KNOW THE COST BEFORE YOU DEPLOY: SIMULATING HE/MPC PRIVATE LLM SERVING WITH CIPHERSCOPE

Anonymous authors
Paper under double-blind review

ABSTRACT

Homomorphic encryption (HE) and secure multi-party computation (MPC) let a server run a large language model (LLM) on a user's prompt without ever seeing it. However, deploying such private LLM serving requires choosing among many cryptographic and hardware configurations, and trying each one is costly: a single encrypted inference can take minutes to hours, and cost depends on how operators are split between HE and MPC, how the conversation grows during generation, and how many users share the system. We present **CipherScope**, a simulator that predicts the latency, throughput, communication, and protected-memory requirements of a specified private LLM deployment without running its cryptography, so that configurations can be compared before any is deployed. On real MPC executions of transformer models from BERT-tiny up to LLaMA-7B dimensions, CipherScope predicts communication within 1% and the latency of complete BERT forward passes within approximately 2–12%; the 32-layer LLaMA-dimension run validates communication only. It predicts a four-user text-generation schedule replay within 2.1% at GPT-2 dimensions and captures effects that simpler estimators miss, such as time spent waiting for precomputed cryptographic material. Harder cases, such as all-HE models and LLaMA-specific layers, show larger timing errors of 15–25%, which we report. In-allocation calibration takes about three minutes in our 2PC test; once calibrated, evaluated full predictions take under a millisecond, 10^5 to 10^7 times faster than running the deployment.

1 INTRODUCTION

Homomorphic encryption (HE) and secure multi-party computation (MPC) enable inference over protected inputs under their respective security assumptions. Recent systems support transformer and large language model (LLM) workloads, including LLaMA-7B inference with MPC and hybrid execution that combines HE with MPC (Dong et al., 2025; Pang et al., 2024; Xu et al., 2025). These advances create a deployment question: given a model, a request workload, and available infrastructure, what performance can a secure serving system deliver, and what resources will it require? Answering this question requires estimating request latency and throughput alongside communication, preprocessing consumption, and memory demand. Providers must compare configurations that satisfy their security requirements and determine whether they can meet the intended service targets.

Evaluating every candidate through execution is costly. PUMA, for example, reports approximately five minutes to generate one token with LLaMA-7B under its evaluated MPC configuration (Dong et al., 2025). Deployment costs also depend on choices spanning cryptographic protocols, HE parameters, operator assignments, packing layouts, and hardware and network resources. The same configuration must then be assessed under different request lengths, arrival patterns, and concurrency levels. This combination of expensive executions and a large configuration space makes repeated measurement difficult to sustain. A useful predictor must support these comparisons at substantially lower cost than executing each candidate deployment.

Existing tools provide complementary capabilities. Plaintext LLM-serving simulators model request arrivals, batching, and KV-cache growth, but their execution models omit cryptographic resources such as preprocessing material and ciphertext levels (Agrawal et al., 2024; Cho et al., 2024).

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.

Cryptographic estimators and compiler cost models characterize communication or homomorphic computation for specified programs and backends (Ruan et al., 2025; Aharoni et al., 2020; Brynds et al., 2026). Applying these capabilities to secure serving requires more than supplying a fixed cryptographic latency to a serving simulator: the cost of each step depends on the complete operator assignment, the resource state inherited from earlier execution, and the requests executing together. These dependencies create three modeling challenges, illustrated in Figure 1.

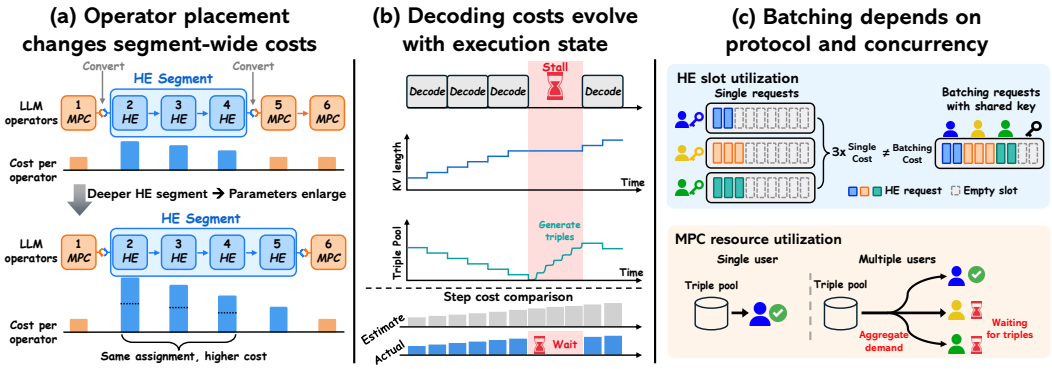


Figure 1: Schematic illustration of three effects a secure-serving predictor must capture. (a) Moving one operator into HE enlarges its segment's parameters and makes unchanged operators more expensive. (b) As decoding proceeds, the KV cache grows and the triple pool can run dry, forcing waits. (c) Batching helps only under compatible keys, and batched requests share one triple pool.

Challenge 1: Placement-dependent inference costs. As Figure 1(a) illustrates, hybrid deployments assign different operators to HE or MPC and incur conversion costs where their data representations change (Pang et al., 2024; Xu et al., 2025). An assignment change can also affect the parameters of an entire HE segment. Depending on the parameter-selection policy, increasing its multiplicative depth may require a larger ciphertext modulus, a larger ring dimension, or additional bootstrapping. Parameter changes can alter the costs of operators whose assignments remain unchanged, while bootstrapping adds maintenance work and affects subsequent level management. Fixed per-operator costs and boundary conversion costs miss these effects unless the evaluator recomputes the relevant segment parameters. Predicting a heterogeneous configuration therefore requires evaluating its assignment as a whole.

Challenge 2: Stateful autoregressive decoding. Even with a fixed assignment (Figure 1(b)), autoregressive decoding changes the work and resources required by subsequent steps. Growing KV caches increase attention work and resident storage. For protocols that consume preprocessed correlated randomness, each step draws from a finite supply of Beaver triples or FSS keys. If consumption exceeds replenishment, available material is depleted and execution must wait. HE execution similarly accumulates retained KV ciphertexts whose memory grows with context, and each step's level consumption determines where bootstrapping is needed.

Challenge 3: Protocol-constrained multi-request batching. Batching is a common throughput optimization (Figure 1(c)), but its feasibility and benefit in secure serving depend on the cryptographic protocol, packing layout, and key configuration. HE packing capacity depends on how each request's tensors occupy ciphertext slots. Combining encrypted data across distinct user keys requires a compatible cryptographic construction with its own computation and memory costs. MPC batching likewise follows protocol-specific communication and preprocessing requirements. Actual concurrency determines how much batching capacity can be used, while requests compete for shared computation, memory, and preprocessing supply. Consequently, a single-request estimate and a generic batching factor are insufficient to predict serving throughput.

We present CipherScope, a performance simulator that incorporates these dependencies into a unified prediction process. Given a model architecture, a workload trace, a per-operator cryptographic assignment, and hardware and network specifications, it predicts latency, throughput, communication, offline material consumption, and ciphertext memory. Global assignment evaluation combines MPC and HE cost engines with conversion models and evaluates HE segment parameters under the selected policy. Dual-state serving simulation jointly advances request state, including queue occu-

pancy and KV lengths, and cryptographic resource state, including preprocessing availability and retained ciphertext state. Protocol-specific batching models determine the resource demands and costs of batches formed under the workload and key configuration.

CipherScope derives structural quantities from the specified operator implementations and execution layouts, then maps them to time using calibrated backend costs. Calibration characterizes the relevant library, cryptographic parameters, and hardware environment. Once these profiles are available, the simulator evaluates candidate workloads and configurations without executing their cryptographic computations. This separation supports reuse across model architectures while retaining implementation-dependent costs and explicit resource constraints.

We evaluate CipherScope against real MP-SPDZ and OpenFHE executions, schedule replays, and measurement-based hybrid references. Across the evaluated MPC transformer-template workloads, communication error remains below 1%, including a block-wise execution at LLaMA-7B dimensions that validates communication only; complete BERT forward passes have approximately 2–12% latency error. At GPT-2 dimensions, a four-user generation replay has 2.1% makespan error and per-request latency errors within 3%. A separate two-party replay captures preprocessing waits that account for 93% of execution time, with 4.4% makespan error. Calibrated CKKS matrix-multiplication workloads have 4–7% timing error, and composed hybrid references have 2–9% error; more challenging HE and LLaMA configurations exhibit timing errors of approximately 15–27%. Beyond accuracy, CipherScope selects the measured best configuration in seven of ten decision tasks over twelve measured MPC configurations, with small regret in the remaining cases. After calibration, the evaluated full predictions take less than a millisecond, enabling rapid comparison within the supported deployment and calibration regimes.

