ALIASSHARE: A PLUG-AND-PLAY MODULE FOR EFFICIENT EMBEDDING OPERATIONS IN MPC WORKFLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose AliasShare, a method designed to accelerate embedding operations in secure multi-party computation (MPC) for private information retrieval (PIR) in ML applications. AliasShare applies a one-to-many mapping for high-frequency tokens along with shuffling in the offline stage, reducing the need to transmit full embeddings and thereby lowering client's online upload communication overhead by $d\times$ (d represents embedding dimension). Meanwhile, it avoids performing costly secret embedding multiplication on the server, mitigating server computation overhead to plaintext-level. Our code is available at `https://anonymous.4open.science/r/AliasShare-main-BC51`.

1 INTRODUCTION

Existing MPC-LLM inference frameworks encounter substantial computational and communication bottlenecks in practical deployments (Hao et al., 2022; Pang et al., 2023). A granular breakdown of the Transformer architecture's (Kei & Chow, 2025) execution overhead under MPC reveals that the cryptographic conversion and transmission of the embedding layer impose a critical latency penalty at the very initialization stage. In realistic edge-computing environments, a massive asymmetric disparity exists between the upstream bandwidth of clients (e.g., residential networks, mobile devices) and the backbone network bandwidth of cloud servers, often spanning a $50\times$ to $100\times$ gap. This physical constraint forces existing MPC inference protocols into an inherent trade-off dilemma:

- The first paradigm requires clients to locally compute and directly upload high-dimensional secret shares of embedding vectors (typically 768 to 4,096 dimensions denoted by d), as instantiated in certain configurations of Mosformer (Cheng et al., 2025) and MPCFormer (Li et al., 2022). Consequently, the highly constrained client uplink bandwidth becomes the primary *latency bottleneck*, incurring prohibitive Time-to-First-Token (TTFT) latency.
- The second paradigm mitigates client-side communication overhead by allowing clients to upload lightweight token ID secret shares, as exemplified by SHAFT (Kei & Chow, 2025) and PUMA (Dong et al., 2023). However, this approach merely shifts the burden from communication to computation. In the encrypted domain, to locate the embedding vector corresponding to a specific token ID, the servers must construct a high-dimensional one-hot selection matrix (of dimensions $V \times m$, where V denotes the vocabulary size and m represents the sequence length) and perform a full-scale matrix multiplication against the massive embedding matrix (of dimensions $d \times V$) which takes up 19% of the full computation overhead (Kei & Chow, 2025). As modern LLM vocabularies expand to tens of thousands of tokens, such high-dimensional ciphertext matrix multiplications lead to excessive computational overhead.

To alleviate this friction, recent works have attempted to introduce obfuscation techniques (e.g., ObfusLM (Lin et al., 2025)); however, since they are designed for single-party private inference, they rely on injecting differential privacy noise to thwart statistical attacks. This causes an irreversible degradation in model generation accuracy (Zhou et al., 2023; Yue et al., 2021; Chen et al., 2023; Plant et al., 2021). Furthermore, the defensive boundaries of these obfuscation schemes remain vulnerable when subjected to advanced statistical exploits, including Token Frequency Match-

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.

ing Attacks (TFMA) (Zanella-Béguelin et al., 2020), Substitution Decipherment Attacks (SDA) (Kambhatla et al., 2018), and Embedding Inversion Attacks (EIA) (Song & Raghunathan, 2020).

To overcome the challenges, we present **AliasShare**, an efficient, plug-and-play secure embedding acceleration protocol tailored for distributed MPC-LLM inference. Our core insight stems from a fundamental cryptographic property in distributed multi-party settings: *since the model could be secret shared across independent servers in MPC, the permutation vector of a vocabulary shuffling* (exhibiting a cryptographic complexity of V^m) *is naturally protected by the distributed ciphertext matrix* (Wang et al., 2025; Zhang et al., 2024b; Sun et al., 2023). Leveraging this crucial observation, **AliasShare** completely eschews accuracy-damaging noise injection, thereby achieving zero loss (0% degradation) in model generation quality. Specifically, **AliasShare** introduces following technical and engineering innovations:

