PARTITION-AND-MERGE ATTENTION: SEQUENCE-AXIS OBFUSCATION FOR PRIVATE LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models are widely deployed as cloud services, making the protection of users' sensitive inputs increasingly important. Cryptographic methods provide strong privacy guarantees, but their high cost limits their practicality for large-scale LLM inference. Obfuscation-based methods retain efficient inference but largely focus on feature transformations, leaving the sequence dimension unprotected. We introduce Partition-and-Merge Attention (PMA), which adds sequence-axis obfuscation to feature-transformed inference. During prefill, a Trusted Client computes attention within the current block, while the Server computes attention over key/value rows permuted within preceding blocks. During decode, the Client accumulates newly generated key/value rows and permutes each completed window before uploading it to the Server cache. Exact merging preserves ordinary attention in real arithmetic while keeping historical attention on the Server. Experiments on Qwen1.5-MoE-A2.7B show reduced ordered prompt reconstruction under the evaluated prefill attack. Under the evaluated completed-window reconstruction attack, PMA reduces ROUGE-L F1 from 0.9813–1.0000 for attention-path adaptations of STIP and AloePri to 0.2419, while maintaining WikiText-2 perplexity close to ordinary inference. PMA adds sequence-axis obfuscation while retaining the asymptotic attention complexity of feature-based obfuscation.

1 INTRODUCTION

Protecting users' sensitive inputs is essential for deploying large language models (LLMs) as cloud services. The service provider processes users' requests and may infer private text from the information available during inference (Zheng et al., 2024; Luo et al., 2025). Homomorphic encryption and secure multiparty computation provide strong privacy guarantees, but their computation and communication costs can make interactive generation expensive (Hao et al., 2022; Pang et al., 2024). For example, NEXUS reports 51.84 s amortized latency per input to generate one output token with Llama-3-8B on four A100 GPUs (Zhang et al., 2025).[1] Obfuscation-based methods offer a more efficient route by transforming feature representations and adapting model parameters to operate on them. Systems such as STIP and AloePri report inference efficiency close to their unprotected baselines (Yuan et al., 2026; Lin et al., 2026), motivating further protection along this route.

Feature obfuscation, however, leaves a separate source of input leakage: the sequence dimension. Changing the feature coordinates of a representation does not hide which token position it represents. During prefill, the pattern of permitted attention connections can reveal token order; during decode, the order in which cached representations arrive provides a second signal. These observations expose *sequence correspondence*, the mapping between visible representations and logical token positions, helping an attacker align observations with candidate text. Simply shuffling representations is insufficient if the computation reveals their original order. STIP already notes this difficulty for causal masking (Yuan et al., 2026); here, we analyze the sequence-correspondence problem across both prefill and decode. Keeping all attention on a trusted client avoids these direct signals, but makes its workload grow with the context. Effective sequence protection must therefore retain cloud computation over the growing history.

We propose *Partition-and-Merge Attention* (PMA) to extend feature obfuscation with protection for sequence correspondence in a Trusted Client–Server setting. PMA assigns the order-revealing

[1] Reported with eight-token inputs and a batch size of 32.

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.

attention within a small group of tokens to the Client, while the Server computes attention over earlier groups whose rows have been independently shuffled. Newly generated representations are likewise collected into small windows and shuffled before upload. The Client exactly combines the two parties' partial results, preserving ordinary attention in real arithmetic. This design obscures token order within each block or window while keeping attention over the growing history on the Server. The Client's attention work per query depends on the block or window size, rather than the full context length.

Our contributions are:

1. We identify and analyze sequence correspondence leakage in obfuscated LLM inference, showing how token order remains exposed during both prefill and decode despite feature protection.
2. We propose PMA to protect against sequence-correspondence leakage in feature-obfuscated inference while preserving attention correctness and efficient cloud computation.
3. We evaluate reconstruction, language-model quality, and execution cost. On Qwen1.5-MoE-A2.7B, PMA lowers reconstruction under the evaluated attacks and retains perplexity close to ordinary inference. It also reduces request latency relative to Trusted-Client Full Attention in our same-host CPU–GPU simulation.

2 Background and Related Work

2.1 LLM Inference

Decoder-only Transformer LLMs map tokens to embeddings, process them through stacked layers, and project the final representations to next-token logits (Vaswani et al., 2017; Qwen Team, 2024b). Each layer combines self-attention, a feed-forward network, normalization, and residual connections. Attention mixes information across token positions; the feed-forward network transforms each token's features. In mixture-of-experts (MoE) models, a router selects feed-forward experts for each token, sometimes alongside shared experts (Qwen Team, 2024a).

Autoregressive inference has two stages. *Prefill* processes the prompt positions in parallel and computes the distribution for the first output token. Each layer stores its keys and values in a ***K/V cache***. *Decode* then processes one new token per step, reusing historical K/V vectors and appending the current pair instead of recomputing the history.

Self-attention uses queries Q, keys K, and values V, obtained by learned projections of token representations. For one head, scaled dot-product attention is (Vaswani et al., 2017)

$$\mathrm{Attn}(Q,K,V;M)=\mathrm{softmax}_{\mathrm{row}}\left(\frac{QK^{\top}}{\sqrt{d_h}}+M\right)V, \qquad M_{ij}=\begin{cases}0, & j\le i,\\ -\infty, & j>i,\end{cases}$$

where d_h is the head dimension and i, j are logical query and key positions. The *causal mask* M is added to the query–key scores before softmax, assigning zero attention weight to future positions. In prefill, its permitted entries form a lower triangle. In single-token decode, the cache contains only past and current positions, so all available keys are permitted. Models using rotary positional embeddings (RoPE) rotate Q/K according to their logical positions before computing these scores (Su et al., 2024). In a token-representation matrix, rows index token positions and columns index feature coordinates; these are the sequence and feature axes of obfuscation.

2.2 Related Work

Cryptographic private inference. Cryptographic protocols protect Transformer inference through homomorphic encryption, secure multiparty computation, or combinations of the two (Chen et al., 2022; Hao et al., 2022; Li et al., 2023; Pang et al., 2024; Zhang et al., 2025; Xu et al., 2025). Their efficiency improvements target linear and nonlinear operators, communication, and conversion between cryptographic representations (Hao et al., 2022; Pang et al., 2024; Xu et al., 2025; Zeng et al., 2026). Repeating protected computation during token generation makes these costs especially important for autoregressive LLMs.

Obfuscation-based and hybrid private inference. Obfuscation-based and hybrid methods reduce inference cost by using permutations or other feature transformations to protect model parameters or intermediate representations (Zheng et al., 2024; Yuan et al., 2026; Luo et al., 2025; Lin et al., 2026). STIP uses semi-symmetric permutations in three-party inference; AloePri jointly transforms inputs and model parameters. STIP also explains why a permuted causal mask reveals the sequence permutation (Yuan et al., 2026). PMA builds on feature obfuscation by protecting within-block and within-window sequence correspondence during attention computation.