Our contributions are:

- **Global evaluation of heterogeneous assignments.** We develop a configuration evaluator that accounts for dependencies between operator placement, representation conversions, and HE segment parameters. It derives segment parameters under a specified policy and evaluates operators under the resulting execution plan, capturing cost changes beyond the reassigned operator.
- **Stateful serving with protocol-specific batching.** We formulate serving prediction as a joint evolution of request progress and cryptographic resource availability, incorporating KV growth, preprocessing demand, and memory constraints. The current implementation supports token-granular MPC simulation and forward-granular HE/hybrid evaluation, with measured batching models for ring MPC and shared-key and per-user-key HE.
- **Implementation-aware cost modeling and calibration.** We separate implementation-specific structural quantities from calibrated timing profiles, accounting for cryptographic parameters, working-set effects, and execution context. In-context HE calibration and held-out workload evaluation establish where profiles can be reused across architectures within the supported backend and allocation.
- **Validation and configuration-selection evidence.** We evaluate execution-cost predictions, assignment-induced repricing, protocol-dependent batching, and multi-user schedule replays, distinguishing direct measurements from composed references and projections. Controlled ablations identify the effects missed by simpler assumptions, while registered predictions support low-regret selection among measured MPC configurations.

2 Background and Related Work

2.1 Private LLM Inference with HE and MPC

MPC splits tensors into secret shares across non-colluding servers; each multiplication requires communication, and dishonest-majority protocols also consume correlated randomness, such as Beaver triples or function-secret-sharing keys, generated offline (Keller, 2020; Gupta et al., 2023). HE lets one server compute on ciphertexts: in CKKS a ciphertext packs $N/2$ values, each multiplication consumes a level, the ring dimension must be large enough for the modulus to stay secure (Albrecht et al., 2022), and exhausted ciphertexts must be bootstrapped. Both now reach transformer and LLM scale, with MPC (Dong et al., 2025; Gupta et al., 2023; Hao et al., 2022; Li et al., 2024),

non-interactive HE (Zhang et al., 2024; He et al., 2025), and hybrids that convert between the two at layer or operator boundaries (Pang et al., 2024; Xu et al., 2025). These systems report the cost of the configurations they built, but do not predict configurations they did not run.

2.2 Cost Modeling for Secure Computation

Existing estimators price a single execution within one family: HawkEye and CostCO for MPC (Ruan et al., 2025; Fang et al., 2022), and CryptOracle and HElayers for HE (Brynds et al., 2026; Aharoni et al., 2020). Mixed-protocol frameworks assign MPC protocols per operation by minimizing per-operation plus conversion costs (Büscher et al., 2018; Ishaq et al., 2019; Chen et al., 2023; Blüml et al., 2026), which assumes costs decompose, and HE compilers select parameters, packing, and bootstrapping for a program (Dathathri et al., 2019; Krastev et al., 2024). Concurrently, EncFormer calibrates where to place the CKKS–MPC boundary in a transformer (Zhu et al., 2026). CipherScope instead evaluates assignments that mix MPC protocols and HE schemes, rederives HE parameters per segment because costs do not decompose, and places the result inside a serving workload; compiler-selected parameters can serve as its inputs.

2.3 LLM Serving Simulation and Secure Serving

Plaintext serving simulators model arrivals, batching, and KV-cache growth (Agrawal et al., 2024; Cho et al., 2024; Bambhaniya et al., 2024), but their operator costs have no rounds, offline material, or ciphertext levels. Secure serving has been studied for CNNs, scheduling concurrent requests around their offline phases (Garimella et al., 2023), and cross-user batching has cryptographic building blocks in multi-key HE and packed secret sharing (Chen et al., 2019; Zhang et al., 2026). No prior work predicts the cost of serving an LLM under HE or MPC, where decoding grows encrypted state, batching depends on protocol and keys, and concurrent requests share offline material.

3 Methodology

3.1 Problem Setup and Overview

A deployment is (G, α, W, H, π). The operator graph $G = (V, E)$ specifies execution order and tensor lifetimes; the implementation uses a transformer-block template (Appendix B.1). Assignment α selects each operator's MPC protocol or HE scheme, packing, and keys. Workload $W = \{(a_r, p_r, g_r, u_r)\}$ records arrival, prompt length, generation length, and user. Hardware H specifies network, memory, and offline-generation resources; $\pi = (\pi_{\text{HE}}, \pi_{\text{KV}}, \pi_{\text{B}})$ sets HE parameters, KV storage, and batching policies. Outputs include per-request TTFT, inter-token and end-to-end latency, throughput, communication, offline demand, waiting, protected memory, and bootstraps.

Figure 2 separates configuration preparation from workload simulation. The assignment evaluator produces a parameterized plan; each serving step instantiates it for the batch and KV lengths, using batching laws and calibrated costs. Generation uses client-side sampling. MPC supports token-granular steps; HE/hybrid scheduling is forward-granular. HE KV entries stay at their write level and are read without rewriting: their memory grows, but per-entry levels are not propagated across steps. Cache rewriting and refresh policies are unsupported.

3.2 Global Assignment Evaluation

To address Challenge 1, the evaluator partitions G at representation boundaries. Each HE segment σ is a maximal connected set of HE operators with subgraph G_σ. Its longest-path depth determines its parameters:

$$D(\sigma) = \max_{P \subseteq G_\sigma} \sum_{o \in P} d(o), \qquad \theta_\sigma = \textsc{SelectParams}(G_\sigma, \pi_{\text{HE}}),$$

where P ranges over execution paths, $d(o)$ is multiplicative depth, and θ_σ contains ring dimension, modulus chain, entry level, and refresh budget. Residual additions have zero depth after level

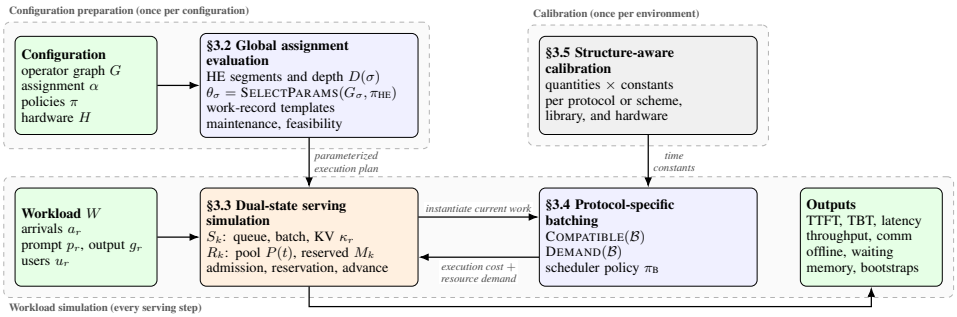


Figure 2: CipherScope prediction pipeline, in three phases. Calibration runs once per environment and turns microbenchmarks into time constants. Configuration preparation runs once per configuration: global assignment evaluation turns the operator graph, assignment, and policies into a parameterized execution plan. Workload simulation runs at every serving step: the serving simulation instantiates the plan for the current batch and KV lengths, and the batching model returns the execution cost and resource demand.

alignment. Segment parameters are selected given their boundary inputs; costs still interact through shared resources.

MPC work records count bytes, rounds, offline material, and computation; HE records count packing-derived rotations, multiplications, and live ciphertexts at their active levels. An execution-order level walk emits these records and conversions, schedules bootstraps before the budget is exhausted, and rejects operators deeper than the refresh budget. Explicit parameters bypass selection but not this walk. Conversion sizes follow adjacent HE parameters, and storage follows tensor liveness, including retained KV entries (Appendices B and B.1). Thus moving one operator can reprice its unchanged neighbors: moving LayerNorm into HE in BOLT-style BERT-base increases segment depth from 2 to 7 and doubles the library-matched ring dimension (§5.3).

3.3 DUAL-STATE SERVING SIMULATION

To address Challenge 2, at boundary t_k we track serving state S_k (queue, batch $\mathcal{B}_k$, request phases and KV lengths κ_r) and resource state R_k (offline pool $P(t_k)$ and reserved memory M_k). Background generation replenishes material at rate R_{gen} up to capacity C. On-demand generation is modeled with $C = 0$, so each step waits for its entire demand.

