LESS WORK, SECURE INFERENCE: ACCELERATING TWO-PARTY LLMS ON SPU VIA PUBLIC CONTEXT

Anonymous authors
Paper under double-blind review

ABSTRACT

Secure two-party computation (2PC) protects private inputs and model parameters during LLM inference. In SPU-based inference, decoding without key-value (KV) reuse recomputes historical states, while cache updates using secret write indices require secure comparisons and selections. Inputs to secure inference need not be entirely confidential and may combine public context with private data. We present **MOSAIC**, a framework for efficient two-party LLM inference on SPU. It moves public-context construction to plaintext computation on the server while securely processing private inputs and generated tokens. Connecting these computations requires preserving attention dependencies across public and private inputs. MOSAIC imports locally computed per-layer KV tensors as server-private inputs to 2PC and reuses them during inference, preserving full causal context. During generation, each party updates its cache shares locally at public positions determined by disclosed lengths and generation progress, avoiding secure comparisons and selections used to hide write positions. We implement MOSAIC on SPU for GPT-2, SmolLM2-135M, and SmolLM2-360M, and evaluate effectiveness, efficiency, component contributions, and robustness to input adaptation across six datasets. MOSAIC achieves a $4.41\times$ request-time speedup over Full2PC without KV reuse in the main GPT-2 configuration and matches native plaintext predictions on 95.45% of the 220 examples.

1 INTRODUCTION

Secure two-party computation (2PC) enables LLM inference while protecting client inputs and server model parameters. SPU provides a compiler and runtime for secure computation and has been used by secure Transformer inference systems such as BumbleBee and PUMA (Lu et al., 2025; Dong et al., 2025). We study how to improve the efficiency of two-party secure LLM inference on SPU.

In some domain-specific LLM applications (Huang et al., 2026; Wu et al., 2025), inputs may combine public reference material with private data. The model must use this reference material to interpret the private data without revealing it to the server. In two-party secure LLM inference with public context, the server holds the model and has access to the public context, while the client holds private inputs. Private inputs and model-derived intermediate results must enter secure computation in protected form, rather than being disclosed to the other party. This motivates our research question: *Can we use public context to accelerate secure two-party LLM inference on SPU while preserving full causal context?* We explore using key-value (KV) caches to connect local computation with secure inference. Realizing this approach raises two connected challenges.

Challenge (C1). KV computed locally depends on model parameters and must remain hidden from the client. The challenge is to import these states into SPU-based two-party inference in a compatible protected form while preserving attention dependencies across public and private inputs.

Challenge (C2). KV reuse reduces historical recomputation, but updating the cache with secret write indices in SPU requires additional secure comparisons and selections, increasing cache maintenance costs. The challenge is how to efficiently reuse and update KV in SPU-based two-party LLM inference while protecting model parameters, private inputs, and KV states.

Prior systems optimize secure operators and module execution (Huang et al., 2022; Lu et al., 2025; Zhang et al., 2026b). MPCache reduces attention costs through KV eviction and dynamic selec-

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.

tion (Zeng et al., 2025), while SOAL caches special prefix tokens to support outlier handling and efficient nonlinear computation (Zhao & Fang, 2026). CryptoGen supports secure generation through encrypted KV reuse and updates (Zhang et al., 2026a). Park et al. (2026) compute KV for public inputs in plaintext and use it with encrypted queries in plaintext-ciphertext attention under fully homomorphic encryption. Our work focuses on integrating locally computed KV into SPU-based two-party inference and maintaining the cache during generation. The resulting execution retains all causally visible KV entries while keeping KV values and generated tokens protected. Figure 1 our execution with two-party inference without KV reuse and with KV reuse using secret write indices.

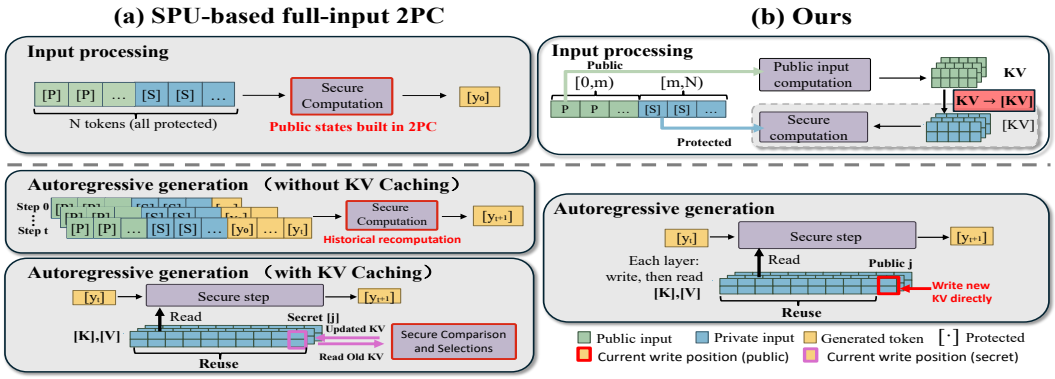


Figure 1: Comparison of MOSAIC with SPU-based full-input 2PC, which processes the entire inputs through secure computation. (a) shows variants without and with KV reuse, and (b) shows MOSAIC.

Our main contributions are summarised as follows:

- We present MOSAIC, an efficient two-party secure LLM inference framework on SPU. It reuses locally computed KV to securely process private inputs and support incremental generation. MOSAIC retains attention to all causally visible positions while protecting KV and tokens.
- We design a KV cache management mechanism that integrates locally computed KV into secure inference as private inputs. We adapt the inference routines to use the imported cache together with securely computed KV in subsequent attention. During generation, each party updates compatible cache shares locally by slicing and concatenation at public positions determined by disclosed lengths and progress. These updates retain existing entries and insert new KV without revealing their contents. We specify the requirements on numerical representations, cache layouts, and positions, and analyze privacy and execution costs.
- We implement MOSAIC on SPU and evaluate it on three models and six datasets. On GPT-2, MOSAIC achieves a $4.41\times$ speedup over Full2PC in request time. It matches native plaintext predictions on 95.45% of the 220 examples from PIQA, HellaSwag, C-Eval, and CoLA. These results show that MOSAIC integrates locally computed KV into secure inference and supports efficient cache reuse and updates during generation while retaining most native plaintext predictions.

2 Background and Model

Background. In two-party secure LLM inference, the client provides private inputs and the server provides model parameters. The two parties jointly execute model inference while protecting their respective inputs. SPU compiles model computations into operations and executes them through the selected backend (Ma et al., 2023). For secure numerical computation, SPU encodes floating-point model values in fixed-point form. To protect these encoded values, two-party additive secret sharing represents each value $\widehat{x} \in \mathbb{Z}_{2^w}$ as two shares satisfying $[x]_0 + [x]_1 = \widehat{x} \pmod{2^w}$, where w is the ring bit width (Huang et al., 2022). The shares are randomized so that either share alone reveals no information about the encoded value. Each party holds one share, and secure protocols operate on these shares without reconstructing intermediate values. Token embedding initializes hidden states, and Transformer layers process them in the model's prescribed order. Attention forms queries, keys, and values and applies Softmax to attention scores before value aggregation. Causal autoregressive

models restrict each query to its own and earlier positions (Vaswani et al., 2017). Prefill processes the input through all model layers. The output layer maps the last input position's final hidden state to vocabulary logits for selecting the first generated token. Subsequent decoding uses each selected token as input to predict the next. Incremental decoding retains per-layer key-value (KV) tensors and computes new states for each subsequent input token, avoiding historical recomputation while retaining attention to earlier positions (Kwon et al., 2023). *Full context* means that each query attends to every causally visible position, with joint score normalization and value aggregation.