Reconstruction from intermediate representations. Intermediate model representations can reveal private information even when the original text is not directly exposed. Hidden-state observations support prompt reconstruction (Thomas et al., 2025), K/V-cache contents support sensitive-input recovery (Luo et al., 2026), and differently shuffled activations can be aligned for model extraction (Li et al., 2026). Thomas et al. also introduce Cascade, which distributes sequence shards across multiple parties. PMA instead combines the attention results of one Trusted Client and one Server, adding sequence protection within a two-party architecture.

3 Sequence Correspondence Exposure under Feature Transformations

Feature transformations can obscure feature coordinates while leaving row-to-position associations visible. We consider two-party inference between a Trusted Client and a semi-honest Server that holds the model, follows the prescribed protocol, and records the information available during normal execution. This information includes transformed attention representations, the query–key pairs admitted by masked attention, and the order of K/V-cache updates. The Server can retain and analyze its complete execution transcript, including returned merged attention outputs. The Trusted Client holds the input, logical token positions, and request-specific sequence permutations.

We define *sequence correspondence* as the association between logical token positions and the sequence-indexed objects observed by the Server. In prefill, logical positions index query rows and paired K/V rows, which in turn index the rows and columns of the attention-score map. Permuting the query and K/V rows changes their physical order and reorders the attention rows and columns accordingly, while preserving these associations. In decode, cache insertion order directly associates each new K/V row with its generation step.

3.1 Causal Masking Reveals Logical Order in Prefill

For a causal block of B valid tokens, let C be the binary attention support: $C_{ij} = 1$ when the causal mask permits query i to attend to key j ($M_{ij} = 0$), and $C_{ij} = 0$ otherwise. Its row counts are $(1, \ldots, B)$ and its column counts are $(B, \ldots, 1)$. For query-row permutation Π and shared K/V-row permutation Σ, the observed support is $\widetilde{C} = \Pi C \Sigma^\top$, with $\mathbf{1}$ denoting the all-ones vector:

$$\widetilde{C}\mathbf{1} = \Pi C \mathbf{1}, \qquad \widetilde{C}^\top \mathbf{1} = \Sigma C^\top \mathbf{1}. \tag{1}$$

The counts remain distinct: sorting rows by increasing count restores query order, and sorting columns by decreasing count restores the shared K/V order. The Server only needs to determine the allowed query–key pairs; the mask need not be materialized. The same argument applies to a shorter final block using its valid length. Appendix B gives the details.

3.2 K/V-Cache Updates Reveal Logical Order in Decode

Autoregressive decode produces one new K/V pair at each generation step. If each pair is immediately added to the Server cache, cache insertion order is generation order. Buffering several rows only delays their visibility; if they are later added in the same order, their relative order remains exposed. Feature transformations do not alter this insertion order. The buffered rows must therefore be permuted before cache insertion.

The attacks in Section 5 combine this ordering information with candidate-token or K/V-state matching. These observations lead to two design choices: the Trusted Client computes masked current-block attention in prefill, and each completed K/V window is permuted before it is appended to the Server cache in decode.

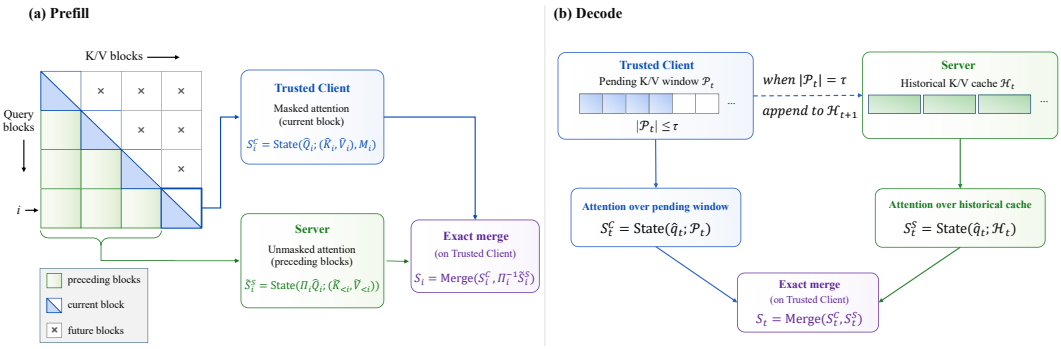


Figure 1: Overview of PMA. In prefill, the Trusted Client computes masked attention within the current block, while the Server computes unmasked attention over preceding K/V blocks using row-permuted queries and K/V. The Client restores the Server state to logical query order and merges the two partial states. In decode, the Client attends to the pending K/V window, while the Server attends to the historical cache. Completed prompt blocks and decode windows are row-permuted before their K/V rows are stored by the Server. Both decode branches use the current query $\widehat{q}_t$; only the K/V paths are shown.

4 Partition-and-Merge Attention

PMA partitions attention between a Trusted Client and a Server, then exactly merges their partial softmax states (Figure 1). The Client handles masked current-block attention in prefill and a pending K/V window in decode; the Server handles unmasked historical attention. Fresh row permutations obfuscate stored row order within each block/window, while block/window order remains visible.

PMA provides sequence protection at the attention interface; protection of the incoming hidden states and the remaining model depends on the underlying feature-obfuscation scheme. In the default attention-scope prototype, the Server sends normalized hidden states in the Ordinary feature basis to the Client. The Client uses its provisioned projection weights to form Q/K/V, applies RoPE to Q/K and the compatible feature transforms, and supplies the transformed queries and permuted K/V blocks described below.

4.1 Compatible Feature and Row Transformations

Using the attention definition in Section 2, let $Q \in \mathbb{R}^{n_q \times d_h}$ and $K, V \in \mathbb{R}^{n_k \times d_h}$ for one head, where n_q, n_k count query and K/V rows and d_h is the head dimension. The additive mask M has shape $n_q \times n_k$. We omit M for unmasked attention. Let $Q^{\mathrm{R}}, K^{\mathrm{R}}$ denote post-RoPE queries and keys. For invertible feature transforms $G, H \in \mathbb{R}^{d_h \times d_h}$, define

$$\widehat{Q} = Q^{\mathrm{R}} G, \quad \widehat{K} = K^{\mathrm{R}} G^{-\top}, \quad \widehat{V} = VH, \quad \widehat{W}_O = H^{-1} W_O. \tag{2}$$

Our implementation uses signed feature permutations. The paired query/key transforms preserve logits:

$$\widehat{Q}\widehat{K}^{\top} = Q^{\mathrm{R}} (K^{\mathrm{R}})^{\top}. \tag{3}$$

The value transform passes through the unchanged attention weights and is removed by the output projection:

$$\mathrm{Attn}(\widehat{Q}, \widehat{K}, \widehat{V}; M)\widehat{W}_O = \mathrm{Attn}(Q^{\mathrm{R}}, K^{\mathrm{R}}, V; M) W_O. \tag{4}$$

Here $W_O \in \mathbb{R}^{d_h \times d_{\mathrm{model}}}$ is the current query head's output-projection block, with model width d_{model}. In grouped-query attention (GQA), multiple query heads share one K/V head and therefore share its feature transforms G, H. Any downstream feature-basis change can also be folded into this projection.