Each step admits arrived requests in FIFO order when the batching policy accepts them and $M_k + \mu_r \le M_{\text{cap}}$. Here μ_r reserves the peak footprint at context $p_r + g_r$, avoiding preemption within the memory model. Prefill contributes p_r tokens; decode contributes one at KV length κ_r. Instantiating the plan returns execution time τ_k and offline demand m_k, giving

$$\omega_k = \frac{\max(0, m_k - P(t_k))}{R_{\text{gen}}}, \qquad t_{k+1} = t_k + \omega_k + \tau_k + \lambda_k,$$

where λ_k is loading time not hidden by prefetching. Reservation leaves $\max(0, P(t_k) - m_k)$ material; production during execution serves later steps. Decode increments each active κ_r by one; prefill enters decode with $\kappa_r = p_r$; finished requests release μ_r. New arrivals wait for the next boundary; running batches are not re-formed. Appendix B.2 gives the full rules. For demand m per token, sustained throughput is bounded by $\min(\lambda_{\text{compute}}, R_{\text{gen}}/m)$, where λ_{compute} is the compute-limited token rate. Two requests reserving $2m$ from an empty shared pool wait twice as long as one, exposing contention absent from independent per-request estimates.

3.4 PROTOCOL-SPECIFIC BATCHING MODELS

To address Challenge 3, COMPATIBLE($\mathcal{B}$, config) checks protocol, key, and packing constraints; DEMAND($\mathcal{B}$) returns time, traffic, rounds, offline demand, memory, and execution passes. The scheduler extends the FIFO batch while it fits resource limits and is no slower than executing its packing groups separately: compatibility alone does not guarantee a gain.

For ring MPC, bytes, compute, and offline demand scale with batch size B, while rounds follow $r_0 + r_1 B$; only fixed round overhead is amortized. For HE, a key group occupying S_g slots needs $\lceil S_g/(N/2) \rceil$ passes. Shared-key requests can share a pass; per-user keys require separate groups and resident evaluation keys. Packed secret sharing (PSS) and multi-key HE extend these interfaces but remain unmeasured, with illustrative multi-key coefficients (Appendix E).

3.5 Structure-Aware Calibration

Work records are mapped to time by profiles for each protocol or scheme, library, and hardware allocation. MPC combines a communication roofline with multiplication and nonlinear costs; HE combines packing-derived work with scheduled bootstraps. Rotations depend on ring dimension and active limbs; conversions add communication and compute. Memory accounts for live tensors, offline material, evaluation keys, and $2NL \cdot 8$ bytes per two-component RNS ciphertext with ring dimension N and L active 64-bit limbs. Full formulas appear in Appendix C.

Constants come from microbenchmarks and transfer across held-out architectures within the profiled implementations and allocation, without target-model fitting. HE rotations are calibrated inside matmuls: isolated benchmarks understate their cost by 17–32% (§5.3).

4 Experimental Setup

CipherScope is a CPU-only Python simulator (about 2.5k lines). Measurements use MP-SPDZ over $\mathbb{Z}_{2^{64}}$ and OpenFHE CKKS on AMD EPYC 7702 nodes, one per MPC party, with LAN RTT 0.05 ms. OpenFHE uses eight threads; CPU allocation is fixed within experiments. Workloads span BERT forward passes, GPT-2 generation, LLaMA-dimension layers, four-user decode, and HE operators, segments, and packing.

Calibration uses synthetic matmuls and nonlinear microbenchmarks, with full models and larger shapes held out. The 2PC serving and configuration-choice tests calibrate on other shapes in the target allocation. Registered predictions precede execution; revised ones are marked. Timing uses five small-workload or three large-workload repeats after warm-up, with stated exceptions; communication is byte-identical across 82 repeats. Error is $(A - B)/B$ for prediction A and measurement B. Appendix C details workloads and procedures. Table 1 defines the evidence scope. Decode replays a preregistered schedule inside one MP-SPDZ execution; hybrid references sum separately measured HE, MPC, and conversion components. Neither measures a live integrated service.

Table 1: Evidence tiers and the claims they support. Detailed workloads and registration procedures are in Appendix C.

Evidence	Supported scope
MPC executions	Full-forward and generation costs; 32-layer LLaMA-dimension run validates communication only.
Schedule replay	Four-user per-step and per-request timing, including on-demand offline waiting; excludes live orchestration.
HE executions	Operator costs, selected-segment correctness, repricing, and shared/per-user-key packing; no multi-key implementation.
Measured choices	Selection among twelve MPC configurations, with predictions registered before target runs.
Composed references	Hybrid component costs and FIFO composition of measured batches; excludes integration overhead.
Projections	Conditional deployment scenarios, reported in Appendix G; not executed on GPUs.

5 Results

5.1 Prediction Accuracy Across Scale and Protocols

Table 2 reports measured MPC costs across model scales. Communication error is within 1%, including 0.085% on the 32-layer LLaMA-7B-dimension execution; that run executes four eight-

layer blocks and validates communication only. Complete BERT forward-pass timing errors are approximately 2–12%, and GPT-2-dimension generation is within 7.4%. These LLaMA-dimension workloads should not be confused with a complete native LLaMA execution. Predictions marked as revised include a working-set-dependent matmul term fitted on synthetic shapes. This reduces BERT-large error from -12% to -7% and gives $+1.3\%$ and $+0.2\%$ for one- and four-layer LLaMA-dimension stacks; eight layers remain at -13.2%. A fresh registered GPT-2-XL-dimension layer has $+9.5\%$ error. Across three protocols, BERT-tiny communication and time errors are within 1% and 3%, respectively (Appendix D). A layer with LLaMA-specific RMSNorm, RoPE, and SwiGLU has 1.8% communication error but 25% timing under-prediction, which remains unresolved.

Table 2: MPC forward passes, generation, and layer stacks (semi-honest 3PC, MP-SPDZ). Errors are $(A-B)/B$. †Registered before the run. ‡Communication registered before the run; time re-predicted after adding the matrix-multiplication term, whose constants were fitted on synthetic shapes only. All times are means over the measured repeats.

Workload	Comm. B	Comm. err.	Time B	Time err.
BERT-tiny forward	1.92 GB	0.03%	16.1 ± 0.8 s	+2.4%
BERT-base forward	135.1 GB	0.08%	963 ± 15 s	+6.1%
BERT-large forward†	359.3 GB	0.08%	2958 ± 18 s	−12.2%
GPT-2 generation (prefill + 8 steps)†	152.0 GB	0.08%	1071 ± 34 s	+7.4%
LLaMA-7B dim., 1 / 4 / 8 layers‡	40.9–327 GB	0.08%	458–4273 s	+1.3 / +0.2 / −13.2%
LLaMA-7B dim., 32 layers (4 blocks)†	1.31 TB	0.085%	–	–

5.2 Prediction Cost

A full prediction, which prices every operator and runs the serving loop, takes 0.025–0.26 ms: 0.025 ms for a BERT-tiny inference, 0.26 ms for a BOLT-style BERT-base inference, and 0.24 ms for the eleven-step four-user GPT-2 session, of which 0.05 ms is the serving loop alone. Relative to executing the configuration, full predictions are 10^5 to 10^7 times cheaper, for example 7×10^6 for the GPT-2 session. Calibration is the only step that runs cryptography and is paid once per protocol or scheme, library, and hardware allocation: in-allocation calibration for the dishonest-majority protocol took about three minutes including compilation.

5.3 Heterogeneous HE–MPC Configurations

HE cost depends on how rotations behave inside real workloads. Calibrated in context, CipherScope predicts CKKS matrix-multiplication stacks within 4–7%: 4% for the BERT-base HE-linear layers and 7% for BERT-tiny, which is a hold-out over input width because the rotation curve was calibrated at a single width. Calibrating from isolated rotations instead leaves an error of up to 14%, because an isolated rotation costs 17–32% less than the same rotation inside a matrix multiplication. For hybrid BOLT-style assignments, the composed references match within 2% for BERT-tiny and 9% for BERT-base, and the MPC-nonlinear and conversion components match almost exactly, so the residual comes from the HE-linear constant. An all-HE BERT-tiny with real bootstrapping reproduces the rotation and bootstrap counts exactly and confirms that bootstrapping accounts for 78% of its runtime; its time is over-predicted by 15% when a single limb count is assumed for all rotations. The HE validations above fix the ring and level explicitly, so they exercise the level walk and bootstrap rule of §3.2 but not parameter selection. We test selection separately.

Parameter selection. HE timings above use fixed parameters. For automatic selection we use a library-matched policy: the default policy disagrees with OpenFHE at 23 of 34 tested depths. The matched policy reproduces all 34 ring choices and the bootstrap ring, but uses modulus sizes queried from OpenFHE, so this is a consistency check rather than independent prediction. The policy differences and corrections are reported in Appendix F.