System model. Let W be the server's model parameters and X the client's tokenized request for joint inference under 2PC (Huang et al., 2022; Lu et al., 2025). In causal Transformers, public tokens appearing after private content may attend to it. The KV for these public tokens cannot generally be computed independently of the preceding private content. The client partitions X into a public prefix P and a private suffix S according to confidentiality requirements, so $X = P\|S$. The public input P contains m tokens approved for disclosure to the server, and the private input S contains n tokens. The total length is $N = m + n$. We assume deterministic inference in a causal Transformer with L layers. Residual, normalization, and feed-forward operations act independently at each position. Model parameters and global positions are fixed, and positional transformations do not depend on later tokens. The first layer receives token embeddings, and subsequent layers receive hidden states from the preceding layer. Causal attention at positions in P cannot access positions in S. Together with the position-wise operations, this allows the server to compute KV for P across all L layers using only P and W, independently of S. For requests with private content interspersed with public text, we provide optional input rewriting to obtain the $P\|S$ layout (see Section 3.1).

Security and adversary model. We assume a static semi-honest adversary that corrupts at most one party before execution (Huang et al., 2022; Lu et al., 2025). Each party follows the prescribed computation and communication steps, but may attempt to infer additional private information from its view. We assume that P, the model architecture, input lengths, the split position, tensor shapes, generation progress, and the resulting cache write positions are public. We assume secure composition of the private input, operator, and conversion protocols under the stated adversary model and the specified disclosure and output rules. The server must not learn S or generated token contents beyond the permitted disclosures. The client must not learn W or model-derived KV, including KV computed from P, beyond the permitted disclosures and its authorized output. Only the authorized client reconstructs the final generated sequence Y. Each party's view should reveal no additional private information beyond its own inputs, public information, and authorized output.

3 MOSAIC FRAMEWORK

Design rationale. MOSAIC computes initial KV from P locally on the server and supplies it as a server-private input to secure inference. The inference routines use the imported per-layer KV together with securely computed KV. Each party maintains its KV cache shares and updates them locally using the public write position. Computations involving private queries remain within 2PC.

3.1 MOSAIC

MOSAIC includes three main components: public input processing, KV cache management, and secure prefill and decoding (see Figure 2). Let $[\![x]\!]$ be the secret-shared representation of x. Local and secure computation preserve the model's causal dependencies.

Public input processing. The server evaluates the public input P locally using W at global positions $0, \ldots, m-1$. The server retains the per-layer cache $C_P = \{K_P^\ell, V_P^\ell\}_{\ell=1}^L$.

KV cache management. We construct the protected cache from the locally computed per-layer KV. The server encodes these tensors in the fixed-point format used by secure inference and supplies them through existing private input protocols. The backend may initially retain a server-private representation and convert it to arithmetic shares when required by subsequent operations. For a KV tensor Z, let $\widehat{Z}$ denote its fixed-point encoding over $\mathbb{Z}_{2^w}$. The resulting client and server shares satisfy $[Z]_0 + [Z]_1 = \widehat{Z} \pmod{2^w}$, where w is the ring bit width. The relation holds elementwise for each imported K and V tensor. We adapt the inference routines to use the imported KV as the initial cache at positions $[0, m)$, preserving the layout expected by each layer.

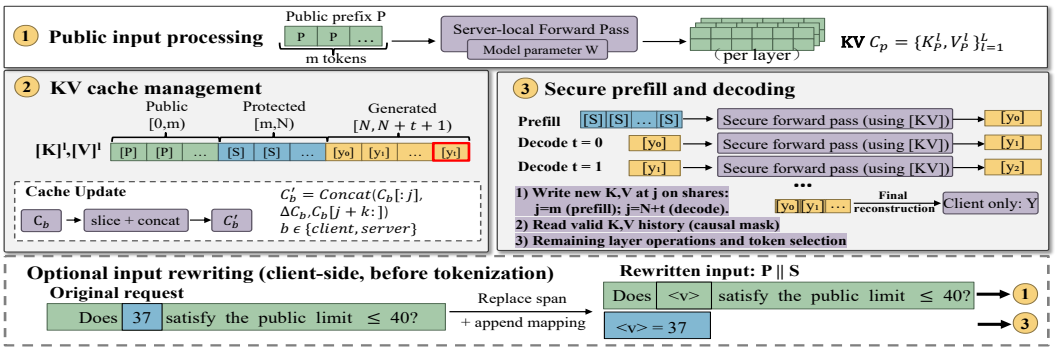


Figure 2: MOSAIC with three components: (1) public input processing, (2) KV cache management, and (3) secure prefill and decoding. The bottom example illustrates optional input rewriting into the $P\|S$ layout. Colors and symbols follow the conventions in Figure 1.

Each layer maintains protected K and V caches. Prefill of the private input adds n entries at $(j, k) = (m, n)$, giving valid cache positions $[0, N)$, and produces the first generated token. Decoding step t adds one entry at $(j, k) = (N + t, 1)$, giving valid cache positions $[0, N + t + 1)$. At decoding step $t = 0$, the first generated token is fed back into the model to predict the next token. The write position j and update length k follow the public lengths and generation progress.

Let $b = 0$ denote the client and $b = 1$ the server. Let C_b be party b's share of a layer's K or V cache, and ΔC_b its share of the k new entries. The cache and incoming entries use compatible share representations and numerical encodings. For a preallocated cache with capacity T, the write interval satisfies $0 \le j \le j + k \le T$. Each party updates its cache share locally by slicing and concatenation: $C_b' = \mathrm{Concat}(C_{b,<j}, \Delta C_b, C_{b,\ge j+k})$. Both parties retain the rows outside $[j, j + k)$ and replace the rows inside it with their shares of the new KV. Appending to the logical history writes new entries into the next unused rows and extends the valid range. If the cache instead stores only the valid history and grows dynamically, appending uses $C_b' = \mathrm{Concat}(C_b, \Delta C_b)$.

Secure prefill and decoding. We adapt prefill and decoding to use the imported KV together with KV computed securely from private inputs and generated tokens. At each layer, computed KV is written to the cache at public positions before attention reads the valid entries. Attention jointly uses all causally visible entries. The cache is retained across decoding steps, and selected tokens remain protected when used as subsequent inputs. Only the authorized client reconstructs the sequence.

Input rewriting. Private content may be interspersed with public text in the original request. To support such requests, the client can replace private spans with placeholders and append mappings from the placeholders to their original private content as a private suffix before tokenization. The rewritten request follows the $P\|S$ layout and is processed by the components described above. Figure 2 illustrates this optional step, and Section 4.4 evaluates its effects.

3.2 Context Preservation, Privacy, and Efficiency Analysis

Context preservation. Under the assumptions in Section 2, attention at each position in P uses only preceding or current positions in P. Position-wise operations preserve this independence across layers. The server can therefore compute KV for public inputs locally using only P and W.

Our reference is the full-input two-party inference under the same backend, using the same model and input. For a fixed generation history, assume matched parameters, positions, masks, and per-layer cache layouts, with KV correctly imported and maintained. Prefill of the private input uses the imported KV for public inputs together with securely computed KV for all causally visible private positions. Each decoding update retains earlier entries and adds the current KV, covering $[0, N+t+1)$. Joint normalization and value aggregation preserve the reference's causal attention dependencies.

Numerical equivalence to this two-party reference holds when the imported KV matches the reference's encoded KV values and subsequent operations use the same numerical semantics, including any randomness affecting numerical results. For a fixed input and generation history, these conditions

yield identical subsequent states and logits. Matching token selection, stopping rules, and sampling randomness extends this equivalence to generated sequences. Fixed-point encoding and operator approximations may still produce differences from native floating-point inference. Appendix A.2 provides the detailed conditions and proof of numerical equivalence.

Privacy. We use the security model and protocol composition assumptions in Section 2. Local construction uses only P, W, which the server already knows. The input and conversion protocols keep imported KV hidden from the client while providing the representations needed by subsequent operations. The secure protocols protect intermediate values, logits, and selected tokens. Once compatible new KV shares are available, each local update is a deterministic function of the party's existing cache shares, new KV shares, and disclosed metadata. It reveals no values and exchanges no messages, adding no information beyond that party's existing view. Under the stated protocol assumptions, a party's full view reveals no additional private information beyond its own inputs, public information, and authorized output. Appendix A.3 details the conditional argument.