The Trusted Client applies RoPE using logical positions before the feature and row transformations. Applying RoPE according to the permuted row indices would change the attention logits.

PMA independently permutes query rows by Π and paired K/V rows by Σ, producing $\Pi\widehat{Q}$, $\Sigma\widehat{K}$, and $\Sigma\widehat{V}$. Because the Server branch is unmasked, attention is equivariant to these row permutations:

$$\mathrm{Attn}(\Pi\widehat{Q}, \Sigma\widehat{K}, \Sigma\widehat{V}) = \Pi\,\mathrm{Attn}(\widehat{Q}, \widehat{K}, \widehat{V}). \tag{5}$$

Sharing Σ between K and V preserves every key–value pair, while Π only reorders the output query rows. Feature transformations are fixed for each layer and K/V head and reused across requests. For each request, PMA samples a query-row permutation for every layer, query head, and prompt block, and a shared K/V-row permutation for every layer, K/V head, and prompt block or completed decode window. All request-specific permutations remain on the Trusted Client.

4.2 Exact Merge of Partial Softmax States

The Client and Server attend to disjoint K/V collections, but their normalized outputs cannot be added directly. Write $\mathrm{State}(Q; \mathcal{A}, M) = (\ell_{\mathcal{A}}, U_{\mathcal{A}})$, where $\mathcal{A}$ contains paired K/V rows, $\ell_{\mathcal{A}}$ is the per-query log-sum-exp of scaled, masked Q/K scores and $U_{\mathcal{A}}$ is the normalized attention output over collection $\mathcal{A}$. We omit M for an unmasked branch.

For two partial states $S_1 = (\ell_1, U_1)$ and $S_2 = (\ell_2, U_2)$ computed for the same query rows over disjoint K/V collections, their exact row-wise merge is

$$\begin{aligned} \ell &= \mathrm{logaddexp}(\ell_1, \ell_2), \\ U &= e^{\ell_1 - \ell} U_1 + e^{\ell_2 - \ell} U_2. \end{aligned} \tag{6}$$

The exponential weights multiply each output row. We denote the result by $\mathrm{Merge}(S_1, S_2)$. In real arithmetic, the merged state is exactly the state obtained by applying softmax over the union of the two K/V collections. It follows the same log-sum-exp principle used by online and blockwise exact softmax computation (Milakov & Gimelshein, 2018; Dao et al., 2022). An empty branch has zero weight and uses $\ell = -\infty$, $U = 0$; each causal query has at least one valid key.

The Server returns the log-normalizer and partial output for each query and head. The Client restores logical query order within each head, merges the two branches, and concatenates the head outputs for each token. It returns this feature-transformed output in logical token order to the Server for output projection.

4.3 Prefill

Divide the prompt into contiguous logical blocks $\mathcal{B}_1, \ldots, \mathcal{B}_m$ of at most B tokens. For queries in block i, the Client computes the causally masked state S_i^{C} over the current block, while the Server computes the unmasked state $\widetilde{S}_i^{\mathrm{S}}$ over all preceding blocks using queries $\Pi_i\widehat{Q}_i$. Prior K/V blocks retain their independent shared row permutations. The Client restores logical query order in both components of the Server state and computes

$$S_i = \mathrm{Merge}(S_i^{\mathrm{C}}, \Pi_i^{-1}\widetilde{S}_i^{\mathrm{S}}). \tag{7}$$

The two branches cover disjoint parts of the ordinary causal-attention domain. After block i, the Client appends $(\Sigma_i\widehat{K}_i, \Sigma_i\widehat{V}_i)$ to the Server cache using a fresh shared K/V permutation. The first block has no Server-side history; a shorter final block uses only its valid rows.

4.4 Decode

At step t, the Client adds the current K/V pair to its pending window $\mathcal{P}_t$, with $|\mathcal{P}_t| \le \tau$. The Server holds historical cache $\mathcal{H}_t$: prompt blocks and completed decode windows, each stored with its independent K/V-row permutation. Both branches use the current query $\widehat{q}_t$; a single query needs no row permutation. The Client merges

$$S_t = \mathrm{Merge}\left(\mathrm{State}(\widehat{q}_t; \mathcal{P}_t), \mathrm{State}(\widehat{q}_t; \mathcal{H}_t)\right). \tag{8}$$

After a step fills the pending window, the Client prepares a freshly permuted K/V block. It is uploaded and appended before historical attention in the next Server request, and the active pending window is cleared. If generation terminates, any unuploaded rows are discarded with the request state.

4.5 Correctness and Trusted-Client Complexity

Correctness. Equations 3–5 preserve attention under the compatible feature and row transformations. In each stage, the two branches partition the ordinary causal-attention domain; restoring query order and applying Equation 6 therefore gives the ordinary attention output in real arithmetic after output projection. Appendix B provides the derivation. Changed reduction order may introduce finite-precision deviations, evaluated in Section 5.4.

Trusted-Client complexity. With prompt length T and block size B, Client prefill attention costs $\mathcal{O}(TBd_h)$ per head, versus $\mathcal{O}(T^2d_h)$ for complete causal attention. Decode attends to at most τ pending rows, costing $\mathcal{O}(\tau d_h)$ per step independently of historical-cache length. Across both parties, total attention work remains $O(T^2d_h)$ in prefill and $O(nd_h)$ per decode step with n available K/V rows, matching feature-based obfuscation: the two branches partition the same query–key pairs, and each query requires only an $O(d_h)$ merge. These are attention-only bounds; Section 5.5 evaluates system costs.

5 Evaluation

We evaluate whether an attacker can recover prompt tokens from prefill attention representations or generated continuation tokens from completed K/V-cache windows. We also measure language-model quality, numerical accuracy, end-to-end latency, and communication cost.