Repricing unchanged operators. We then run the example of §3.2 on OpenFHE. Under the library-matched policy, moving the first LayerNorm of a BOLT-style BERT-base layer into HE merges a depth-2 segment into a depth-7 segment and moves the ring from 2^{14} to 2^{15}. Calibrated on contexts of other depths and input widths, and registered before the run, CipherScope predicts that the

attention-context and output projections, whose assignment is unchanged, slow down by 26%; the measured slowdown is 51%, and an evaluator with fixed parameters predicts no change. The newly merged operators match within 4%, while the two unchanged ones are under-predicted by about 20% in the merged segment. The calibration on a shared node was not monotonic in limb count, which is the likely source of this gap. Global evaluation therefore captures the direction and about half of the repricing effect that fixed-parameter evaluation misses entirely.

Correctness of selected configurations. Finally, we execute every HE segment that the planner selects for three operator moves in BERT-base: moving the first LayerNorm, moving GELU, and, as a control that does not cross a ring boundary, moving the output projection. This gives six segments of depth 1 to 7. For each, we build the context at the planned depth with 128-bit security and let OpenFHE choose the ring: its choice equals the planned ring dimension in all six cases (2^{14} or 2^{15}). Decrypting each segment's output and comparing it with the same computation in plaintext gives a relative error of at most 4.2×10^{-9}, supporting executable depth and numerical correctness for these segments under OpenFHE's 128-bit-security configuration.

5.4 Stateful Serving and Protocol-Constrained Batching

Growing context and shared resources. Across single decode steps with KV lengths 32–4096, communication error is below 1% and timing error is 0.8–10%; six unseen KV configurations are within 0.1% and 3.8%, respectively, after separating per-row and per-element nonlinear costs. Shared preprocessing also matters: two tenants on one generation core each slow down by $1.82\times$, versus the predicted $2\times$, while spare cores remove contention ($1.02\times$). Detailed sweeps and composed BERT traces are in Appendix D.

Four-user decode. Four staggered users, a batch limit of three, and prompts of 16–32 tokens produce eleven steps, including mixed prefill/decode and queueing. Predictions and the replay schedule are registered before execution. At GPT-2 dimensions, makespan is within 2.1%, each step within 4.4%, and per-request TTFT and end-to-end latency within 3% (Table 3). The two-layer LLaMA-dimension replay retains accurate communication but under-predicts makespan by 17.1%; the nearly uniform step offset suggests an environment-dependent constant, without establishing its cause.

Under the triple-consuming 2PC protocol, the same schedule tests offline waiting inside the serving loop. Both prediction and measurement attribute 93% of makespan to triple generation; makespan error is 4.4% and communication error 0.09% on 165 GB of traffic. Because generation is on demand and pool capacity is zero, this validates waiting for each batch's material, not the depletion trajectory of a nonzero background pool. Finite-pool saturation is a projection in Appendix G.

Table 3: Continuous-batching decode with four users, registered before the run (eleven steps; n=3 measured repeats). Errors are $(A-B)/B$. The first two rows use semi-honest three-party replicated sharing, the last the dishonest-majority two-party protocol with OT-generated triples. L: layers.

Configuration	Comm.	Makespan B	Makespan	Steps	TTFT / E2E
GPT-2, 3PC (12 L)	$+0.08\%$	1796 s	-2.1%	-1.0 to -4.4%	-1.8 to -3.0%
LLaMA-7B, 3PC (2 L)	$+0.08\%$	1935 s	-17.1%	-16.1 to -19.8%	-17.0 to -22.7%
BERT-tiny, 2PC (2 L)	$+0.09\%$	534 s	$+4.4\%$	$+3.4$ to $+8.0\%$	$+3.4$ to $+6.2\%$

Batching depends on protocol and keys. Measured ring-MPC batches retain nearly constant bytes and compute per request: on a LAN BERT-tiny takes 13.2 s per request at batch one and 13.1 s at batch eight. Thus round amortization alone need not improve throughput. HE instead shares ciphertext slots within a key group. Table 4 summarizes the OpenFHE test at $N = 2^{15}$.

The memory observation was used to calibrate the per-key term; it is not a held-out memory test. Absolute HE timing remains imperfect: calibration in a separate process was 20% slower, and the 22–42-request range grew from 8.4 to 12.6 s despite using the same number of ciphertexts. The model captures the pass boundaries but not this within-pass occupancy effect. PSS and multi-key batching remain unmeasured.

Table 4: Measured HE packing behavior. The model captures pass boundaries and key grouping, but absolute timing errors reach 27%.

Configuration	Observation
Shared key, 1–21 requests	One pass takes 3.8–4.1 s; additional passes begin at 22 and 43 requests, as predicted.
Eight distinct user keys	33.7 s versus 4.0 s for eight requests sharing a key.
Additional user key	47.0 MiB: 39.0 MiB of evaluation keys plus two 4.0 MiB ciphertexts.

5.5 Value over Simpler Predictors

Table 5 replaces one mechanism at a time, retaining the same calibration data and measurements. Fixed operator costs miss repricing; idealized MPC batching under-predicts makespan by 29% (individual steps by up to 68%). Ignoring HE packing or key separation produces large errors in opposite directions, while omitting offline waiting misses 93% of the 2PC session. Each mechanism is therefore necessary on the workload that exercises it.

Table 5: Replacing one mechanism with a simpler assumption, on the same measurements. Except for the first row, entries are errors $(A-B)/B$. The first row reports predicted slowdown against a measured $+51\%$, not relative prediction error.

Mechanism removed	Simpler assumption	Simpler	CipherScope	Metric
Parameter re-derivation	fixed per-operator cost	0% vs. $+51\%$	$+26\%$	slowdown of unchanged operators
MPC batching law	batch costs one request	-29%	-2.1%	GPT-2 four-user makespan
HE slot packing	single request $\times$ batch	$> +1200\%$	$\pm 27\%$	shared key, 21–64 requests
HE key configuration	per-user requests merge freely	-86%	$+15\%$	eight per-user keys
Shared offline pool	online cost only	-93%	$+4.4\%$	two-party session makespan

5.6 Choosing Configurations

We test whether prediction accuracy supports useful choices among twelve measured MPC configurations: three semi-honest protocols, batches of one or four requests, and prompts of 32 or 64 tokens at BERT-tiny dimensions. Protocols have different trust requirements; this comparison is conditional on the candidates being acceptable to the deployment. Predictions are registered after in-allocation calibration and before target runs; the decision-task definitions were recorded beforehand.

Across ten tasks involving latency, throughput, and communication budgets, the predicted choice is the measured best in seven and satisfies the measured target in all ten. The other three choices lose 1.7% of throughput in one case and less than 0.01% of communication in two. Ranking agreement is Kendall $\tau = 0.91$ for latency per request and 0.88 for communication. These results support selection within this candidate set: 3PC is over $2.5\times$ faster than the alternatives, so the misses concern near-ties between batch sizes rather than difficult protocol tradeoffs. Appendix D.3 reports every decision. Conditional HE/MPC GPU comparisons are kept separate as projections in Appendix G.

6 Conclusion

CipherScope is a performance simulator for heterogeneous HE/MPC LLM serving. It combines global assignment evaluation, stateful resource modeling, and protocol-specific batching. After calibration, predictions of CipherScope take less than a millisecond, enabling rapid comparison without executing each candidate deployment.

AI USE STATEMENT

We used AI-assisted tools to improve the clarity, organization, and language of the manuscript, to assist with literature discovery during the early stages of the research, and to support the development and debugging of experimental code. All literature references, technical claims, and AI-assisted code were independently verified and reviewed by the authors. The authors revised all AI-generated suggestions as needed and take full responsibility for the final manuscript, including its accuracy, originality, experimental results, and citations.

REPRODUCIBILITY STATEMENT

We provide an anonymous code repository containing the CipherScope simulator, calibration scripts, experimental configurations, and scripts for reproducing the reported results. Section 4 describes the experimental setup, while Appendices B–G provide implementation details, calibration procedures, workload configurations, additional validation results, and assumptions used in the deployment studies. These materials are intended to enable reproduction of the simulator predictions and the corresponding evaluation.

ETHICS STATEMENT

This work aims to make privacy-preserving LLM serving easier to deploy. It involves no human subjects or personal data. CipherScope predicts performance only; it does not assess, weaken, or certify the security of any protocol, and a configuration it ranks highly must still meet the deployment's security requirements on its own. We state the security settings of every validated configuration and report the cases where the simulator's predictions are less accurate, so that its results are not mistaken for guarantees.

REFERENCES

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav S Gulavani, Ramachandran Ramjee, and Alexey Tumanov. Vidur: A large-scale simulation framework for llm inference. *Proceedings of Machine Learning and Systems*, 6:351–366, 2024.

Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Guy Moshkowich, Dov Murik, et al. Helayers: A tile tensors framework for large neural networks on encrypted data. *arXiv preprint arXiv:2011.01805*, 2020.

Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. In *Protecting privacy through homomorphic encryption*, pp. 31–62. Springer, 2022.

Abhimanyu Bambhaniya, Ritik Raj, Geonhwa Jeong, Souvik Kundu, Sudarshan Srinivasan, Suvinay Subramanian, Midhilesh Elavazhagan, Madhu Kumar, and Tushar Krishna. Demystifying ai platform design for distributed inference of next-generation llm models. *arXiv preprint arXiv:2406.01698*, 2024.

Jannis Blüml, Moritz Huppert, Nora Khayata, Joachim Schmidt, and Thomas Schneider. {SING}: Improving the efficiency of {MPC} protocol assignment using graph neural networks. In *35th USENIX Security Symposium (USENIX Security 26)*, pp. 2923–2942, 2026.

Cory Brynds, Parker McLeod, Lauren Caccamise, Asmita Pal, Dewan Saiham, Sazadur Rahman, Joshua San Miguel, and Di Wu. Cryptoracle: A modular framework to characterize fhe. In *2026 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 335–347. IEEE, 2026.

Niklas Büscher, Daniel Demmler, Stefan Katzenbeisser, David Kretzmer, and Thomas Schneider. Hycc: Compilation of hybrid protocols for practical secure computation. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 847–861, 2018.

Edward Chen, Jinhao Zhu, Alex Ozdemir, Riad S Wahby, Fraser Brown, and Wenting Zheng. Silph: A framework for scalable and accurate generation of hybrid mpc protocols. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 848–863. IEEE, 2023.

Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 395–412, 2019.

Jaehong Cho, Minsu Kim, Hyunmin Choi, Guseul Heo, and Jongse Park. Llmservingsim: A hw/sw co-simulation infrastructure for llm inference serving at scale. In *2024 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 15–29. IEEE, 2024.

Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. Chet: an optimizing compiler for fully-homomorphic neural-network inferencing. In *Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation*, pp. 142–156, 2019.

Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, Wen-Guang Chen, et al. Puma: Secure inference of llama-7b in five minutes. *Security and Safety*, 4:2025014, 2025.

Vivian Fang, Lloyd Brown, William Lin, Wenting Zheng, Aurojit Panda, and Raluca Ada Popa. Costco: An automatic cost modeling framework for secure multi-party computation. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pp. 140–153. IEEE, 2022.

Karthik Garimella, Zahra Ghodsi, Nandan Kumar Jha, Siddharth Garg, and Brandon Reagen. Characterizing and optimizing end-to-end systems for private inference. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 89–104, 2023.

Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. Sigma: Secure gpt inference with function secret sharing. *Cryptology ePrint Archive*, 2023.

Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. *Advances in neural information processing systems*, 35:15718–15731, 2022.

Zhiyu He, Maojiang Wang, Xinwen Gao, Yuchuan Luo, Lin Liu, and Shaojing Fu. Ensi: Efficient non-interactive secure inference for large language models. In *2025 44th International Symposium on Reliable Distributed Systems (SRDS)*, pp. 83–94. IEEE, 2025.

Muhammad Ishaq, Ana L Milanova, and Vassilis Zikas. Efficient mpc via program analysis: A framework for efficient optimal mixing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1539–1556, 2019.

Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 1575–1590, 2020.

Aleksandar Krastev, Nikola Samardzic, Simon Langowski, Srinivas Devadas, and Daniel Sanchez. A tensor compiler with automatic data packing for simple and efficient fully homomorphic encryption. *Proceedings of the ACM on Programming Languages*, 8(PLDI):126–150, 2024.

Zhengyi Li, Kang Yang, Jin Tan, Wen-jie Lu, Haoqi Wu, Xiao Wang, Yu Yu, Derun Zhao, Yancheng Zheng, Minyi Guo, et al. Nimbus: Secure and efficient two-party inference for transformers. *Advances in Neural Information Processing Systems*, 37:21572–21600, 2024.

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)*, pp. 4753–4771. IEEE, 2024.

Wenqiang Ruan, Xin Lin, Ruisheng Zhou, Guopeng Lin, Shui Yu, and Weili Han. {HawkEye}: Statically and accurately profiling the communication cost of models in multi-party learning. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2673–2691, 2025.

Tianshi Xu, Wen-jie Lu, Jiangrui Yu, Yi Chen, Chenqi Lin, Runsheng Wang, and Meng Li. Breaking the layer barrier: Remodeling private transformer inference with hybrid {CKKS} and {MPC}. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2653–2672, 2025.

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. *Cryptology ePrint Archive*, 2024.

Qinghui Zhang, Xiaojun Chen, Yansong Zhang, and Xudong Chen. High-throughput and scalable secure inference protocols for deep learning with packed secret sharing. *arXiv preprint arXiv:2601.13041*, 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*, 2026.

A Limitations and Future Work

Timing profiles depend on library, parameters, hardware, and CPU allocation. Transferring eight-core constants to four-core allocations shifted timing by 16–43%; in-allocation recalibration restored 3% accuracy in our tests. Shared-node variability may contribute to the gap between predicted (+26%) and measured (+51%) HE repricing, but does not establish its cause. The 25% under-prediction for the LLaMA-specific operator set is unresolved and provides no lower-bound guarantee. Calibration uncertainty and exclusive-node repricing tests are priorities; Appendix F gives detailed error analysis.

Validation covers costs and schedule replay rather than a complete deployed service. Hybrid references compose separately measured components; decode replays omit live network arrivals and independent-process contention. An earlier driver added about 9 s of orchestration per batch. Token-granular serving is implemented only for MPC, and fixed-level HE KV storage excludes cross-step cache rewriting or refresh. The on-demand 2PC test validates offline waiting, while depletion of a nonzero background pool remains simulation-only. KV and HE key-memory measurements support parts of the memory accounting, but capacity-constrained admission has not been tested. PSS and multi-key HE remain unmeasured extensions. Finally, the configuration-choice results cover a small MPC candidate set; the GPU studies in Appendix G are conditional projections, not measured deployment recommendations.

B Default Policies and Profiles

Table 6 lists the default policies used by CipherScope and the library-matched HE policy used for OpenFHE. They are instantiations of the interfaces in §3, not general laws of HE or MPC; the default HE policy chooses a ring that is too small for OpenFHE at 23 of 34 depths (§5.3), which is why the policy is a replaceable profile.

B.1 Execution Template and Work Records

The current implementation instantiates the operator graph from a fixed transformer block template: QKV projection, attention scores, Softmax, attention context, output projection, LayerNorm, FFN up-projection, GELU, FFN down-projection, and LayerNorm, with residual additions and KV reads attached to this chain. Tensor metadata records producers, consumers, and last use; KV entries persist across decode steps.

The evaluator then walks each segment in execution order with a level counter ℓ and emits work records. An MPC operator's record holds its online bytes, rounds, offline material, interactive multiplications, and multiply-accumulate count from the protocol's per-primitive formulas, including fixed-point truncation of every product. An HE operator's record holds its rotations, derived from the packing layout, its ciphertext multiplications, its depth, and its live ciphertexts at $L_\sigma+1-\ell$ active limbs. Before an operator o, three cases arise. If $d(o)$ exceeds the refresh budget, no single bootstrap can make o executable, and the configuration is marked unsupported unless o's internal circuit is split into smaller stages. Otherwise, if $\ell + d(o)$ exceeds the remaining budget, a bootstrap is scheduled and ℓ is reset. Otherwise o is recorded and ℓ increases by $d(o)$. Each boundary contributes a conversion record whose ciphertext size follows the adjacent segment's parameters. When the deployment fixes the parameters explicitly, SelectParams returns them unchanged and only the level walk applies.

B.2 Serving-Step Semantics

The following expands the state transition in §3.3.

The pool produces material continuously at rate R_{gen}, including while the system is idle, up to capacity C. A library that generates material only on demand, with no background producer, corresponds to $C{=}0$: every step then waits for its whole offline demand. One step then applies the following rules.

1. *Admission.* Requests with $a_r \le t_k$ are considered in FIFO order. A request r joins if π_{B} accepts $\mathcal{B}_k \cup \{r\}$ (§3.4) and $M_k + \mu_r \le M_{\text{cap}}$, where μ_r is the request's peak protected footprint at

Table 6: Default policies and profiles. q: per-level prime size; q_{sp}: key-switching special prime; w: maximum modulus in the bootstrapping regime; q_{boot}, q_{ks}: bits reserved for bootstrapping and key switching.

