The polynomial operators are likewise fitted to this model and this depth budget rather than being general-purpose approximations.

Comparison with reported accuracy. THOR reports encrypted losses of 0.80, 1.08, and 0.49 points on SST-2, RTE, and MRPC, while OBELISK loses 0.80, 2.52, and 2.21. In absolute encrypted accuracy the two are close, OBELISK higher on SST-2 and MRPC and lower on RTE. Neither comparison is controlled, since the systems fine-tune different checkpoints and our unencrypted baselines are the stronger on all three. The like-for-like statement is the main text's: a longer chain with more bootstraps has more headroom to absorb noise, and OBELISK spends that headroom on the reductions in bootstraps and keys.

I LLaMA-3-8B Generality

This appendix backs the generality claim in the Conclusion of the main text. The three decisions transfer as method, with their values re-derived rather than reused. We assembled one LLaMA-3-8B layer, hidden width 4096, grouped-query attention over 32 query and 8 key-value heads, feed-forward width $14{,}336$, under BERT's column packing at a batched prefill of 256 sequences of 128 tokens. On a rented H200 it reproduces its double-precision reference at cosine 0.997 with no diverged head. The peak forms at the first hidden-wide bootstrap, the decoder's full-width bootstrap over the width-4096 hidden tensor. There the normalized hidden tensor at the top of the chain holds roughly 69 of the 92 GiB peak, the evaluation keys and bootstrap precomputation 18, and the retained residual 4.

Per-block fidelity. Table I.1 gates each decoder block against its double-precision plaintext reference on the 8B layer-0 fixtures, SiLU and SwiGLU over all $14{,}336$ columns and the softmax over all 32 heads at its entry level $L{=}13$. Four of the five clear the same 0.999 bar the BERT blocks do. The causal softmax is the exception and is also the block that sets the budget, so its row is the one that matters.

How the schedule was derived. Table I.2 runs the same procedure on the decoder and arrives at the same $(2, 16, 5)$ through a different binding constraint. The causal softmax clears its gate at entry

Table I.1: Per-block cosine against the plaintext reference, one decoder layer.

Block	Cosine	Note
RoPE	1.000000	max err 5×10^{-6}
RMSNorm	1.000000	
SiLU	0.999991	max err 2.5×10^{-3}
SwiGLU FFN	0.999968	min column 0.99537
Causal softmax	0.9930	$0/32$ heads diverge

Table I.2: The budget re-derived for a decoder layer.

	BERT encoder	LLaMA decoder
Deepest unsplittable	attention stream, 14	causal softmax, 10
Binding constraint	stream depth	softmax entry level
Consumption D	30	29
Bootstraps B	2	2
Chain split	15 compute, 14 boot	16 compute, 13 boot
α, total levels	5, 29	5, 29

level 13 ($0/32$ heads diverging) and fails at 12 ($9/32$), which forbids removing the first hidden-wide bootstrap. The chain splits differently, the decoder moving one prime from bootstrapping to compute, leaving the chain length and hence $\log QP$ and the security argument untouched. The two bootstraps were won, not assumed. The first assembled pass needed three. Targeting the softmax output one level higher dropped the post-softmax bootstrap, and refreshing the normalization's reciprocal input ahead of the layer boundary dropped the inter-layer one. Four of the 29 levels are slack, deliberately spent on memory. Table I.3 gives the resulting per-stage schedule, with the representation and the granularity each stage runs at. The attention pairing of Appendix D transfers to this GQA attention as well, the causal mask and the shared key-value heads being orthogonal to the head-dimension pairing, measured lossless at -5.2% per layer, though the reported layer runs it in the real representation.

The normalization lands cheaper than on BERT, the width-4096 tensor paying one level while a single-ciphertext bootstrap absorbs the rest, the same idiom as in Appendix B.3. Grouped-query attention moves the lifetime optimum, since sharing keys and values across each group of four makes two heads per piece optimal against one on BERT. Pre-normalization forces the residual to survive a whole sub-block, so it is copied down to a low level before the norm rather than held beside the normalized tensor.