5.1 Experimental Setup

Models and implementation. We use Qwen1.5-MoE-A2.7B (Qwen Team, 2024a) throughout the main evaluation. All methods run in BF16 with batch size one. PMA uses a prefill block size of $B = 16$ and a decode pending-window capacity of $\tau = 16$. This checkpoint uses multi-head attention (MHA). We additionally report execution and latency results on Qwen2.5-32B (Qwen Team, 2024b), which uses GQA.

Hardware and deployment. We use one NVIDIA RTX PRO 6000 Blackwell Server Edition GPU (96 GB) and two Intel Xeon Platinum 8470Q CPUs. The intended same-host deployment places the Trusted Client in a CPU TEE and the Server on the GPU. We simulate the Client with an ordinary CPU process and measure Client computation together with CPU–GPU transfers. Deployment details are in Appendix C.

Baselines. **Ordinary** is unmodified GPU inference. **STIP-Attn** (Yuan et al., 2026) folds feature permutations into the Server attention path; **AloePri-Attn** (Lin et al., 2026) applies covariant obfuscation to that path. These attention-path adaptations retain logical sequence order and provide feature-obfuscation baselines for the attention protocol. **PMA w/o causal partition** exposes current-block causal support while retaining PMA's feature and row transforms (prefill ablation). **Trusted-Client Full Attention** keeps the entire K/V cache and computes causal attention on the CPU Client, with the remaining model on the GPU (efficiency baseline).

Datasets. Both reconstruction experiments use 30 source prompts from the WikiText-2 validation split (Merity et al., 2017); calibration subsets are specified below. Perplexity uses the complete test split.

5.2 Prefill Prompt Reconstruction

Inspired by Hidden No More (Thomas et al., 2025), we reconstruct tokens from Server-visible layer-12 attention representations. Each of 30 prompts contains a 16-token known prefix and eight target tokens, totaling 240 targets. At each position, a fixed public Ordinary model proposes its top 256 tokens from the reconstructed prefix; the closest candidate extends that prefix. All settings share the proposal model, candidate budget, layer, and scoring rule.

The score averages a Q/K/V cosine distance with a distance between attention scores over known prefix positions (Appendix D). The attack learns no method-specific inverse. Ordinary, STIP-Attn,

and AloePri-Attn retain logical row order; PMA is scored at observed row indices without additional row assignment. Only the PMA w/o causal partition ablation uses causal-support counts to restore logical query and K/V order before scoring.

Following AloePri (Lin et al., 2026), we report Text Token Recovery Success Ratio (TTRSR): the percentage of target token IDs recovered at their logical positions. This measures ordered recovery, including row-assignment and accumulated prefix errors.

Table 1: Prefill reconstruction at layer 12 over 240 target tokens.

Setting	TTRSR (%) ↓
Ordinary	85.42
STIP-Attn	58.75
AloePri-Attn	26.67
PMA w/o causal partition	30.00
PMA	**0.00**

Results. PMA reduces TTRSR to 0.00% over the 240 target tokens, compared with 85.42% for Ordinary and 26.67–58.75% for the two feature-transformation baselines (Table 1). The public proposal model includes the target among its 256 candidates at 91.25% of positions when given the correct prefix; the sequential attack uses its own reconstructed prefix.

Exposing the current-block causal support raises TTRSR from 0.00% to 30.00%. With the feature transformations and row permutations unchanged, the support counts recover the logical query and K/V order before candidate matching. The drop to 0.00% under full PMA therefore shows that causal partition removes the direct sequence-order signal used by this attack. Comparisons with Ordinary, STIP-Attn, and AloePri-Attn additionally reflect differences in their Server-visible feature coordinates.

5.3 Decode Continuation Reconstruction

Attack setup. We adapt the released Shadow in the Cache attack (Luo et al., 2026), using six prompts for calibration and 24 for evaluation. Each evaluation prompt has a 16-token known prefix and a fixed 16-token Ordinary continuation (384 targets in total). For PMA, we construct the completed cache window offline from the fixed continuation's K/V states, applying the protocol's feature transforms and per-head row permutations before reconstruction. The attack uses upload row order without estimating head-specific permutations.

The attack uses 32 candidate tokens per position and beam width four. Candidate and observed K/V states are compared in a common feature basis, so this evaluation tests reconstruction after feature alignment. K/V matching distances are scaled using the six calibration prompts, with the scales then fixed for evaluation. We report layer 23, which gives the highest ROUGE-L F1 against PMA among layers 0, 12, and 23. Appendix E gives the scoring procedure and PMA feature-basis equivalence.

Following the released evaluator, we measure sentence-embedding cosine similarity (Reimers & Gurevych, 2019) and ROUGE-L F1 (Lin, 2004), capturing semantic similarity and word-sequence overlap.[2]

Table 2: Decode reconstruction from layer 23 over 384 target tokens.

Setting	Sentence cosine similarity ↓	ROUGE-L F1 ↓
Ordinary	1.0000	1.0000
STIP-Attn	1.0000	1.0000
AloePri-Attn	0.9965	0.9813
PMA	**0.3997**	**0.2419**

[2]The released evaluator calls sentence-embedding cosine similarity BERTScore; standard BERTScore instead compares token representations (Zhang et al., 2020).

Results. PMA reduces both semantic and word-sequence agreement: ROUGE-L F1 falls from 0.9813–1.0000 for the other settings to 0.2419, and sentence cosine similarity falls to 0.3997 (Table 2). Because feature coordinates are aligned, this comparison shows the benefit of permuting completed windows under the evaluated K/V-matching procedure.

5.4 Language Quality and Numerical Accuracy

PMA maintains language-model quality close to Ordinary inference: its perplexity is 6.5148, compared with 6.5153 for Ordinary, and its full-vocabulary logit mean absolute error (MAE) is 0.0601 (Table 3). Perplexity uses the complete WikiText-2 test split; logit MAE compares the methods on identical inputs at 256 uniformly spaced token positions. AloePri-Attn exhibits a larger logit deviation of 0.2534 in this setting. Quality is evaluated with PMA integrated into the model forward pass; latency is measured with separate CPU Client and GPU Server execution.

Table 3: WikiText-2 test perplexity and full-vocabulary logit MAE relative to Ordinary inference on the same inputs.

Setting	PPL ↓	Logit MAE ↓
Ordinary	6.5153	0.0000
STIP-Attn	6.5146	0.0530
AloePri-Attn	6.6193	0.2534
PMA	6.5148	0.0601

5.5 End-to-End Efficiency

We measure the same-host deployment described above. Each request contains one prefill pass followed by 16 cached decode steps, replaying the same fixed reference token trajectory across methods. We evaluate prompt lengths $T \in \{256, 512, 1024, 2048\}$ in increasing and decreasing order.