Policy	Default rule
Depth rule $d(o)$	Linear operator (ct-pt or ct-ct matrix multiplication): 1. Polynomial nonlinear of degree δ: $\lceil \log_2(\delta+1) \rceil$, with $\delta = 6$ for Softmax and 4 for GELU and LayerNorm. Residual addition: 0, after aligning to the lower operand level. TFHE nonlinears: evaluated by programmable bootstrapping, depth 0.
Candidate rings	$N \in \{2^{10}, \ldots, 2^{16}\}$ with 128-bit bounds $\log Q_{\max}(N) = 27, 54, 109, 218, 438, 881, 1761$ bits (Albrecht et al., 2022); $q = 50$, $q_{\text{sp}} = 60$.
SELECTPARAMS	Leveled if some candidate satisfies $D(\sigma)\, q + q_{\text{sp}} \le \log Q_{\max}(N)$: smallest such N, entry level $L_\sigma = D(\sigma)$. Otherwise bootstrapping regime at $N = 2^{16}$, the smallest ring whose 128-bit bound (1761 bits) admits the bootstrapping modulus $w = 1552$, with refresh budget $l_0 = \lfloor (w - q_{\text{boot}} - q_{\text{ks}})/q \rfloor = 10$ and $q_{\text{boot}} = 742$, $q_{\text{ks}} = 305$. An earlier version paired $w = 1552$ with $N = 2^{15}$, whose bound is only 881 bits; that combination is not secure and was corrected.
Library-matched (OpenFHE)	Security checked on $\log_2(QP)$, including the key-switching modulus, against OpenFHE's ternary-secret table (438, 881, 1747, 3523 bits for $N = 2^{14}$–2^{17}); modulus sizes per depth as reported by the library; Chebyshev depth 4 for degree 3–5 and 5 for degree 6–13; bootstrapping at $N = 2^{17}$. Reproduces OpenFHE's choice at all 34 tested depths.
Maintenance schedule	Lazy: a bootstrap is placed before the first operator whose depth would exceed the remaining budget. An operator deeper than the refresh budget is reported unsupported.
KV cache policy π_{KV}	Fixed-level layout: an entry is stored at the level at which it is written and read without rewriting. Rewrite-on-read and refresh-at-position policies are unsupported.
Batching policy π_{B}	FIFO; batch-size limit; peak-footprint memory reservation; merge only if not slower than executing the batch's packing groups separately.

context $p_r + g_r$. Reserving the peak at admission keeps later KV growth within the modeled capacity, so this policy requires no preemption; admission accuracy against a real memory limit remains untested.

2. *Instantiation.* The plan is instantiated for the batch: a prefilling request contributes p_r tokens and a decoding request one token at KV length κ_r. The batching model returns the execution cost τ_k, traffic, memory, and offline demand m_k of the step.
3. *Reservation.* The step reserves m_k when it starts. It waits $\omega_k = \max(0,\, m_k - P(t_k))/R_{\text{gen}}$, after which the pool holds $\max(0,\, P(t_k) - m_k)$. Material produced while the step executes is available only to steps that start later.
4. *Execution.* The next boundary is $t_{k+1} = t_k + \omega_k + \tau_k + \lambda_k$, where λ_k is storage-loading time that prefetching cannot hide behind communication.
5. *Advance.* Each decoding request's κ_r increases by one; prefilling requests move to decode with $\kappa_r = p_r$; finished requests depart and release μ_r. Requests arriving during (t_k, t_{k+1}) wait for the next boundary, and a running step is never re-formed.

A step is either one decode iteration under continuous batching or, for non-preemptible execution, one complete forward pass of the batch; both modes use the same pool and memory rules. In the current implementation, the token-granular mode is instantiated for MPC backends, and HE and hybrid assignments use the forward-granular mode.

As a check, rule 3 implies a sustained-throughput ceiling of $\min(\lambda_{\text{compute}}, R_{\text{gen}}/m)$ for offline demand m per token, which the simulated saturation throughput must approach. The same rule explains contention among tenants: two requests batched together reserve $2m$ from one pool, doubling the wait of a single request when the pool is empty.

Table 7: Validation workloads. L: layers; d: model width; seq: prompt tokens.

Workload	Shape	Backend	What is measured
BERT-tiny forward	L=2, d=128, seq 32	3 MPC protocols	comm, time (n=5)
BERT-base / BERT-large forward	L=12/24, d=768/1024, seq 64	3PC	comm, time (n=3)
Decode step	d=128, KV 32–4096; d=768, KV 128–1024	3PC	per-step comm, time
GPT-2 generation	prefill 64 + 8 decode steps	3PC	comm, time (n=3)
LLaMA-7B dimensions	d=4096, 1/4/8 and 32 layers	3PC	comm; time for 1/4/8 layers
CKKS matmul stacks	N=2^{15}–2^{17}, d_{in} 64–3072	OpenFHE	time
Hybrid (BOLT-style)	BERT-tiny, BERT-base	OpenFHE + 3PC	composed time
Multi-user traces	2–8 users, batch 1–8	3PC	makespan
Continuous-batching decode	4 users, GPT-2 and LLaMA-7B dim.	3PC	per-step time, comm
Decode with offline pool	4 users, BERT-tiny dim.	dishonest-majority 2PC	per-step time, offline share
HE batching by key	1–64 requests; 1–8 users	OpenFHE	pass time, key memory
Parameter selection	depth 1–34, bootstrapping	OpenFHE	ring dimension
Segment repricing	depth 2 vs. 7	OpenFHE	per-operator time
Selected parameters	six planner segments, depth 1–7	OpenFHE	ring, decryption error
Configuration choice	3 protocols × 2 batch × 2 prompt lengths	3PC, 4PC, 2PC	latency, comm
Offline pool	10^5 mults; 2 tenants	dishonest-majority 2PC	offline phase

C Calibration and Reproducibility

C.1 Workloads and Measurement Procedure

Workloads. Table 7 lists the workloads. Complete forward passes cover BERT-tiny, BERT-base, and BERT-large. Autoregressive execution is covered by single decode steps at BERT-tiny dimensions with KV lengths from 32 to 4096 and at GPT-2 dimensions with KV lengths from 128 to 1024, and by one real generation episode at GPT-2 dimensions: a 64-token prefill followed by eight decode steps in a single execution, with the KV cache growing across steps. Large-model execution is covered by one, four, and eight layers at LLaMA-7B dimensions (d=4096, d_{ff}=11008) and by all 32 layers executed as four eight-layer blocks, because a monolithic run would need about 108 GB of weight shares per party. We also run one layer with the LLaMA operator set (RMSNorm, RoPE, SwiGLU). HE workloads are CKKS matrix-multiplication stacks, an all-HE BERT-tiny with polynomial nonlinears and real bootstrapping, and the HE-linear part of BOLT-style hybrid assignments (Pang et al., 2024). Serving workloads are bursty and staggered multi-user arrival traces over BERT-tiny and BERT-base.

Calibration and hold-out. All constants come from microbenchmarks: isolated secure nonlinears at several shapes, synthetic matrix multiplications that do not match any model's shape, and, for HE, matrix multiplications at three ring dimensions and five input widths plus a limb sweep. Every full model or larger configuration is a hold-out with respect to these benchmarks. For BERT-large, GPT-2 generation, and the LLaMA-7B-dimension runs, the prediction was written to a timestamped file before the corresponding job was submitted. Communication is deterministic and byte-identical across all 82 repeated runs, so each communication value is a single measurement. Wall-clock times are the mean and standard deviation over five (small workloads) or three (large workloads) repeats after a discarded warm-up. For the multi-user decode run under the dishonest-majority protocol, the constants are instead fitted on thirteen microbenchmarks of other shapes run in the same allocation immediately before the target, with online and offline phases separated by MP-SPDZ's accounting; the library generates triples on demand, so the pool is modeled with capacity zero. The configuration-choice study uses the same in-allocation procedure for each of three semi-honest protocols, and its five decision tasks, with their latency and communication targets expressed

Table 8: Fixed model, different protocols: BERT-tiny full forward. z: time error in units of the combined standard deviation of prediction and measurement.

Protocol	Comm. B	Comm. err.	Time B (n=5)	Time err. (z)
Semi-honest 3PC	1924 MB	0.0%	16.1 ± 0.8 s	2.4% (0.4)
Malicious 3PC	14908 MB	1.0%	203.6 ± 11.4 s	2.7% (0.3)
Semi-honest 4PC	4938 MB	0.1%	41.3 ± 4.6 s	3.0% (0.2)

as multiples of the predicted values, were written to a timestamped file before any of its predictions were registered.