Efficiency and cost. Local computation moves the operations for public inputs out of secure execution, while private queries retain attention to the KV of public inputs. Importing this KV incurs encoding, private input, and required representation conversion costs. During generation, KV reuse avoids recomputing historical states, although each query attends to all causally visible positions. With public write positions, each party updates compatible cache shares through local slicing and concatenation. This avoids secure comparisons and selections for hiding write positions and requires no communication for the update itself. Computing and converting new KV remain separate costs.

4 EVALUATION

Our evaluation covers effectiveness (task quality and prediction consistency), efficiency (inference costs), ablation (component contributions), and robustness (input adaptation).

Models and datasets. We evaluate GPT-2, SmolLM2-135M (135M), and SmolLM2-360M (360M) (Radford et al., 2019; Hugging Face, a;b) on six datasets. PIQA (Bisk et al., 2020) and HellaSwag (Zellers et al., 2019) evaluate commonsense reasoning through multiple-choice questions. C-Eval (Huang et al., 2023) covers Chinese examination questions across subjects. CoLA (Warstadt et al., 2019) evaluates linguistic acceptability. WikiText-2 (Merity et al., 2017) and ASSET (Alva-Manchego et al., 2020) evaluate language modeling and text simplification, respectively.

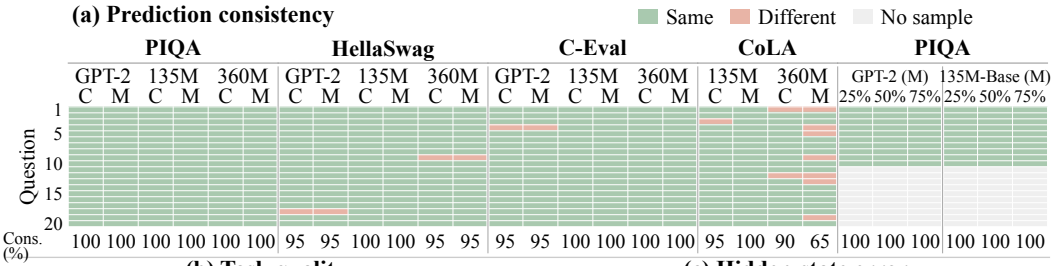
Implementation and environment. We implement MOSAIC (M) for GPT-2, SmolLM2-135M, and SmolLM2-360M (Radford et al., 2019; Hugging Face, a;b) using Hugging Face Flax models and OpenBumbleBee's secure operators on SPU with the Cheetah two-party backend (Lu et al., 2025; Huang et al., 2022). For GPT-2, we adapt inference to import server-provided KV as private input and reuse it across decoding steps. For 135M and 360M, we adapt the Flax LLaMA implementation, preserve rotary position embeddings and cache layouts, and rescale RMSNorm intermediates for fixed-point stability. KV computed via the compatible plaintext numerical path enters SPU through private input and representation conversion. We extend fixed-point `dynamic_update_slice` to update each party's cache shares locally at public positions. Generated tokens remain protected as inputs to subsequent steps. For system measurements, both parties share a CPU host with 64 vCPUs and 256 GiB memory, using batch size 1 and FM64 with 16 fractional bits. We use Python 3.10.12, JAX 0.4.33, Flax 0.8.3, and Transformers 4.56.0 on Linux 5.15. SPU's high-level intermediate representation (HLO) profiles report operator call counts, execution time, and communication.

Baseline. Full2PC securely processes the entire input using SPU-based 2PC and recomputes historical states for each prediction without KV reuse. We implement compatible plaintext (C) using the numerical operators and approximations adopted for secure inference. Native plaintext (N) uses the original floating-point model implementation. Prediction consistency is the exact match rate of M or C against N. Comparing N and C assesses the effects of these operators and approximations. Comparing C and M tests whether KV integration and secure execution preserve task quality.

4.1 EFFECTIVENESS

We evaluate MOSAIC's task quality and prediction consistency after integrating locally computed KV into secure inference on GPT-2, 135M, and 360M. For all models, we report prediction consistency

with native plaintext on PIQA, HellaSwag, and C-Eval, and perplexity (PPL) on WikiText-2. The two SmolLM2 models are also evaluated on CoLA using prediction consistency against native plaintext, and on ASSET using SARI and exact match rates between complete texts generated independently by N, C, and M. We compute $\mathrm{PPL} = \exp\big(-\frac{1}{T}\sum_i \sum_{t\in\mathcal{V}_i} \log p_\theta(x_{i,t} \mid x_{i,<t})\big)$, where $\mathcal{V}_i$ contains valid scoring positions excluding padding and $T = \sum_i |\mathcal{V}_i|$. We also measure prediction consistency across private input ratios and hidden-state relative L2 errors for N–C, C–M, and N–M.



(b) Task quality

WikiText-2: PPL ↓

Model	N	C	M	ΔPPL
GPT-2	51.337	51.395	51.412	+0.15%
135M	37.338	38.299	38.309	+2.60%
360M	25.897	26.238	26.256	+1.39%

ASSET: SARI ↑ / exact match

Model	N	C	M	C–N	M–N	M–C
135M	39.199	37.337	37.501	55%	45%	75%
360M	42.180	42.761	42.545	55%	55%	75%

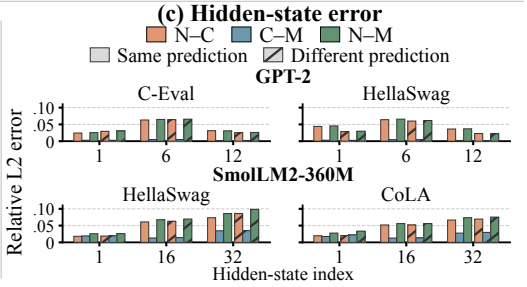


Figure 3: **Effectiveness.** In (c), solid and hatched bars indicate matching and differing predictions from M and N, respectively. Each bar averages the relative L2 errors across a question's candidates.

Figure 3(a) shows prediction consistency against native plaintext. Across the 220 examples, C and MOSAIC match N on 214 and 210 examples, respectively. Across the three models, MOSAIC achieves 100% prediction consistency on PIQA and 95–100% on HellaSwag and C-Eval, while CoLA results are 100% for 135M and 65% for 360M. For 360M on CoLA, the Matthews correlation coefficient (MCC) decreases from 0.2425 under N to -0.1155 under MOSAIC. This may reflect sensitivity of CoLA predictions to numerical approximations. Refining the approximations used for 360M could help reduce this difference. The rightmost PIQA group in Figure 3(a) compares private input ratios of 25%, 50%, and 75% for GPT-2 and 135M. MOSAIC matches native plaintext predictions on all questions at every ratio for both models.

Figure 3(b) reports language modeling and text simplification quality. On WikiText-2, MOSAIC increases PPL by 0.145–2.601% relative to N across the three models, while remaining close to C. On ASSET, its SARI scores differ from N by -1.698 and $+0.365$ points for 135M and 360M, respectively, and remain within 0.22 points of C.

To examine prediction differences in panel (a), Figure 3(c) compares hidden-state relative L2 errors at selected layer positions with 25% private input. For each of four model and dataset combinations exhibiting prediction differences, we select one example with matching N and MOSAIC predictions and one with differing predictions. The similar N–C and N–M errors, together with smaller C–M errors, suggest that the observed differences mainly reflect the numerical approximations adopted for secure inference, with smaller additional differences from KV integration and secure execution.

Overall, these results support the effectiveness of supplying locally computed KV as secret shares for subsequent secure inference. With the shared KV, MOSAIC retains native predictions on most evaluated questions and achieves PPL and SARI close to compatible plaintext. The main differences occur in 360M's CoLA predictions and the exact text produced during generation.

4.2 Efficiency

We evaluate request time and communication cost against Full2PC on GPT-2 under different network conditions, compilation reuse settings, input lengths, private input fractions, and generation lengths.

Request time includes preparation after model loading, input sharing, compilation, secure execution, and result retrieval. Execution traffic excludes input sharing and deployment RPC and is reported in decimal MB or GB. Speedup is Full2PC's request time divided by the method's.