At each length, we report medians over ten PMA and ten Trusted-Client Full Attention requests and twenty Ordinary requests, combining increasing/decreasing length sweeps. Warmup and timing details are in Appendix F.

Table 4: Median same-host end-to-end request latency. Full denotes Trusted-Client Full Attention. Each request contains one prefill pass and 16 cached decode steps.

T	Ord. (ms)	PMA (ms)	PMA/ Ord.	Full (ms)	Full/ Ord.
256	553.592	1486.440	2.685×	1758.830	3.177×
512	559.674	1481.687	2.647×	1962.145	3.506×
1024	577.581	1591.153	2.755×	2930.024	5.073×
2048	615.551	1767.318	2.871×	6062.750	9.849×

End-to-end latency. PMA reduces request latency relative to Trusted-Client Full Attention at every tested prompt length. At $T = 2048$, it takes 1767.318 ms versus 6062.750 ms, a 3.43× speedup (Table 4). Across $T = 256$–2048, the PMA-to-Ordinary ratio remains 2.647–2.871×, whereas the Full Attention ratio grows from 3.177× to 9.849×. This trend is consistent with the bounded Client attention work in Section 4.5: PMA keeps historical attention on the GPU and limits the Client to the current block or pending window.

Communication cost. The same-host measurements include local CPU–GPU data movement. We also apply controlled delay to the same logical protocol messages using 1 Gbps bandwidth and 1 ms round-trip latency. Appendix F specifies the emulation setup.

At $T = 2048$, the measured 16-step request transfers 1161.023 MiB of core protocol tensors, excluding metadata and framing, and its median latency increases from 1767.318 ms in the same-host deployment to 12582.258 ms. Under this condition, the emulated transfer delay accounts for most of

Table 5: Qwen2.5-32B same-host request latency for one prefill pass and 16 cached decode steps.

T	Ordinary (ms)	PMA (ms)	PMA/Ordinary
256	929.314	4409.579	4.745×
512	960.601	5379.032	5.600×
1024	1045.048	5443.676	5.209×
2048	1246.847	6467.061	5.187×

the request latency. The same-host deployment avoids this external-link cost while retaining local CPU–GPU data movement.

A completed 16-token window adds 3 MiB of core upload traffic across the 24 layers, giving an asymptotic core decode traffic of approximately 0.564 MiB per cached token. Appendix H provides traffic accounting and upload verification.

Larger model with GQA. Table 5 reports the same-host execution results on Qwen2.5-32B (64 layers, 40 query heads, eight K/V heads). It uses BF16, $B = \tau = 16$, and one prefill pass followed by 16 cached decode steps. On this larger GQA model, PMA has a 4.75–5.60× request-latency ratio relative to Ordinary. The stage breakdown is in Appendix F.2.

6 Conclusion

PMA extends feature obfuscation to the sequence axis by partitioning attention between a Trusted Client and a Server. The Client handles causal blocks and pending decode windows, while the Server computes historical attention over permuted rows; exact state merging preserves ordinary attention in real arithmetic.

In the evaluated reconstruction attacks, PMA reduces prefill TTRSR to 0.00% over 240 target tokens and decode ROUGE-L F1 to 0.2419, while maintaining language-model quality close to Ordinary inference. At $T = 2048$, PMA achieves a 3.43× speedup over Trusted-Client Full Attention in the same-host 16-cached-step benchmark. Execution on Qwen2.5-32B further demonstrates compatibility with GQA. These results demonstrate the value of extending feature obfuscation with sequence protection and bounded Client attention work. Future work will strengthen protection against adaptive reconstruction from the remaining block/window structure and interaction transcript, while reducing communication cost.

Reproducibility Statement

The supplementary code includes PMA, evaluation entry points, fixed configurations, prompt records, and compact results. The appendix provides derivations, implementation settings, and traffic accounting. The public package does not include the third-party STIP-Attn/AloePri-Attn implementations or the Qwen2.5-32B result directory. The model-level quality evaluation and separate CPU–GPU execution used for timing are described in Sections 5.1 and 5.4.

AI Use Statement

Generative AI tools assisted with literature search and synthesis, manuscript editing and formatting, code review and revision, experimental-methodology feedback, and interpretation of validation results. We reviewed all AI-assisted revisions and verified the reported experimental results against the implementation and experiment records. The authors are responsible for the final manuscript and supplementary materials.

References

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 Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 3510–3520, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.277. URL https://aclanthology.org/2022.findings-acl.277/.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359. Curran Associates, Inc., 2022. doi: 10.52202/068431-1189. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

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*, volume 35, pp. 15718–15731. Curran Associates, Inc., 2022. doi: 10.52202/068431-1143. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/64e2449d74f84e5b1a5c96ba7b3d308e-Abstract-Conference.html.

Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P. Xing, and Hao Zhang. MPCFormer: Fast, performant and private transformer inference with MPC. In *The Eleventh International Conference on Learning Representations*, Kigali, Rwanda, May 2023. URL https://openreview.net/forum?id=CWmvjOEhgH-.

Zhengyi Li, Yakai Wang, Jingwen Leng, Kang Yang, Yu Yu, Jiaping Gui, Yu Feng, Ning Liu, and Minyi Guo. On the (in-)security of the shuffling defense in the transformer secure inference. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29084–29098, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1341. URL https://aclanthology.org/2026.acl-long.1341/.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Yu Lin, Qizhi Zhang, Wenqiang Ruan, Daode Zhang, Jue Hong, Ye Wu, Hanning Xia, Yunlong Mao, and Sheng Zhong. Towards privacy-preserving LLM inference via covariant obfuscation (technical report). *arXiv preprint arXiv:2603.01499*, 2026. doi: 10.48550/arXiv.2603.01499. URL https://arxiv.org/abs/2603.01499.

Jinglong Luo, Guanzhong Chen, Yehong Zhang, Shiyu Liu, Hui Wang, Yue Yu, Xun Zhou, Yuan Qi, and Zenglin Xu. CENTAUR: Bridging the impossible trinity of privacy, efficiency, and performance in privacy-preserving transformer inference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22751–22770, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1109. URL https://aclanthology.org/2025.acl-long.1109/.

Zhifan Luo, Shuo Shao, Su Zhang, Lijing Zhou, Yuke Hu, Chenxu Zhao, Zhihao Liu, and Zhan Qin. Shadow in the cache: Unveiling and mitigating privacy risks of KV-cache in LLM inference. In *Proceedings of the 33rd Annual Network and Distributed System Security Symposium (NDSS 2026)*, San Diego, CA, USA, February 2026. Internet Society. ISBN 979-8-9919276-8-0. doi: 10.14722/ndss.2026.240258. URL https://www.ndss-symposium.org/ndss-paper/shadow-in-the-cache-unveiling-and-mitigating-privacy-risks-of-kv-cache-in-llm-inference/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax. *arXiv preprint arXiv:1805.02867*, 2018. doi: 10.48550/arXiv.1805.02867. URL https://arxiv.org/abs/1805.02867.