- **In the input obfuscation phase**, model owner shuffles the plaintext token sequence and introduces a frequency-aware, one-to-many *alias mapping bucket* mechanism. Grounded in the intrinsic language statistical properties of *Zipf's Law* (Zipf, 2016), our scheme allocates multiple randomized bucket IDs to high-frequency tokens (e.g., punctuation), while maintaining a one-to-one mapping for the vast majority of low-frequency tokens.
- **In the dynamic interaction phase**, **AliasShare** incorporates an offline dynamic re-shuffle mechanism. Upon the completion of one inference session, model owner reshuffles the active token set prior to the subsequent upload. If the re-shuffling is executed within a window of K ($K < V$) tokens, it introduces an inherent $A_V^K > K!$ matching complexity, effectively blocking the server's computational path to mount SDA-style attacks.

Furthermore, **AliasShare demonstrates robust generalizability across diverse application scenarios**. As a lightweight, plug-and-play cryptographic primitive, it is not only compatible with existing semi-honest MPC frameworks (Hou et al., 2023; Lu et al., 2023; Zhang et al., 2024a), but also generalizes effectively to the **malicious security model** (e.g., Mosformer (Cheng et al., 2025)). Concurrently, this highly efficient secure retrieval mechanism from ciphertext indices to large-scale matrices enables **AliasShare** to generalize to **Private Information Retrieval (PIR)** (Corrigan-Gibbs & Boneh, 2019; Chen et al., 2017; Choubey et al., 2021; Liu et al., 2023) in ML scenarios. This flexibility allows it to be directly deployed in high-concurrency commercial recommendation systems (Mudigere et al., 2022; Gupta et al., 2020; Zhou et al., 2018), or to empower edge-LLM privacy-preserving inference where massive embedding tables must be outsourced to untrusted clouds due to local memory constraints (Jouppi et al., 2023).

The core contributions of this work are as follows:

- **Efficient Design Paradigm:** We identify the limitations of existing single-party obfuscation schemes (Lin et al., 2025; Du et al., 2023; Yue et al., 2021), and propose a distributed, secure embedding obfuscation framework with plaintext-level server computation overhead and client online upload communication overhead.
- **Theoretical Security Guarantee:** Driven by the linguistic properties of *Zipf's Law*, we develop the alias mapping bucket and offline dynamic re-shuffle protocols. These primitives provide security guarantees against TFMA, SDA, and EIA based on information theory, while eliminating noise injection and ensuring zero accuracy loss.
- **Extensive Experimental Validation:** Compared to the baseline Mosformer configuration that directly uploads embedding shares, **AliasShare** slashes the client's uplink communication volume by **32$\times$ to 288$\times$**, effectively mitigating the edge network bandwidth bottleneck. **AliasShare** maintains a total PIR latency at plaintext-level under **61 ms** (on two workstation-class CPUs) which outperforms state-of-the-art (SOTA) PIR scheme **GPU DPF** (Lam et al., 2024) (on two NVIDIA RTX 4090 nodes) by **15$\times$**.
- **Robust Scenario Generalizability:** As a highly generalized cryptographic acceleration module, our solution seamlessly integrates with mainstream semi-honest MPC frameworks (Hou et al., 2023; Zhang et al., 2024a), scales to malicious-security paradigms like Mosformer (Cheng et al., 2025), and further extends to commercial recommendation systems (Mudigere et al., 2022; Gong et al., 2020; Zhao et al., 2020) and cloud-hosted PIR embedding retrieval for mobile applications (Jouppi et al., 2023).

Table 1: Representative network configurations derived from nPerf India 2025–2026 (nPerf, 2026c;a;b) mobile and fixed Internet measurements. The reported latency is used as the RTT in our experiments.

Metric	Weak mobile	Typical mobile	5G mobile	Fixed broadband	Residential WiFi
Downlink (Mbps)	7.43	37.04	74.74	73.50	45.13
Uplink (Mbps)	2.94	6.19	11.44	49.14	39.11
RTT (ms)	83.11	84.31	74.79	54.35	52.79

2 BACKGROUND

Early MPC-based approaches (Cheng et al., 2025; Li et al., 2022) typically required clients to store the embedding matrix locally and directly secret-share the embeddings of the prompt with the server. However, this incurs $O(d \times m)$ upstream communication overhead, where d denotes the embedding dimension and m is the prompt length. As clients in MPC-LLM inference are typically in mobile or home fixed internet environments with **limited uplink bandwidth**, as indicated in Table 1, these methods (**Local Emb.**) lead to large upload delay.