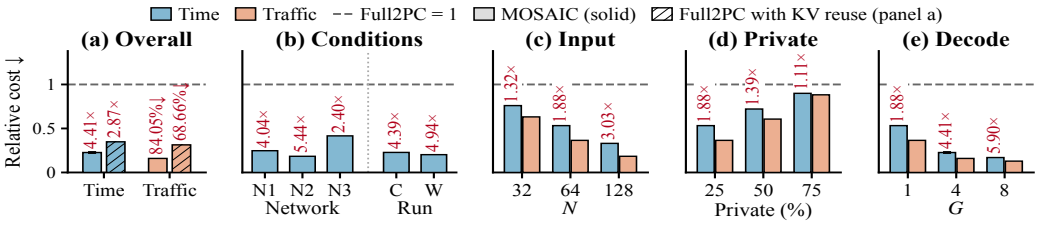


Figure 4: **Efficiency.** GPT-2 costs normalized to Full2PC without KV reuse (dashed at 1). G counts generated tokens: 4 in (a)–(b), 1 in (c)–(d), and $1, 4, 8$ in (e). Blue and orange denote request time and execution traffic; (b) reports time only. Solid bars show MOSAIC; hatched bars in (a) show Full2PC with KV reuse under the same baseline. Red labels show time speedups and, in (a), traffic reductions.

Figure 4 presents these comparisons. All methods execute the generation loop in one SPU call with secure argmax. Panels (a) and (b) use $N = 64, G = 4$, with $n = 16$ for MOSAIC. Panel (a) reports three-run means and sample standard deviations for Full2PC without KV reuse and MOSAIC. The measured gains are a $4.41\times$ speedup and an 84.05% traffic reduction. Panel (a) also includes one separately collected greedy run of Full2PC with KV reuse on the same model and input, using public cache write positions. It takes 92.870 s and transfers 4.099 GB. Its request time excludes initial cache allocation, while MOSAIC timing includes public-prefix preparation. These separate measurements are normalized to the same Full2PC baseline. Panel (b) evaluates three network conditions. N1, N2, and N3 use bandwidth/RTT pairs of (1000 Mbps, 1 ms), (100 Mbps, 1 ms), and (1000 Mbps, 10 ms), respectively. Network speedups range from $2.40\times$ to $5.44\times$. Cold (C) and warm (W) runs achieve $4.39\times$ and $4.94\times$, respectively. Warm runs reuse the compiled graph but recompute public KV.

Panels (c)–(e) vary one factor at a time from the $N = 64, n = 16, G = 1$ configuration: input length, private-input fraction, and generation length, respectively. In panel (c), increasing N from 32 to 128 while fixing $n = 16$ raises the speedup from $1.32\times$ to $3.03\times$ and the traffic reduction from 36.85% to 81.63%. Thus, the relative efficiency gain grows with public input in these settings. In panel (d), increasing the private input fraction from 25% to 75% lowers the speedup from $1.88\times$ to $1.11\times$ and the traffic reduction from 63.53% to 11.76%. The benefit decreases as more input requires secure computation. In panel (e), increasing G from 1 to 8 raises the speedup from $1.88\times$ to $5.90\times$ and the traffic reduction from 63.53% to 87.13%. The larger gains with longer generation are consistent with avoiding historical recomputation through KV reuse.

Table 1: **HLO operation costs on GPT-2**, averaged over three runs at $N = 64, n = 16, G = 4$.

		Full2PC				MOSAIC			
Operation	Role	Calls	Time (s)	Sent (MB)	Recv. (MB)	Calls	Time (s)	Sent (MB)	Recv. (MB)
equal	Equality	4	0.000631	0.000	0.000	12↑	0.000105↓	0.000	0.000
gather	Indexed read	4	0.090	0.000	0.000	1↓	0.006↓	0.000	0.000
dot	Projections / FFN	196	136.524	610.563	1291.068	196	21.205↓	116.887↓	124.065↓
dot_general	Attention	96	13.948	565.805	556.649	96	5.469↓	350.865↓	350.189↓
less	Comparison	129	0.939	1.250	0.843	121↓	0.521↓	0.785↓	0.491↓
select	Selection	20	0.163	1.637	1.637	12↓	0.149↓	0.119↓	0.119↓
gelu	Activation	48	12.328	2301.120	1539.511	48	3.044↓	176.700↓	132.257↓
neg_exp	Softmax	48	6.087	844.479	704.933	48	2.274↓	111.996↓	124.472↓
reciprocal	Softmax	48	5.795	70.199	77.590	48	0.710↓	20.859↓	27.911↓
multiply	Norm. / arithmetic	796	7.809	1644.685	1712.983	871↑	2.541↓	152.659↓	205.546↓
rsqrt	Normalization	100	0.920	8.476	7.387	100	0.572↓	0.618↓	0.200↓
dynamic_update_slice	KV updates	0	—	—	—	96 (new)	0.045	0.000	0.000
Request time (s)			**266.702**				**60.441** (77.34% ↓)		
Execution traffic (MB)			**13,080.867**				**2,086.392** (84.05% ↓)		

Table 1 reports HLO call counts, aggregate time, and communication for the same three GPT-2 runs as Figure 4(a). The final two rows report independently measured request time and execution traffic.

Operator times may overlap and are not summed into request time. The `dot` and GELU call counts remain unchanged, while their aggregate times decrease from 136.524 to 21.205 s and from 12.328 to 3.044 s, respectively. Their communication costs also decrease. Local public KV computation and KV reuse reduce work per call without requiring fewer calls. The `equal` and `multiply` call counts increase, but their aggregate times also decrease. Cache maintenance introduces 96 `dynamic_update_slice` calls in MOSAIC. Each party locally updates compatible cache shares at public positions. These local updates take 0.045 s in total with no recorded communication. This excludes new KV computation, private input, and representation conversion.

Overall, MOSAIC reduces request time and execution traffic relative to Full2PC in the evaluated GPT-2 configurations. The results support combining local public KV computation, private KV input, cache reuse, and updates at public positions.

4.3 Ablation Study

We evaluate the contributions of KV reuse and public indices to inference efficiency on GPT-2 through two inference experiments and a cache update microbenchmark. We measure time and SPU traffic. The two inference experiments compare argmax outputs under the same fixed token history.

Figure 5(a) presents the three comparisons. Reuse compares execution with and without KV reuse at $N = n = 64$ and $G = 4, 8$, using public indices in both variants. Cached execution timing excludes initial allocation. KV reuse achieves speedups of $2.87\times$ and $4.37\times$ and reduces traffic from 13.081 to 4.099 GB and from 26.973 to 5.489 GB, respectively. The larger gain at $G = 8$ shows the greater benefit of avoiding history recomputation during longer generation. Pub. compares public and secret decode indices at $N = n = 32, G = 4$, with KV reuse enabled in both variants. Both use public position zero for prefill; only the subsequent decode index handling differs. Public indices achieve a $17.77\times$ speedup and reduce traffic from 70.985 to 2.621 GB. Capacity compares cache updates using synthetic KV tensors for one GPT-2 layer at capacities of 32, 64, and 128. Each run performs one K write and one V write without a model forward pass. Public indices achieve speedups of $150.7\times$, $196.6\times$, and $288.2\times$, respectively, and record zero SPU traffic for these updates. Across the tested capacities, public updates take 0.123–0.160 s, compared with 18.562–46.009 s for secret updates.

These experiments show that KV reuse reduces history recomputation and public indices lower cache update costs. Both inference experiments retain argmax outputs at all tested positions. These comparisons support the use of KV reuse and public indices in MOSAIC.

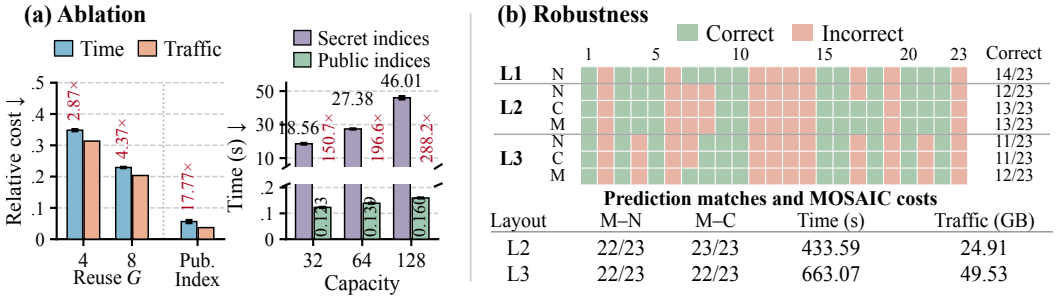


Figure 5: **Ablation and robustness.** (a) GPT-2 cost controls. (b) Per-question PIQA correctness, agreement with plaintext references, and MOSAIC time and traffic across input layouts. Reuse and Pub. show relative time and traffic (range: 0–0.5; reference: 1). Capacity plots update times (s) on a broken axis. Red labels show speedups. Error bars are sample standard deviations, normalized by the reference mean for Reuse and Pub., and in seconds for Capacity.