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, San Francisco, CA, USA, May 2024. IEEE. ISBN 979-8-3503-3130-1. doi: 10.1109/SP54263.2024.00130. URL https://doi.org/10.1109/SP54263.2024.00130.

Qwen Team. Qwen1.5-MoE: Matching 7b model performance with 1/3 activated parameters, March 2024a. URL https://qwenlm.github.io/blog/qwen-moe/.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024b. URL https://arxiv.org/abs/2412.15115.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, February 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Rahul Krishna Thomas, Louai Zahran, Erica Choi, Akilesh Potti, Micah Goldblum, and Arka Pal. Hidden no more: Attacking and defending private third-party LLM inference. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 59434–59469. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/thomas25b.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

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, Seattle, WA, August 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/xu-tianshi.

Mu Yuan, Lan Zhang, Yihang Cheng, Miao-Hui Song, Guoliang Xing, and Xiang-Yang Li. STIP: Three-party privacy-preserving and lossless inference for large transformers in production. In *Proceedings of the 33rd Annual Network and Distributed System Security Symposium (NDSS 2026)*, San Diego, CA, USA, February 2026. Internet Society. ISBN 979-8-9919276-8-0. doi: 10.14722/ndss.2026.230035. URL https://www.ndss-symposium.org/ndss-paper/stip-three-party-privacy-preserving-and-lossless-inference-for-large-transformers-in-production/.

Bo Zeng, Zhi Pang, Yuyang Zhang, Kai Zhao, Tian Wu, Geying Yang, Lina Wang, and Run Wang. PCFormer: Accelerating privacy-preserving transformer inference by partition and combination. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(33):28076–28084, March 2026. doi: 10.1609/aaai.v40i33.40033. URL https://ojs.aaai.org/index.php/AAAI/article/view/40033.

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)*, San Diego, CA, USA, February 2025. Internet Society. ISBN 979-8-9894372-8-3. doi: 10.14722/ndss.2025.230868. URL https://www.ndss-symposium.org/ndss-paper/secure-transformer-inference-made-non-interactive/.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

Fei Zheng, Chaochao Chen, Zhongxuan Han, and Xiaolin Zheng. PermLLM: Private inference of large language models within 3 seconds under WAN. *arXiv preprint arXiv:2405.18744*, 2024. doi: 10.48550/arXiv.2405.18744. URL https://arxiv.org/abs/2405.18744.

A SCOPE

This supplement provides the derivations, experimental settings, and additional measurements omitted from the main paper for space. Unless stated otherwise, experiments use Qwen1.5-MoE-A2.7B in BF16 with batch size one, prefill block size $B = 16$, and decode window capacity $\tau = 16$.

B CORRECTNESS DETAILS

B.1 FEATURE TRANSFORMATIONS

For one attention head, let Q^{R} and K^{R} denote the query and key representations after RoPE. With invertible $G, H \in \mathbb{R}^{d_h \times d_h}$, PMA uses

$$\widehat{Q} = Q^{\mathrm{R}} G, \qquad \widehat{K} = K^{\mathrm{R}} G^{-\top},$$
$$\widehat{V} = V H, \qquad \widehat{W}_O = H^{-1} W_O.$$

The query/key product is unchanged:

$$\widehat{Q}\widehat{K}^{\top} = Q^{\mathrm{R}} G (G^{-\top})^{\top} (K^{\mathrm{R}})^{\top} = Q^{\mathrm{R}} (K^{\mathrm{R}})^{\top}.$$

The attention weights are therefore unchanged. The transformed value basis passes through the weighted sum and is removed by $\widehat{W}_O$:

$$\mathrm{Attn}(\widehat{Q}, \widehat{K}, \widehat{V}; M)\widehat{W}_O = \mathrm{Attn}(Q^{\mathrm{R}}, K^{\mathrm{R}}, V; M) W_O.$$

Under grouped-query attention, query heads assigned to the same K/V head share G and H. Each query head uses the corresponding row block of W_O.

B.2 ROW PERMUTATIONS

Let Π permute query rows, and let Σ permute paired K/V rows. For an unmasked attention branch,

$$\mathrm{softmax}_{\mathrm{row}}\left(\frac{\Pi Q (\Sigma K)^{\top}}{\sqrt{d_h}}\right) \Sigma V$$
$$= \Pi\, \mathrm{softmax}_{\mathrm{row}}\left(\frac{Q K^{\top}}{\sqrt{d_h}}\right) V.$$

The equality follows because right multiplication of the logits by $\Sigma^{\top}$ only permutes entries within each softmax row, and multiplication by ΣV restores every key–value pair. The remaining Π only permutes output rows. The Client applies the inverse row mapping before the partial states are merged.

B.3 EXACT MERGE

For one fixed query q and disjoint K/V index sets $\mathcal{A}_1, \mathcal{A}_2$, let $a_s = q k_s^{\top}/\sqrt{d_h} + M_s$ be the scaled, masked score and v_s its paired value vector. For a nonempty set, define

$$Z_j = \sum_{s \in \mathcal{A}_j} \exp(a_s), \qquad \ell_j = \log Z_j,$$
$$U_j = Z_j^{-1} \sum_{s \in \mathcal{A}_j} \exp(a_s) v_s.$$

The normalization over their union is

$$\ell = \log(Z_1 + Z_2) = \mathrm{logaddexp}(\ell_1, \ell_2).$$

The corresponding output is

$$U = \frac{\sum_{s \in \mathcal{A}_1} \exp(a_s) v_s + \sum_{s \in \mathcal{A}_2} \exp(a_s) v_s}{Z_1 + Z_2}$$
$$= \exp(\ell_1 - \ell) U_1 + \exp(\ell_2 - \ell) U_2.$$

Thus the merge used by PMA equals ordinary attention over $\mathcal{A}_1 \dot{\cup} \mathcal{A}_2$ in real arithmetic. Finite-precision evaluation and changed reduction order can introduce BF16 deviations.

B.4 CAUSAL-SUPPORT COUNTS

For a causal block of B valid tokens, the support matrix C has row sums $(1, \ldots, B)$ and column sums $(B, \ldots, 1)$. If the Server observes $\widetilde{C} = \Pi C \Sigma^\top$, then

$$\widetilde{C}\mathbf{1} = \Pi C \mathbf{1}, \qquad \widetilde{C}^\top \mathbf{1} = \Sigma C^\top \mathbf{1}.$$

All counts are distinct. Sorting row counts in increasing order recovers the logical query order; sorting column counts in decreasing order recovers the logical K/V order. This is the recovery procedure used only for the PMA without causal partition ablation.