Scope. One layer in isolation, not a cascade and not a full model, and the evidence is tensor-level fidelity against a plaintext reference, not task accuracy. Prefill only, with no key-value cache, since generation with a growing cache needs a packing beyond the batch-filled trace used here. Causality is a plaintext mask costing one level and no bootstrap, so every masked position is still computed and roughly half the slots carry no useful work, a saving a generation packing would have to recover.

J What Each Decision Demands of the Packing Layout

This appendix backs the opening sentence of Section 3's preamble. The three decisions are posed over a packing layout but do not presuppose the column packing this paper instantiates them on, and what each one needs from a layout differs. Table J.1 summarizes.

Budget. The feasibility constraints are layout-free. Equation (1) reads the prime bit-widths and the security cap from the ring dimension and the library's table, while coverage $(L-1)B \geq D$ and the floor $L-1 \geq \delta$ read only operator depths. The cost of Eq. (4) touches the layout once, through n_{ct}, the ciphertexts one full-width bootstrap refreshes, which column packing fixes at the hidden dimension H. Another layout re-runs the same selection over the same candidate set with its own n_{ct} substituted, and nothing else in the procedure changes.

Table I.3: The deployed decoder schedule, its two retained hidden-wide bootstraps paired.

Stage	Levels	Repr.	Piece
layer input, clamped	3	real	
RMSNorm1 (front boot)	$3 \to 2$	real	1 ct
BTS#1 (hidden-wide)	$2 \to 16$	complex	
Q/K/V proj (GQA)	$16 \to 15$	real	kv-group
RoPE (Q, K)	$15 \to 14$	real	
$QK^\top$	$14 \to 13$	real	2 heads
causal softmax ($32\times$ partial BTS)	$13 \to 3$	real	per head
SM·V (V parked at 3)	$3 \to 2$	real	2 heads
W_O, residual add	$2 \to 1$	real	
BTS#2 (hidden-wide)	$1 \to 16$	complex	
RMSNorm2 (internal BTS)	$16 \to 14$	real	1 ct
SwiGLU FFN	$14 \to 4$	real	K=128
layer output, sink clamp	$4 \to 3$	real	

Lifetime. PiP needs the consuming block to decompose into pieces that can be evaluated and freed independently. Column packing supplies that decomposition without extra work, since a head owns d_h whole ciphertexts and a feed-forward chunk owns K of them, so a piece is a set of ciphertexts rather than a slot range. A layout that spread one head across the slots of a shared ciphertext would not admit head granularity, because the piece could not be released before its neighbours in the same ciphertext. The requirement is that the tiling axis fall on ciphertext boundaries, not that the layout be column-packed.

Representation. The three criteria of Section 3.3 are themselves the requirement, and two of them are properties of the layout. Pairability needs two ciphertexts the operation can pair, which column packing supplies as adjacent columns. Separability needs the two channels to return apart, which holds when the operation keeps them apart or a conjugation projects them. Net gain is a cost question and does not involve the layout. A complex-native layout fails pairability outright, having no idle lane left to fill.

Table J.1: What each decision requires of a packing layout, what column packing supplies, and what would withhold it.

Decision	Requires	Column packing	Withheld by
Budget	a value for n_{ct}	$n_{\mathrm{ct}} = H$	nothing
Lifetime	tiling on ct boundaries	head d_h ct, chunk K ct	slot-split heads
Representation	pairable, separable	adjacent columns	complex-native

What this does and does not claim. These are conditions on a layout, not a demonstration that the deployed values carry across layouts. The decoder probe of Appendix I is the one place the paper crosses an architecture, and it re-derives δ, D, and the budget rather than reusing them (Table I.2). A layout change should be read the same way, as re-running the three procedures rather than transporting their outputs.