4.4 Robustness

We evaluate the input rewriting described in Section 3.1 for requests whose private content does not appear at the end. Using 360M on PIQA, we measure task quality, prediction consistency, and execution cost. L1 uses the original input layout. L2 replaces a question span with a placeholder, and L3 replaces candidate content. Both adaptations append a private mapping containing the removed

text. We score each teacher-forced candidate sequence by mean token log-probability and select the higher-scoring candidate. We report correct answer counts against dataset labels, prediction consistency with native plaintext, and exact match rates against compatible plaintext.

Figure 5(b) shows results on the same 23 PIQA questions under L2 and L3, with native plaintext under L1 as the original input reference. Green and red cells indicate correct and incorrect answers according to dataset labels. Native plaintext answers 14/23 questions correctly under L1, compared with 12/23 under L2 and 11/23 under L3, showing that input adaptation itself affects task quality. Within L2 and L3, MOSAIC matches native plaintext on 22/23 questions, giving a prediction consistency of 95.65% for both layouts. Its exact match rates against compatible plaintext are 100% (23/23) and 95.65% (22/23), respectively. Against dataset labels, MOSAIC answers 13/23 and 12/23 questions correctly, one more than native plaintext in each layout. These results show that MOSAIC retains most plaintext predictions on the adapted inputs, while task quality also changes with the input layout. Appendix B.4 provides the input layouts and sample coverage. Figure 5(b) also reports MOSAIC costs averaged over the same 23 questions, summing the two candidate calls for each question. L2 averages 433.59 s and 24.91 GB per question, while L3 averages 663.07 s and 49.53 GB. Timing includes compilation, input sharing, secure execution, and result retrieval, but excludes model loading, public-prefix computation, and reference checks. Execution traffic sums rank 0's sent and received bytes during SPU execution (decimal GB), excluding input sharing and deployment RPC.

Overall, MOSAIC retains native plaintext predictions on most questions under both tested adaptations. Input adaptation extends the applicability of MOSAIC to requests whose private content does not originally appear at the end. Rewriting also affects task quality in native plaintext inference.

5 Related Work

Cryptographic secure inference. Pang et al. (2024) and Lu et al. (2025) optimize secure matrix multiplication and nonlinear operations for interactive Transformer inference. Gupta et al. (2024) combine function secret sharing with GPU acceleration for secure inference. Zhang et al. (2026b) jointly optimize modules and ciphertext packing for non-interactive inference. Li et al. (2023) combine MPC-friendly approximations with knowledge distillation. These optimizations complement MOSAIC, which imports locally computed KV and maintains protected caches during generation.

KV cache management. Kwon et al. (2023) and Gim et al. (2024) improve model serving through KV memory management and reusable prompt attention states, respectively. For secure inference, Zeng et al. (2025) and Zhao & Fang (2026) exploit KV selection and eviction and special-prefix caching, respectively, to reduce attention costs or accelerate nonlinear operations. Zhang et al. (2026a) combine batched ciphertext operation optimizations with encrypted KV cache reuse for secure autoregressive generation. Park et al. (2026) use plaintext KV from public context and encrypted KV from private inputs in homomorphic attention, combining their scores before Softmax. MOSAIC uses existing private input protocols and representation conversions to integrate locally computed public-context KV as secret shares into SPU-based two-party inference.

Prompt processing. Chen et al. (2023) and Hou et al. (2025) propose methods that replace sensitive entities before remote inference and restore them in the outputs. Roy Chowdhury et al. (2026) introduce prompt sanitization using format-preserving encryption or metric differential privacy according to the task's utility requirements. In MOSAIC, placeholders reorganize the input, while the original sensitive content participates in secure inference through private mappings. This adaptation extends MOSAIC to requests whose private content does not originally appear at the end.

6 Conclusion

We presented MOSAIC, a framework for full-context two-party LLM inference on SPU. It integrates locally computed KV through private input protocols and supports cache reuse and local share updates at public positions, preserving full causal attention dependencies. We implemented MOSAIC and evaluated it on three models and six datasets. MOSAIC reduces inference costs while retaining most native plaintext predictions, achieving a $4.41\times$ request-time speedup over Full2PC in the main GPT-2 configuration and matching native predictions on 95.45% of the 220 examples.

AI USE STATEMENT

We used generative AI tools, including OpenAI Codex, to assist with literature search and review, mathematical reasoning, code development, interpretation of results, and manuscript drafting and revision. The authors take responsibility for all content, claims, results, and artifacts in the submission.

REPRODUCIBILITY STATEMENT

Section 4 specifies the models, datasets, baselines, software versions, hardware, and measurement definitions used in our experiments. Appendix B provides additional experimental results, task-specific evaluation settings, and sample coverage. Code is available at `https://anonymous.4open.science/r/MOSAIC-EED8/`.

REFERENCES

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4668–4679, 2020.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Yu Chen, Tingxin Li, Huiming Liu, and Yang Yu. Hide and seek (HaS): A lightweight framework for prompt privacy protection, 2023.

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.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. volume 6, pp. 325–338, 2024.

Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. SIGMA: secure GPT inference with function secret sharing. *Proc. Priv. Enhancing Technol.*, 2024(4):61–79, 2024.

Shilong Hou, Ruilin Shang, Zi Long, Xianghua Fu, and Yin Chen. A general pseudonymization framework for cloud-based llms: Replacing privacy information in controlled text generation, 2025.

Haoliang Huang, Zihuang Cai, Zhuo Tang, Yifan Liu, Chen Tian, Kenli Li, and Changjian Chen. Integrating data validation with large language models for regulation-guided tabular anomaly detection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6559–6581, 2026.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36: 62991–63010, 2023.

Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and fast secure {Two-Party} deep neural network inference. In *31st USENIX security symposium (USENIX Security 22)*, pp. 809–826, 2022.

Hugging Face. SmolLM2-135M-Instruct, a. URL `https://huggingface.co/HuggingFaceTB/SmolLM2-135M-Instruct`.

Hugging Face. SmolLM2-360M-Instruct, b. URL `https://huggingface.co/HuggingFaceTB/SmolLM2-360M-Instruct`.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Dacheng Li, Hongyi Wang, Rulin Shao, Han Guo, Eric P. Xing, and Hao Zhang. MPCFormer: Fast, performant and provate transformer inference with MPC. In *ICLR*, 2023.

Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and Wenguang Chen. Bumblebee: Secure two-party inference framework for large transformers. In *NDSS*. The Internet Society, 2025.

Junming Ma, Yancheng Zheng, Jun Feng, Derun Zhao, Haoqi Wu, Wenjing Fang, Jin Tan, Chaofan Yu, Benyu Zhang, and Lei Wang. Secretflow-spu: A performant and user-friendly framework for privacy-preserving machine learning. In *USENIX ATC*, pp. 17–33. USENIX Association, 2023.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. BOLT: privacy-preserving, accurate and efficient inference for transformers. In *SP*, pp. 4753–4771. IEEE, 2024.

Jaiyoung Park, Sejin Park, Jai Hyun Park, Jung Ho Ahn, Jung Hee Cheon, Guillaume Hanrot, Jung Woo Kim, Minje Park, and Damien Stehlé. Scaling up privacy-preserving ml: A ckks implementation of llama-2-7b. *arXiv preprint arXiv:2601.18511*, 2026.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Amrita Roy Chowdhury, David Glukhov, Divyam Anshumaan, Prasad Chalasani, Nicholas Papernot, Somesh Jha, and Mihir Bellare. Pr$\varepsilon\varepsilon$mpt: Sanitizing sensitive prompts for LLMs. In *Network and Distributed System Security Symposium*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641, 2019.

Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, Yueming Jin, and Vicente Grau. Medical graph RAG: Evidence-based medical large language model via graph retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28443–28467. Association for Computational Linguistics, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *ACL (1)*, pp. 4791–4800. Association for Computational Linguistics, 2019.

Wenxuan Zeng, Ye Dong, Jinjin Zhou, Jin Tan, Lei Wang, Tao Wei, Runsheng Wang, and Meng Li. MPCache: MPC-Friendly KV Cache Eviction for Efficient Private LLM Inference. In *NeurIPS*, 2025.

Hedong Zhang, Neusha Javidnia, Shweta Pardeshi, Qian Lou, and Farinaz Koushanfar. CryptoGen: Secure transformer generation with encrypted KV-cache reuse. *CoRR*, abs/2602.08798, 2026a.

Linru Zhang, Xiangning Wang, Jun Jie Sim, Zhicong Huang, Jiahao Zhong, Huaxiong Wang, Pu Duan, and Kwok Yan Lam. MOAI: Module-optimizing architecture for non-interactive secure transformer inference. In *International Conference on Learning Representations*, volume 2026, pp. 27927–27953, 2026b.

Lifan Zhao and Zhixuan Fang. Secure outlier-aware large language model inference. In *International Conference on Learning Representations*, volume 2026, pp. 101033–101045, 2026.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. SGLang: Efficient execution of structured language model programs. volume 37, pp. 62557–62583, 2024.

A Execution Details and Analysis

This section supplements Section 3 with cache update and attention details, context preservation and numerical correspondence arguments, privacy analysis, and cost accounting.

A.1 Cache Updates and Full-Context Attention

Public-position updates. Let C be a layer's K or V cache and ΔC contain k new entries. The local update assumes compatible secret-shared representations and numerical encodings for both. Let $b = 0$ denote the client and $b = 1$ the server, and let C_b and ΔC_b be party b's shares of the cache and

new entries. For a public starting position j, each party computes

$$C'_b = \mathrm{Concat}\big(C_{b,<j}, \Delta C_b, C_{b,\geq j+k}\big).$$

For a cache with T allocated rows, the public write position j and update length k satisfy $0 \leq j \leq j+k \leq T$. Let $c_{b,i}$ denote party b's share of cache row i, and $\delta c_{b,r}$ its share of update row r. The update can be written as

$$\begin{aligned} C_b &= [c_{b,0}, \ldots, c_{b,j-1}, c_{b,j}, \ldots, c_{b,j+k-1}, c_{b,j+k}, \ldots, c_{b,T-1}], \\ \Delta C_b &= [\delta c_{b,0}, \ldots, \delta c_{b,k-1}], \\ C'_b &= [c_{b,0}, \ldots, c_{b,j-1}, \delta c_{b,0}, \ldots, \delta c_{b,k-1}, c_{b,j+k}, \ldots, c_{b,T-1}]. \end{aligned}$$

Equivalently, for $0 \leq i < T$,

$$c'_{b,i} = \begin{cases} \delta c_{b,i-j}, & j \leq i < j+k, \\ c_{b,i}, & \text{otherwise.} \end{cases}$$

Because j, k are public, each party implements this update by local slicing and concatenation. The piecewise expression specifies the result without requiring secure comparisons. For example, with $j = 2, k = 2, T = 5$, $[c_{b,0}, c_{b,1}, c_{b,2}, c_{b,3}, c_{b,4}] \longrightarrow [c_{b,0}, c_{b,1}, \delta c_{b,0}, \delta c_{b,1}, c_{b,4}]$.

Appending to the logical history preserves all existing valid entries. In preallocated storage, this writes the new entries into the next unused rows. If the cache instead stores only the valid history and grows dynamically, the append concatenates ΔC_b at the end, with an empty final slice.

Prefill of private inputs uses $(j, k) = (m, n)$, giving valid positions $[0, N)$. At decoding step t, the current token's KV is inserted at $j_t = N + t$, giving valid positions $[0, N + t + 1)$. Here, $t = 0$ is the first token feedback step after prefill. Each layer writes its current KV before attention reads it. The attention mask excludes unused storage.

Secret-index comparison. One alternative update path hides j by comparing it with every cache row i. For a new row v, it computes

$$[\![C'_i]\!] = \mathrm{SecSelect}\big([\![j = i]\!], [\![v]\!], [\![C_i]\!]\big).$$

This path incurs secure comparisons and conditional selections. It is an implementation choice, not a requirement of 2PC. MOSAIC avoids these operations because the positions are determined by disclosed lengths and progress. The update still involves local memory access and may require data copying.

Joint attention. The following equations express the logical attention computation in secret-shared form. Operand representations and required conversions follow the existing backend protocols. For one head, omit the layer index and let Q_U contain current queries. Let K_U, V_U contain the available KV for private inputs and generated history, including current entries. The score computation is

$$[\![Z_P \mid Z_U]\!] = \mathrm{SecMatMul}\big([\![Q_U]\!], [\![[K_P; K_U]]\!]^\top\big)/\sqrt{d}.$$

Joint normalization gives

$$[\![A_P \mid A_U]\!] = \mathrm{SecSoftmax}\big([\![Z_P \mid Z_U]\!] + M\big).$$

Value aggregation computes

$$[\![O_U]\!] = \mathrm{SecMatMul}\big([\![A_P \mid A_U]\!], [\![[V_P; V_U]]\!]\big).$$

Here, d is the head dimension, the semicolon joins sequence rows, and $\mid$ joins score or weight columns. The public mask M excludes future positions and unused storage. During batched prefill of private inputs, it also excludes later private input tokens for each query. All visible entries share the same row-wise normalization. Scores for public and private input positions are not normalized separately.

A.2 Context Preservation and Numerical Correspondence

Causal attention excludes future positions, while position-wise sublayers act separately on each token (Vaswani et al., 2017). KV reuse retains previously computed states (Zheng et al., 2024). The following lemma and proposition concern these dependencies rather than numerical equality.

Lemma 1 (Prefix state independence)**.** *Consider deterministic inference in a causal Transformer with position-wise residual, normalization, and feed-forward operations. Fix the model parameters and global positions, with positional transformations independent of later tokens. For the input layout $X = P\|S$, the hidden states and KV tensors for the public input P at every layer depend only on P, the parameters, and positions.*

Proof. Initial representations of public inputs have this property. Assume it holds at one layer. At $i < m$, causal attention uses only positions $j \leq i < m$. Projections and position-wise operations preserve the property at the next layer. Induction proves the claim. □

Full-input reference inference evaluates the complete input without moving computation on public inputs outside secure execution. The following proposition states the resulting attention dependencies.

Proposition 1 (Full-context preservation)**.** *Assume the conditions of Lemma 1. MOSAIC uses the reference parameters, global positions, causal masks, and compatible per-layer cache layouts. It supplies per-layer KV for public inputs and correctly maintains KV for private inputs and generated history. For the same input and fixed generation history, each query during private input prefill or decoding accesses the same causally visible KV positions with joint normalization, preserving full causal attention dependencies.*

Proof. Lemma 1 permits independent computation on public inputs. During prefill of private inputs, each layer adds their KV before attention. Matching global positions and masks expose all public input positions and exactly the causally visible private input positions. The joint attention in Appendix A.1 normalizes their scores together and aggregates their values. After prefill, the caches cover $[0, N)$. At step t, the fixed history supplies y_t at position $N + t$. Each layer retains prior entries and adds the current KV, so the query accesses exactly $[0, N + t + 1)$. Induction over layers and decoding steps establishes the same causal dependencies. □

Numerical correspondence. Numerical agreement with full-input execution under the same backend requires the imported KV, after encoding and representation conversion, to represent the same encoded per-layer values as that execution. The remaining operations must use the same numerical semantics, including parameter encoding, scales, function approximations, truncation, and accumulation order (Lu et al., 2025). Any numerical randomness must also be matched.