B.5 TRUSTED-CLIENT ATTENTION WORK

For prompt blocks $\mathcal{B}_i$ of size at most B,

$$\sum_i |\mathcal{B}_i|^2 d_h \le B \sum_i |\mathcal{B}_i| d_h = BT d_h.$$

The Client's prefill attention work is therefore $O(TBd_h)$, while complete causal attention is $O(T^2 d_h)$. During decode, the Client attends to at most τ pending rows per step, so its attention work is $O(\tau d_h)$ independently of the historical-cache length. These bounds cover attention computation rather than data movement or the remaining model.

C EXPERIMENTAL CONFIGURATION

Table 6: Shared experimental settings.

Item	Setting
Primary model	Qwen1.5-MoE-A2.7B
Precision and batch	BF16, batch size 1
Prefill block size	$B = 16$
Decode window	$\tau = 16$
Reconstruction data	WikiText-2 validation
Perplexity data	Complete WikiText-2 test split
Timing lengths	$256, 512, 1024, 2048$
Decode steps per timed request	16 cached steps

Data preparation. The 30 reconstruction groups come from the WikiText-2 validation split. Both reconstruction experiments use these same source groups, from which experiment-specific prefill and decode records are derived. Prefill uses all 30 groups; decode uses six groups for calibration and 24 for evaluation. The exact records, group identifiers, and preprocessing script are included with the code. Language quality uses all 299,078 tokenizer tokens in the WikiText-2 test split, with context length 2048 and stride 512.

Hardware and software. The primary experiments use one NVIDIA RTX PRO 6000 Blackwell Server Edition GPU with 96 GB memory and two Intel Xeon Platinum 8470Q CPUs. The process is bound to 52 physical cores on one NUMA node, and the Trusted Client uses 40 CPU threads. The software environment uses Ubuntu 22.04.1 LTS, Python 3.10.20, PyTorch 2.11.0+cu128 with CUDA runtime 12.8, and a repository-local Transformers 5.13.0.dev0 source tree. The Trusted Client is simulated by an ordinary CPU process; hardware-TEE execution, protected-memory, and attestation costs are outside this measurement setup.

AloePri-Attn quality configuration. For language-quality evaluation, AloePri-Attn folds its Q/K feature transformations into the projection weights before RoPE, with no additional transformation after RoPE. This attention-scope adaptation differs from the literal Algorithm 2 construction in AloePri. Across all 24 layers, the block parameters are $\beta = 8$, $\gamma = 1000$, and a maximum window size of 8; the Q/K log-scale standard deviation is 0.35 and the minimum absolute scale is 0.25. The V/O map uses a controlled spectrum with target condition number 16. Embeddings, post-attention RMSNorm, FFN/MoE, final RMSNorm, and the language-model head retain Ordinary computation.

Randomness. The performance input seed is 20260711, the quality-evaluation seed is 20260728, and the decode victim and permutation seeds are 20260726. Prefill victim traces are generated once and then reused by every setting; the trace artifacts record commitments for the request-specific PMA and AloePri transformation seeds. Each formal experiment uses the recorded seed and saved victim trace.

D PREFILL RECONSTRUCTION

D.1 SCORING RULE

At target index i, the fixed public ORDINARY model proposes 256 candidate token IDs. For $X \in \{Q, K, V\}$, let H_X be the number of heads, $X^{\mathrm{pub}}_{v,i,h}$ the representation of candidate token v in head h, and $\widetilde{X}_{i,h}$ the observed representation supplied to the scorer. The candidate distance is

$$d_X(v,i) = \frac{1}{H_X} \sum_h \left[1 - \cos\Big(X^{\mathrm{pub}}_{v,i,h}, \widetilde{X}_{i,h}\Big)\right],$$

$$d_{\mathrm{raw}} = \frac{d_Q + d_K + d_V}{3}.$$

Here $\cos$ is cosine similarity and $d_{\cos}(x,y) = 1 - \cos(x,y)$. The raw-coordinate comparison uses the coordinates visible to the Server; the attack does not learn an inverse feature transformation.

For the attention comparison, let $A \in \mathbb{R}^{H_q \times 16}$ contain the pre-softmax attention scores across H_q query heads from the current query row to the 16 known-prefix positions. We compute

$$\mu(A) = \frac{1}{H_q} \mathbf{1}^\top A,$$

$$C(A) = \frac{1}{H_q} (A - \mathbf{1}\mu(A))^\top (A - \mathbf{1}\mu(A)).$$

The sequence score is

$$d_{\mathrm{seq}} = \frac{1}{2}\Big[d_{\cos}\Big(\mu(A^{\mathrm{pub}}), \mu(\widetilde{A})\Big) + d_{\cos}\Big(\mathrm{vech}(C(A^{\mathrm{pub}})), \mathrm{vech}(C(\widetilde{A}))\Big)\Big],$$

where vech contains the upper-triangular entries. This descriptor is invariant to one common permutation of the query heads but retains all 16 prefix coordinates and their cross-position covariance. The final distance is fixed for all settings:

$$d_{\mathrm{total}} = \frac{d_{\mathrm{raw}} + d_{\mathrm{seq}}}{2}.$$

The lowest-distance candidate is appended to that setting's reconstructed prefix, and the next position is then evaluated.

D.2 SERVER-VISIBLE ROWS

Table 7: Rows used by the prefill reconstruction procedure.

Setting	Rows supplied to the scorer
ORDINARY, STIP-ATTN, ALOEPRI-ATTN	Logical sequence order
PMA w/o causal partition	Degree-recovered logical order
PMA	Physical order; no assignment or causal support

All settings use the same public proposal model, candidate budget, layer 12, initial prefix, and distance. The public proposal contains the target at 91.25% of positions when conditioned on the true preceding tokens. The main result uses sequential reconstruction, for which earlier predictions determine later candidate sets. TTRSR is the fraction of the 240 target positions whose reconstructed token ID equals the reference token ID at the same logical position.

E DECODE RECONSTRUCTION

The decode experiment follows the released KV-cache reconstruction procedure used in the main paper. The six calibration groups and 24 evaluation groups are disjoint. Each evaluation example contains a 16-token known prefix and a fixed 16-token continuation generated by ORDINARY inference. The target generator uses temperature 0.8 and a top-32 set; the same frozen continuations are used by every setting. The evaluation materializes the completed 16-token window offline; in the live protocol, this window is uploaded in the next Server request. Reconstruction is scored on those 16 target tokens, giving 384 evaluation tokens.