C.2 Timing Profiles

For MPC, time follows a secure roofline plus a compute term,

$$T_{\text{MPC}} = \max\Big(\text{rounds} \cdot \text{RTT},\ \frac{\text{bytes}}{\text{BW} \cdot \eta}\Big) + \text{mults} \cdot s_{\text{mult}} + \text{MACs} \cdot s_{\text{mac}}(\text{ws}) + n_{\text{mm}} \cdot s_{\text{mm}},$$

where η is the achieved link efficiency, the per-MAC constant is binned by working-set size to capture cache effects, n_{mm} counts matrix multiplications, and each secure nonlinear costs $r \cdot \text{rows} + e \cdot \text{elements}$ with two constants that follow its implementation. For HE, time is homomorphic compute plus scheduled bootstraps: a matrix multiplication costs $n_{\text{rot}} \cdot t_{\text{rot}}(N, L_\sigma{+}1{-}\ell)$, a polynomial nonlinear of degree δ over P slot passes costs $\delta P s_{\text{nl}}$, and each bootstrap costs $t_{\text{boot}}(N)$. Conversions add measured per-element compute to their transmitted bytes. The memory model accounts for: offline material per MPC operator and $2NL \cdot 8$ bytes per live RNS ciphertext.

Constants transfer only within the profiled backend, implementation, and hardware allocation. New operator implementations require their own structural laws or calibration; the held-out architecture results do not establish transfer to arbitrary operator sets.

D Extended Validation

D.1 Cross-Protocol Forward Passes

The same accuracy holds across protocols, which is what a deployment comparison needs. Running the full BERT-tiny forward under three MPC protocols spans a $7.7\times$ range in communication, and CipherScope, calibrated per protocol from isolated microbenchmarks, predicts all three within 1% in communication and 3% in time (Table 8). The time errors are within measurement noise: the largest deviation is 0.4 standard deviations of the repeated measurements.

D.2 KV Sweeps, Shared Generation, and Composed Traces

The per-step cost of decoding grows with the KV cache as predicted. Across single decode steps at KV lengths from 32 to 4096, communication is predicted within 1% at every point and time within 0.8–10%. Fitting nonlinear costs as $r \cdot \text{rows} + e \cdot \text{elements}$ removes an 11% over-prediction of the per-entry slope that a single per-element constant produces, and the resulting constants predict six unseen KV configurations within 0.1% in communication and 3.8% in time.

The offline pool is a binding resource on real hardware. Under the dishonest-majority two-party protocol, generating the triples for 10^5 multiplications moves $64\times$ more data than the online phase that consumes them, and generation runs at 1.4×10^5 triples per second per party. When two tenants share one generation core, each tenant's offline phase slows from 0.84 to 1.54 s, a $1.82\times$ slowdown against the $2\times$ that the reservation rule of §3.3 predicts; with spare cores the two streams do not contend ($1.02\times$). A model with an independent pool per tenant predicts no slowdown in either case.

For multi-user traces, CipherScope matches the measurement-based references within 1–9% (Table 9). These references use the same FIFO composition as the simulator, so they check propagation of measured batch costs rather than independently validating the serving scheduler or its integration overhead. The measured batches also confirm the data-parallel batching law of §3.4. Per-request

Table 9: Multi-user serving: predicted and reference makespan. C is the simulator's runtime for the whole trace. BERT-base references use the repeated single-request measurement (963 ± 15 s). *The batch-of-two reference is a single run.

Model	Trace (batches formed)	A	B	Err.	C
BERT-tiny	8 users at once [8]	109.0 s	104.4 s	4%	0.6 μs
BERT-tiny	6 users at once, max batch 4 [4, 2]	81.7 s	75.5 s	8%	0.8 μs
BERT-tiny	4 users at once [4]	54.5 s	49.9 s	9%	0.4 μs
BERT-tiny	two bursts of 4 [4, 4]	114.5 s	109.9 s	4%	0.9 μs
BERT-base	4 users, 2000 s apart [1, 1, 1, 1]	7022 s	6963 s	1%	1.2 μs
BERT-base	2 users at once [2]*	2044 s	1935 s	6%	0.3 μs

Table 10: Configuration choice from registered predictions. "3PC-B": three-party replicated sharing with batch B. Regret is the measured loss of the predicted choice relative to the measured best; every predicted choice met its target.

	32-token prompts			64-token prompts		
Task	Predicted	Measured	Regret	Predicted	Measured	Regret
Lowest latency	3PC-1	3PC-1	0	3PC-1	3PC-1	0
Throughput, tight target	3PC-1	3PC-1	0	3PC-1	3PC-1	0
Throughput, loose target	3PC-4	3PC-1	1.7%	3PC-4	3PC-4	0
Lowest comm. per request	3PC-1	3PC-4	<0.01%	3PC-1	3PC-4	<0.01%
Latency, comm. budget	3PC-1	3PC-1	0	3PC-1	3PC-1	0

bytes and compute stay constant with batch size, so on a LAN batching leaves per-request time nearly unchanged: 13.2 s at batch one and 13.1 s at batch eight for BERT-tiny, and 963 s and 968 s at batch one and two for BERT-base. Round amortization pays off only when round latency dominates.

D.3 Choosing Configurations

Accurate predictions matter only if they lead to good choices, and a small error can still reorder close candidates. We therefore let CipherScope choose among twelve configurations, three semi-honest protocols (three-party and four-party replicated sharing, and the dishonest-majority two-party protocol) with one or four requests of 32 or 64 tokens at BERT-tiny dimensions, and then measured all twelve. Predictions for each protocol were registered after an in-allocation calibration and before its configurations ran; latency errors were $+3$ to $+6\%$ for three-party, -8 to -15% for four-party, and -2 to -8% for two-party sharing, with communication within 0.2% throughout. Across the configurations, the predicted and measured rankings agree with Kendall $\tau = 0.91$ for latency per request and 0.88 for communication per request.

Table 10 reports five decision tasks for each prompt length: lowest latency, highest throughput under a tight and a loose latency target, lowest communication per request, and lowest latency under a communication budget. The predicted choice is the measured best in seven of ten tasks, and every predicted choice meets its target when measured. The three misses are near-ties between batch sizes of the same protocol, costing 1.7% of throughput in one case and less than 0.01% of communication in the other two. The protocol choice itself is unambiguous here, because three-party sharing is more than $2.5\times$ faster than the alternatives at every point; the harder part of the decision is the batch size, where batching on a LAN changes per-request cost by only a few percent.

E Batching Laws and Unmeasured Extensions

The following laws instantiate the compatibility and demand interfaces of §3.4. Ring MPC and shared-key and per-user-key HE have measured support. Packed secret sharing and multi-key HE are structural extensions; their end-to-end performance is not validated here.

The mechanisms differ in what requests share. For data-parallel MPC on a replicated or additive ring, every batch is compatible; bytes and compute scale linearly in $|\mathcal{B}|$ while rounds grow as $r_0 +$

$r_1|\mathcal{B}|$, so only the fixed round overhead is amortized. Packed Shamir sharing with pack factor k, n parties, and threshold t requires $n \geq 2(t+k-1)+1$ to multiply degree-$(t{+}k{-}1)$ sharings (Zhang et al., 2026); when this holds, a batch runs as $\lceil |\mathcal{B}|/k \rceil$ packed executions, each costing about one unpacked execution with dummies in unfilled lanes, and otherwise the effective pack factor is one. For HE, one pass over a set of ciphertexts costs T_{pass} however many of the $N/2$ slots are occupied, and requests are grouped by key: a group occupying S_g slots needs $\lceil S_g/(N/2) \rceil$ passes. A shared key puts all requests in one group. Per-user keys are compatible for co-scheduling but place each user in separate passes, so merging brings no amortization. Multi-key HE lets k keys share ciphertexts, which is compatible up to the construction's key limit, but each pass grows to $T_{\text{pass}}[\, s(1+\gamma(k^2-1)) + (1-s)k \,]$ because key switching in the linear share s grows quadratically in k and per-key nonlinear work linearly, and resident memory grows by $(k{+}1)/2$ (Chen et al., 2019). In every key configuration, each distinct key also keeps its evaluation keys resident on the server, so a batch is charged one set of keys under a shared key and k sets under per-user or multi-key keys.

Our default policy π_{B} extends the head of the FIFO queue while the batch is compatible, within its size limit and the memory reservation of §3.3, and not slower than executing its packing groups separately. The ring law is measured on MP-SPDZ, where from one to eight requests per-request bytes stay within 2% while per-request rounds fall from 8 to 4.5; the packed-sharing law is structural; and the multi-key coefficients s and γ are illustrative until calibrated against a multi-key implementation. For example, with $s = 0.5$ and $\gamma = 0.1$, merging any number of distinct keys is compatible, but the default policy stops at nine because a tenth key makes the merged pass slower than serving the keys separately.

F Measurement-Forced Corrections

Table 11 lists the structural corrections that real measurements forced on the cost model. Each initial model followed published formulas or isolated microbenchmarks; component-by-component comparison with a real execution exposed the error and its cause, and the final column records the validation scope of each correction, including held-out executions and the separately identified published-anchor comparison.