For a fixed input and generation history, the current token representations and imported cache values are equal under these conditions. At each layer, equal inputs to the same numerical operations produce equal queries, new KV, attention outputs, and subsequent hidden states. The cache update preserves this equality for the next decoding step. Induction over layers and steps therefore gives equal logits. Matching token selection and stopping rules, together with matching sampling randomness when used, extends the correspondence to generated sequences. Encoding locally computed KV alone does not establish these conditions. Fixed-point encoding and operator approximations may still produce differences from native floating-point inference.

The input is fixed throughout these comparisons. For placeholder-based adaptation, Proposition 1 applies to the adapted sequence, rather than asserting equivalence between the original and rewritten requests. Section 4.4 evaluates rewriting separately.

A.3 Privacy Argument

We use the static semi-honest model in Section 2 and rely on the security of the backend's input, operator, and conversion protocols. These protocols compose under the stated public information and output rules. Local computation on public inputs uses only P and W, both already known to the server. This local computation sends no protocol messages. Model parameters and KV computed from public inputs enter secure execution as server-private inputs, while S enters as a client-private input. Subsequent KV, intermediate values, logits, and selected tokens remain protected, with output reconstruction restricted to the client.

Once compatible shares of new KV are available, each public-position update is a deterministic function of a party's existing shares and the disclosed positions and shapes. These shares include both the current cache shares and the shares of the new entries. The shared rows are retained or

replaced at the same public positions on both sides, preserving the represented cache. No KV value is opened, and the local transformation creates no additional transcript. It therefore adds no information beyond the party's existing view and public metadata. Under the assumed composition of the backend protocols, the simulator for either corrupted party can extend the simulated view by applying the same local update to the simulated cache shares, shares of the new entries, and public metadata. Applying this efficient deterministic computation preserves computational indistinguishability between the real and simulated views. Under the security and composition assumptions for the underlying protocols, a party's protocol view reveals no additional private information beyond its own inputs, the stated public information, and its authorized output.

A.4 Efficiency and Cost Accounting

For $N = m + n$, full-input prefill has $N(N+1)/2$ causally valid attention scores per head and layer. With positions indexed from zero, a query at position i has $i + 1$ causally visible keys. Queries from private inputs require $\sum_{i=m}^{N-1}(i+1) = mn + n(n+1)/2$ scores. Local computation on public inputs removes $m(m+1)/2$ attention scores among public input positions from secure evaluation while retaining the mn scores between private queries and public keys. Projections, normalization, and feed-forward operations for public inputs also move to local computation. These are logical workload counts. Actual costs depend on tensor layouts, secure protocols, and compilation.

With KV reuse, decoding step t processes one new position and reads $N + t + 1$ valid history positions. This count includes the current position, whose KV is written before attention as described in Appendix A.1. KV reuse avoids recomputing historical states while retaining attention to public input positions. Public-position updates avoid comparisons and selections for hiding write positions. Once compatible shares of new KV are available, each party performs the update locally, and the update itself sends no messages. Generating new KV, converting representations, and moving local data remain separate costs.

The Full2PC baseline in Section 4 processes the full input securely and recomputes historical states. Its comparison with MOSAIC measures the combined effect of local public KV computation, private KV input, cache reuse, and public-position updates. The all-private controls in Section 4.3 separately compare execution with and without KV reuse, and cached execution with public and secret decode indices.

Complete request costs include local preparation, protected KV input, compilation, secure prefill, decoding, cache maintenance, representation conversion, and result retrieval. KV input costs include encoding, input processing and transfer, and conversions required to use the imported tensors. Each cost is counted once, including conversions performed during later operations. The request timer includes input sharing. Execution traffic excludes input sharing and deployment RPC, as specified in Section 4.2. Initial cache allocation is excluded from the timing of Full2PC with KV reuse and from the cached timing in the KV-reuse control, as specified in Sections 4.2 and 4.3.

B Additional Evaluation Results

This appendix supplements the effectiveness, efficiency, ablation, and robustness evaluations in Section 4.

B.1 Effectiveness Evaluation Settings

Models and references. The main task evaluation uses GPT-2 Base, SmolLM2-135M, and SmolLM2-360M. The two SmolLM2 models use the Instruct checkpoints. The separate private input ratio experiment uses GPT-2 and SmolLM2-135M-Base. N, C, and M are defined in Section 4.

Language modeling. WikiText-2 PPL aggregates token log-probabilities across texts before exponentiation, excluding padding. There are 2,201 scoring tokens for GPT-2 and 2,330 for each SmolLM2 model.

Text simplification. For ASSET, N, C, and M generate texts independently. We report SARI and exact match between complete generated texts. The secure generation budget is the compatible plaintext EOS position plus eight tokens, capped at 128.

Private input ratios. The rightmost PIQA group in Figure 3(a) uses the same ten questions per model at private input ratios of 25%, 50%, and 75%.

Hidden-state comparisons. Figure 3(c) uses a 25% private input ratio: GPT-2 on C-Eval and HellaSwag, and SmolLM2-360M-Instruct on HellaSwag and CoLA. Each model–dataset pair includes one question with matching N and M predictions and one with differing predictions. N–C, C–M, and N–M relative L2 errors are measured at hidden-state indices 1, 6, and 12 for GPT-2 and 1, 16, and 32 for SmolLM2-360M-Instruct, averaging candidate errors within each question.

B.2 Additional Efficiency Results

Tables 2–4 supplement the GPT-2 measurements in Section 4.2. Full2PC in these tables does not reuse KV.

Table 2: GPT-2 request time and execution traffic at $N = 64, n = 16, G = 4$. Time is mean $\pm$ sample standard deviation over three runs. Traffic is the mean sum of bytes sent and received by rank 0 during execution, excluding input sharing and deployment RPC.

Execution	Request time (s)	Traffic (GB)
Full2PC	266.702 ± 1.346	13.081
MOSAIC	60.441 ± 0.597	2.086

Table 3: GPT-2 workload sweeps. Each setting uses one observation per execution method, except $G = 4$, which uses the three-run means in Table 2, consistent with Figure 4(e). Dashes indicate omitted traffic values, not zero traffic.

				Time (s)		Traffic (GB)	
Sweep	N	n	G	Full2PC	MOSAIC	Full2PC	MOSAIC
Input length	32	16	1	47.054	35.753	1.527	0.964
	64	16	1	68.406	36.407	2.873	1.048
	128	16	1	114.908	37.980	6.567	1.207
Private fraction	64	16	1	68.406	36.407	—	—
	64	32	1	68.406	49.317	—	—
	64	48	1	68.406	61.508	—	—
Generation length	64	16	1	68.406	36.407	—	—
	64	16	4	266.702	60.441	—	—
	64	16	8	541.706	91.820	—	—

Table 4: Single GPT-2 observations under shared-loopback traffic shaping on one host, $N = 64, n = 16, G = 4$. Bandwidth and RTT are nominal settings.

Bandwidth (Mbps)	RTT (ms)	Full2PC (s)	MOSAIC (s)	Ratio
1000	1	362.78	89.71	4.04×
100	1	1280.38	235.19	5.44×
1000	10	579.95	241.16	2.40×

Compilation reuse. One cold/warm pair takes 60.13/52.10 s for MOSAIC and 264.03/257.48 s for Full2PC, with matching outputs within each pair. Cold runs include compilation (2.424 s for MOSAIC and 2.250 s for Full2PC); warm runs reuse the compiled graph with zero compilation time. MOSAIC recomputes public KV in both cases. Other runtime variation contributes to the remaining request-time differences.

SmolLM2-135M-Base. Table 5 reports supplementary system measurements; the corresponding controls and prediction differences appear in Appendix B.3. A separate single-observation sweep

at $N = 64, G = 1$ gives MOSAIC times of 73.999, 93.483, and 115.704 s at $n = 16, 32, 48$, respectively; Full2PC takes 122.012 s.

Table 5: SmolLM2-135M-Base† system measurements at $N = 64, n = 16, G = 4$. Time is mean ± sample standard deviation over three runs. † marks measurements using RoPE base 10000; the Base checkpoint specifies 100000. This note applies only to marked measurements. Control prediction differences are reported in Table 9.