The reconstruction procedure uses 32 attack candidates, beam width 4, calibration candidate budget 64, assumed target rank 8, language-model weight 0.001, and threshold margin 0.01. Calibration estimates separate K and V distance scales from the six calibration groups. For each candidate, the scorer computes the Frobenius distances to the observed K/V row, divides each distance by its calibrated scale, and sums the two values. Thus normalization acts on matching distances rather than on the K/V vectors.

The evaluation assumes matched feature coordinates for candidate and observed states. For PMA, scoring is performed in the Ordinary feature basis: applying the same signed feature permutation to both tensors preserves their Frobenius distance. This change of basis leaves the independently permuted row order used by the PMA reconstruction test unchanged.

For ORDINARY, STIP-ATTN, and ALOEPRI-ATTN, the K/V rows are processed in generation order. For PMA, the attack receives each completed window in its Server-visible upload order and does not estimate the independent row permutations used by different K/V heads. The 16-row PMA window is materialized offline with the same feature transformations and per-head row permutations as the protocol. This artifact corresponds to the completed window that would be appended in the next Server request; it is not a termination-time capture of Client-only pending rows.

We evaluate layers 0, 12, and 23. Table 8 reports ROUGE-L F1 for every layer. Layer 23 gives the highest score against PMA and is therefore used in the main paper together with sentence-embedding cosine similarity.

Table 8: Decode ROUGE-L F1 across attack layers.

Setting	Layer 0	Layer 12	Layer 23
ORDINARY	1.0000	1.0000	1.0000
STIP-ATTN	1.0000	1.0000	1.0000
ALOEPRI-ATTN	0.9764	0.9766	0.9813
PMA	0.0911	0.2116	0.2419

The sentence score in the main paper is the cosine similarity between mean-pooled, L2-normalized all-mpnet-base-v2 sentence embeddings. ROUGE-L is word-level longest-common-subsequence F1. These definitions match the released evaluator used by the decode attack; the former is not the token-matching definition of standard BERTScore.

F LATENCY AND COMMUNICATION

F.1 QWEN1.5-MOE-A2.7B

Each timed request consists of one prefill pass followed by 16 cached decode steps using fixed reference token trajectories. These runs stop before the first completed-window upload at cached step 17. Table 9 gives the prefill and decode components of the main-paper request totals. Prefill values are milliseconds per request, and decode values are milliseconds per cached token. Request, prefill, and per-token decode medians are aggregated separately, so their displayed values need not add up exactly.

Lengths are run in increasing and decreasing order. In each direction, PMA uses two warmup requests followed by five measured requests. Ordinary is measured for five requests before and five requests after the protected paths, with no second warmup phase. The reported medians therefore use 10

Table 9: Same-host latency breakdown for Qwen1.5-MoE-A2.7B.

T	Ord. prefill	PMA prefill	Ord. decode	PMA decode
256	61.142	153.397	30.753	83.112
512	65.290	193.756	30.906	80.472
1024	79.593	286.461	31.037	81.381
2048	117.796	458.312	31.096	80.991

PMA and 20 Ordinary measurements per length. Model loading, offline provisioning, and one-time initialization are excluded.

For the controlled 1 Gbps, 1 ms round-trip condition, the logical protocol traffic and resulting PMA request latency are shown in Table 10. This experiment adds bandwidth serialization and propagation delay to the same messages while retaining the measured same-host computation and staging path.

Table 10: Controlled 1 Gbps communication experiment.

T	Traffic (MiB)	Request (ms)	Same-host (ms)
256	150.398	3792.983	1486.440
512	294.773	5088.339	1481.687
1024	583.523	7612.960	1591.153
2048	1161.023	12582.258	1767.318

The delay experiment does not model socket processing, packetization, congestion control, external-process scheduling, protected memory, attestation, or other TEE-specific costs.

F.2 QWEN2.5-32B EXECUTION

We also run the PMA execution path on Qwen2.5-32B in BF16. The model has 64 layers, hidden size 5120, 40 query heads, and eight K/V heads. The protocol remains $B = \tau = 16$, and each request again contains one prefill pass and 16 cached decode steps. Table 11 reports the same-host request latency and its prefill and decode components.

Table 11: Same-host latency for Qwen2.5-32B. Request and prefill values are ms/request; decode values are ms/token.

Stage	T	Ordinary	PMA	Ratio
Request	256	929.314	4409.579	4.745×
	512	960.601	5379.032	5.600×
	1024	1045.048	5443.676	5.209×
	2048	1246.847	6467.061	5.187×
Prefill	256	88.093	536.088	6.085×
	512	118.520	969.163	8.177×
	1024	200.303	1387.956	6.929×
	2048	395.590	2476.315	6.260×
Decode	256	52.540	239.596	4.560×
	512	52.595	276.706	5.261×
	1024	52.764	246.445	4.671×
	2048	53.199	251.889	4.735×

The execution evaluation covers both Qwen1.5-MoE-A2.7B and Qwen2.5-32B; reconstruction is evaluated on Qwen1.5-MoE-A2.7B.

G Reproduction Files

The code/data supplement contains the PMA implementation, the exact prompt records, preprocessing code, fixed PMA configurations, PMA attack and evaluation entry points, and compact machine-readable summaries for the released Qwen1.5-MoE experiments. Model weights are not included. The package includes the WikiText-2 test text and metadata used by the evaluation; the README gives the public model and dataset identifiers, expected hashes for derived records, and foreground commands. Third-party STIP-Attn and AloePri-Attn implementations and the Qwen2.5-32B result directory are not included in the public package.

H Decode-Window Upload Verification

A separate functional run on an NVIDIA A100 40 GB used Qwen1.5-MoE-A2.7B in BF16, $T = 256$, and $B = \tau = 16$. One prefill pass and 17 cached decode steps completed. All 24 layers uploaded and acknowledged one 16-token K/V window at cached step 17; the Server cache contained 272 tokens and the Client retained one active token after releasing the prepared window. The core upload payload was 3,145,728 bytes and total decode core traffic was 9,856,512 bytes. This functional run verifies the first periodic upload and cache lifecycle; the formal latency measurements use the Blackwell setup in Section 5.5.

For $D \geq 1$ cached decode steps, the current MHA configuration transfers

$$\mathrm{Bytes}_{\mathrm{decode}}(D) = D\,L(4dw + 4n_h) + \left\lfloor \frac{D-1}{\tau} \right\rfloor L(2\tau dw),$$

where $L = 24$ is the layer count, $d = 2048$ the total head width, $n_h = 16$ the head count, and $w = 2$ bytes for BF16; log-normalizers use four bytes. These are core protocol tensors only, excluding metadata and framing. The second term accounts for completed-window uploads; it is zero at $D = 16$ and first becomes nonzero at $D = 17$.