Table 11: Structural corrections forced by real measurement.

Component	Initial model	Error	Cause	After correction
MPC linear layer	resharing only	$50\times$ under	fixed-point truncation of every output dominates communication	$\sim 0\%$ comm.
MPC nonlinears	published FSS-optimized formulas	$170\times$ under	nonlinear cost depends on the implementation	0–1% comm., full-model hold-out
HE matmul	MAC-dominated compute	$60\times$ over	cost is rotation-dominated and proportional to active limbs	5% across configurations
HE rotation constant	isolated-primitive constant	8–15%	in-matmul rotations cost 17–32% more than isolated ones	4–7%, hold-out over input width
Offline key volume	weight masks per inference	98% over	fixed-operand masks are reused across inferences	34%, published-anchor tier
Nonlinear constants	one per-element constant	11% in KV slope	per-row overhead folded into the per-element constant	0.09% comm., 3.8% time, registered
Matmul time	not modeled	-36% at d=4096	per-MAC cost varies from 4.6 to 21 ns with working-set size	$+1.3\%$ / $+0.2\%$ at 1 / 4 layers; -13.2% at 8 layers; $+9.5\%$ fresh hold-out
Public multiplications (RoPE)	free	2.8% comm.	every fixed-point product with a public constant is truncated	0.1% on the RoPE on/off difference

F.1 HE PARAMETER-POLICY CHECKS

Parameter selection. We let OpenFHE choose the ring dimension itself at 128-bit security for every depth from 1 to 34 and for its bootstrapping setup, and compare with SELECTPARAMS. The default policy of Appendix B chooses a ring that is too small at 23 of the 34 depths, always by a factor of two, and 2^{15} instead of 2^{17} for bootstrapping. The bootstrapping choice also exposed an inconsistency in the default policy itself: it paired a 1552-bit bootstrapping modulus with $N{=}2^{15}$, whose 128-bit bound is 881 bits. We corrected the default to $N{=}2^{16}$ (Appendix B); the comparison above uses the policy as registered, and the corrected default still differs from OpenFHE's 2^{17}. The cause of the leveled mismatches is that OpenFHE checks security on the product of the ciphertext modulus and the key-switching modulus, which adds 120–660 bits, whereas the default policy reserves one 60-bit prime. A library-matched policy that applies OpenFHE's security table to the modulus sizes the library reports for each depth reproduces all 34 choices and the bootstrapping ring. Because those modulus sizes come from the same query, this confirms the selection rule rather than predicting it. The run also exposed a second policy difference: OpenFHE's Chebyshev evaluation of degree 3–5 consumes four levels rather than three, because it first maps the input interval onto $[-1, 1]$.

F.2 SOURCES OF ERROR

The accuracy above is the result of corrections forced by measurement. Table 11 lists eight modeling corrections, each found by comparing modeled components with measurements or published anchors; their validation scopes are listed separately. Examples include the truncation of every fixed-point product, which made the first linear-layer model under-count communication by $50\times$, and the in-context rotation cost discussed above.

Time constants, unlike structural quantities, depend on the execution environment. Communication was byte-identical across all 82 repeated runs, whereas moving eight-core constants to a four-core allocation shifted time by 16–43%, and the two-layer LLaMA-7B-dimension decode run was under-predicted by a uniform 17% with per-step standard deviations below 1%, consistent with an allocation-level offset; this observation does not isolate its cause. Shared nodes add a second effect: the HE repricing calibration ran on a node with 120 of 128 cores allocated to other jobs, its per-rotation times were not monotonic in limb count, and target repeats varied by about 10%, which may help explain why the predicted repricing (+26%) reaches only half of the measured one (+51%). One result is not explained by either effect: the single LLaMA-operator-set layer matches communication within 1.8% after the RoPE truncation correction but under-predicts time by 25%, which we have not yet resolved.

G DEPLOYMENT PROJECTIONS AND ASSUMPTIONS

Table 12 lists the assumptions behind the projections below. None of these configurations was executed on GPUs; the studies illustrate decision boundaries and should not be read as measured recommendations.

G.1 DEPLOYMENT STUDIES

Figure 3 applies these assumptions to three conditional comparisons. HE latency is the 37.3 s per-pass NEXUS anchor; MPC uses the SIGMA profile. Both are projections at published operating points, not executions on the devices compared. Shared-key slot packing and the placeholder price are essential to the conclusions.

Preprocessing supply. In Figure 3a, throughput saturates within 1.1% of R_{gen}/m; an infinite-pool model over-predicts it by $5.3\times$ at the highest load. Beyond sustainable load the queue and latency grow. This finite-pool curve is simulated. Related measured evidence is narrower: triple generation moves $64\times$ more data than the online phase, two tenants sharing one generation core slow down by $1.82\times$, and the on-demand four-user run spends 93% of its time waiting for triples (§5.4).

Batching and memory. For BERT-base with 128-token prompts (Figure 3b), shared-key HE overtakes MPC at batch 32. Under the modeled footprint, an L4 admits only 16 requests whereas a B200 admits 128, so the L4 never reaches this crossover. A single-request estimate scaled linearly with

Table 12: Assumptions of the deployment studies.

Item	Assumption
MPC operating point	SIGMA two-party FSS protocol; per-inference cost from CipherScope's SIGMA profile scaled to SIGMA's published BERT-base result; two nodes, one per party; LAN at 1 ms RTT and 10 Gb/s
HE operating point	CKKS; per-pass latency scaled to NEXUS's published BERT result (37.3 s at batch one); ring and bootstrapping from the default policy; one node
HE memory	$2NL \cdot 8$ bytes per live ciphertext; live set of four activation ciphertexts plus the KV-cache ciphertexts of all layers
Device capacity	L4 23.7 GB and B200 191.5 GB, as reported by the devices on our cluster; a batch is admissible only if its peak memory fits, with no spill to host memory
Price	3 per node-hour for every node type (placeholder); throughput per dollar divides tokens per second by the hourly cost of all nodes used
Scaling	HE per-pass latency is independent of slot occupancy up to $N/2$ slots; MPC throughput is flat in batch size on a LAN (§5.4)
Key configuration	All HE requests share one key; per-user and multi-key configurations are not used in these studies, and multi-key coefficients are illustrative
Offline supply	GPT-2 decoding under SIGMA; generation rate set to four tokens' worth of material per second; pool capacity 2 GB, initial level 0.3 GB

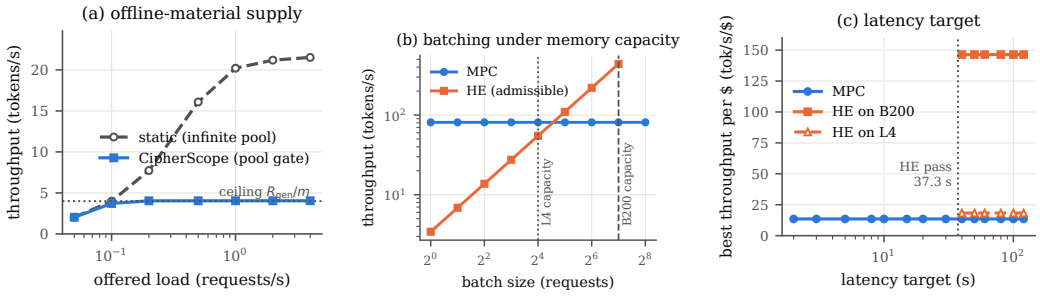


Figure 3: Resource constraints change the preferred configuration in hypothetical scenarios built on published operating points (Appendix G). (a) GPT-2 decoding under MPC: with the pool gate, throughput saturates at the analytic ceiling R_{gen}/m, while an infinite-pool model keeps rising. (b) BERT-base, 128-token prompts: HE throughput grows as requests share ciphertext slots and overtakes MPC at batch 32; HE batches are admissible only up to the device memory capacity. (c) Best throughput per dollar under a latency target: HE becomes eligible once the target exceeds its per-pass latency, and its advantage depends on how large a batch the device admits.

batch size misses the HE slot-sharing benefit. These admission limits depend on the memory model and have not been tested under an enforced device cap.

Latency target and price. Below the 37.3 s HE-pass latency, only MPC is eligible (Figure 3c). Above it, the projected HE throughput per dollar is $10.8\times$ MPC on a B200 and $1.4\times$ on an L4, at the assumed equal per-node hourly price. HE uses one node while MPC uses one per party, explaining the L4 cost advantage despite lower throughput. These ratios change with prices, key configuration, and attainable memory capacity; they illustrate the decision inputs rather than establish a preferred production backend.