Execution	Time (s)	Traffic (GB)
Full2PC	523.756 ± 0.533	35.657
MOSAIC	141.740 ± 0.174	5.918

HLO profiles. Table 1 reports HLO profiles, not lower-level kernel counts; one HLO may invoke several protocol operations. Parent `while` time can include child operations, so aggregates cannot be summed across levels. The final request time and execution traffic use independent request counters. The `dynamic_update_slice` row measures updates to compatible cache representations, excluding new KV computation, private KV input, and representation conversion. Its zero recorded communication applies only to the update.

B.3 Additional Ablation Results

GPT-2 controls. Tables 6 and 7 report the all-private controls in Figure 5(a). The reuse control uses public cache positions and excludes initial cache allocation from cached timing. Its fixed-history repetitions are separate from the single greedy Full2PC run with KV reuse in Figure 4(a). The index control uses a separate input length and compares public decode positions with the tested address-hiding path; both variants reuse KV and prefill at public position zero.

Table 6: GPT-2 cache-reuse control, $N = n = 64$. Time reports mean ± sample standard deviation over three fixed-history repetitions. Agreement counts replay positions across repetitions.

G	No reuse (s)	Cached (s)	Ratio	Agreement
4	267.212 ± 0.544	93.044 ± 0.684	2.87×	12/12
8	542.960 ± 1.143	124.368 ± 0.695	4.37×	24/24

Table 7: GPT-2 decode-index control, $N = n = 32, G = 4$. Both variants reuse KV and prefill at public position zero. Time uses three-run means and sample standard deviations. Fixed-history predictions agree at 12/12 positions.

Decode index	Time (s)	Execution traffic (GB)
Secret	1364.986 ± 13.386	70.985
Public	76.815 ± 5.771	2.621

SmolLM2-135M-Base controls. Table 8 reports reuse and index controls for the marked Base configuration. Table 9 counts argmax agreement at fixed-history positions across three repetitions, not independent questions. The GPT-2 controls agree at all tested positions; the Base controls show prediction differences whose cause remains unresolved. These output comparisons do not establish equality of encoded cache updates.

Update microbenchmarks. Synthetic single-layer GPT-2 and SmolLM2-135M KV shapes isolate cache updates without model forward computation or RoPE. Each run performs one K write and one V write. Timing includes share assembly, compilation, and full output retrieval; SPU execution traffic excludes input sharing and retrieval RPC. Public updates record zero SPU execution traffic. At capacities 32, 64, and 128, secret-index traffic is 1.701, 3.402, and 6.804 GB for GPT-2 shapes, and 0.109, 0.218, and 0.436 GB for SmolLM2 shapes.

Table 8: SmolLM2-135M-Base† controls. Time is mean ± sample standard deviation over three runs. † marks measurements using RoPE base 10000; the Base checkpoint specifies 100000. This note applies only to marked measurements. Prediction agreement is reported in Table 9.

Comparison	Execution	Time (s)	Traffic (GB)
Reuse control: $N = n = 64$, public positions			
$G = 4$	No reuse	523.354 ± 0.613	35.657
	Cached	191.443 ± 0.639	11.342
$G = 8$	No reuse	1076.779 ± 0.803	72.569
	Cached	282.839 ± 0.639	15.342
Index control: $N = n = 32, G = 4$, cached execution			
Decode index	Secret	591.296 ± 2.459	18.314
	Public	170.943 ± 0.205	7.633

Table 9: Fixed-history argmax agreement in the KV reuse and public/secret index comparisons. Counts cover generated positions across three repetitions, not independent questions. † marks RoPE base 10000 rather than the Base checkpoint setting of 100000.

Comparison	G	GPT-2	135M-Base†
With / without KV	4	12/12	10/12
With / without KV	8	24/24	21/24
Public / secret index	4	12/12	8/12

Table 10: Synthetic KV updates without model forward computation. Time is mean ± sample standard deviation over three runs and includes the preparation and retrieval steps described above. Capacity counts writable slots. Small standard deviations round to 0.000 s.

KV shape	Capacity	Public index (s)	Secret index (s)
GPT-2	32	0.123 ± 0.001	18.562 ± 0.348
	64	0.139 ± 0.001	27.384 ± 0.175
	128	0.160 ± 0.000	46.009 ± 0.810
SmolLM2-135M	32	0.116 ± 0.003	3.561 ± 0.028
	64	0.117 ± 0.000	4.127 ± 0.045
	128	0.128 ± 0.002	5.513 ± 0.018

Four checks per shape cover the first and last slots at capacity 64. Maximum absolute differences from a floating-point reference are 3.05101×10^{-5} for GPT-2 shapes and 3.05027×10^{-5} for SmolLM2-135M shapes. These checks do not establish exact equality of encoded cache values.

B.4 Input Adaptation and Sample Coverage

SmolLM2-360M-Instruct uses the official chat template. L2 replaces a span from the middle of the question, containing roughly one quarter of its words, with `[PRIVATE_1]`; the private suffix maps this placeholder to the original span. L3 replaces candidate content and appends a private mapping containing the removed text. The word fraction is distinct from a fixed private token ratio.

Each full candidate is scored by mean token log-probability under teacher-forced histories, using a separate secure pass with public KV computed through the compatible plaintext numerical path.

Table 11 covers the 24 L2 and 23 L3 questions with secure inference results. Within each layout, all methods use the same questions; Figure 5(b) uses the common 23.

Table 11: Input adaptation results on questions with secure inference results. Within each layout, native plaintext, compatible plaintext, and MOSAIC are compared on the same questions. Prediction consistency and exact match are reported as matching predictions over evaluated questions.

Metric	L2 Question span	L3 Candidate content
Questions with secure results	24	23
Native correct	13/24	11/23
Compatible correct	13/24	11/23
MOSAIC correct	13/24	12/23
MOSAIC prediction consistency with native plaintext	22/24	22/23
MOSAIC exact match against compatible plaintext	24/24	22/23

Across the 24 L2 questions, native-to-secure prediction changes include one correct-to-incorrect and one incorrect-to-correct change. On the common 23, L2 has one prediction difference and MOSAIC answers one more question correctly than native plaintext. L3 has one incorrect-to-correct change. Section 4.4 reports costs over the common 23 questions, summing both candidate calls per question, and specifies their timing and traffic boundaries.

C Discussion

Application conditions. The public prefix is designated by the client according to its confidentiality requirements (Section 2). Under the stated causal-model assumptions, its KV can be computed locally without access to the private suffix. Public tokens that follow private content may depend on that history, so disclosure of a token alone does not make its KV independently computable. Optional rewriting in Section 3.1 provides the required public-prefix/private-suffix layout.

Interpreting efficiency gains. The comparison with Full2PC without KV reuse measures the combined effect of local public KV computation, private KV input, KV reuse, and updates at public positions. In Section 4.2, MOSAIC request time includes local preparation and KV import. The additional Full2PC measurement with KV reuse in Figure 4(a) was collected separately and uses a different timing boundary; it does not isolate the contribution of local public KV computation. The all-private controls in Section 4.3 separately examine KV reuse and public versus secret decode indices. Their speedups apply to their respective comparisons and cannot be multiplied to recover the overall gain. Appendix A.4 specifies the associated cost accounting.

Input rewriting and task quality. On a fixed rewritten input, we measure MOSAIC's prediction consistency with native plaintext and exact match with compatible plaintext. Comparing native plaintext task quality across the original and rewritten layouts assesses the effect of input adaptation. Section 4.4 reports these comparisons and shows that rewriting can change task quality. The analysis of context preservation and numerical correspondence in Appendix A.2 applies to the adapted sequence. Preserving its causal attention dependencies therefore does not establish semantic or output equivalence between the original and rewritten requests.

Evaluation scope. The measured gains concern the evaluated models and workloads under the stated SPU configurations. System measurements use batch size one with both parties on one CPU host; network variants use shared-loopback traffic shaping (Section 4 and Appendix B.2). KV reuse avoids recomputing historical states while retaining attention to all causally visible positions. Performance for larger models, longer contexts, or separate machines requires corresponding measurements.