To reduce this, some schemes upload secretly shared Token IDs instead of the embeddings, which largely decreases the communication overhead of the client. Specifically, PUMA and SHAFT both move the embedding operation from the client side into the MPC protocol, denoted as **PUMA-PIR** and **SHAFT-PIR**, respectively.

In **PUMA-PIR**, the client secret-shares the token index id to three computing parties under replicated secret sharing. Given an embedding table $E \in \mathbb{R}^{V \times d}$, the parties first construct a secret-shared one-hot vector by comparing the private index with every public vocabulary position, i.e., $o_i = \Pi_{\mathrm{Eq}}(id, i)$ for all $i \in [V]$. The selected embedding is then obtained by applying the one-hot mask to the secret-shared embedding table using Boolean-arithmetic multiplication, which is equivalent to computing $\sum_i o_i \cdot E_i$. This design preserves the standard Transformer workflow, since the client only provides token IDs rather than locally generated one-hot vectors or embeddings. However, it requires scanning the entire vocabulary and therefore incurs $O(V)$ secure equality tests and $O(Vd)$ secure selection cost per token.

SHAFT-PIR also supports embedding lookup directly from secret-shared token indices, but avoids the vocabulary-wide equality tests used by PUMA. It introduces a preprocessed random index r together with its corresponding secret-shared one-hot vector $\boldsymbol{e}_r$. During the online phase, given a secret-shared token index id, the parties compute and open the masked offset $\delta = id - r$. Since r is uniformly random, revealing δ does not disclose the true token index. The parties then locally cyclically shift $\boldsymbol{e}_r$ by δ positions to obtain the secret-shared one-hot vector $\boldsymbol{e}_{id}$. Finally, the embedding vector is computed as $\boldsymbol{e}_{id} \cdot E$. Compared with PUMA, SHAFT reduces the index-to-one-hot conversion to one online round and removes the $O(V)$ equality tests. Nevertheless, the final oblivious selection is still implemented as a one-hot vector times the full embedding matrix, leading to an $O(Vd)$ secure multiplication cost per token.

Although uploading secret-shared token IDs reduces the client-side upload from $O(md)$ embedding elements to $O(m)$ token IDs, the dominant cost is shifted to the MPC servers. In both PUMA and SHAFT, the final private lookup is realized as a one-hot selection over the full embedding table, which is equivalent to multiplying a secret-shared one-hot vector of length V with the secret-shared embedding matrix $E \in \mathbb{R}^{V \times d}$. Therefore, it incurs $O(Vd)$ secure multiplication through Beaver Triples and, correspondingly, $O(Vd)$ server communication per token. Note that modern LLMs typically use vocabularies ranging from tens of thousands to hundreds of thousands of tokens, e.g., 50K for GPT-2/3 models (Cohen & Gokaslan, 2020), 128K for LLaMA-3 (Dubey et al., 2024), 151K for Qwen (Bai et al., 2023), and 256K for Gemma (Team et al., 2024). As admitted by SHAFT, embedding lookup becomes the core bottleneck and accounts for 19% of the total computation time.

In contrast to embedding lookup for MPC-LLM, **GPU DPF** (Lam et al., 2024) based on Function Secret Sharing (FSS) focuses on industrial recommendation systems, which significantly reduces the server communication but does not eliminate client-side communication and server-side computation. The client generates two compact DPF keys and sends them to two non-colluding servers. This changes the query upload from the $O(d)$ communication of a naive one-hot sharing approach to

$O(\lambda \log V)$, where λ is the security parameter representing AES encryption level. Each server then expands its DPF key into a secret share of a length-V indicator vector and computes a dot product between this vector and the embedding table. Therefore, the server-side computation remains linear in the table size, consisting of $O(\lambda V)$ DPF expansion plus $O(Vd)$ table-selection computation.

3 FRAMEWORK OF ALIASSHARE

AliasShare is a lightweight plug-and-play embedding acceleration framework for MPC-based LLM inference. The core objective of **AliasShare** is to simultaneously reduce the client-side upload communication overhead and eliminate the expensive secure embedding lookup computation on MPC servers, while preserving compatibility with existing 2PC and 3PC inference pipelines.

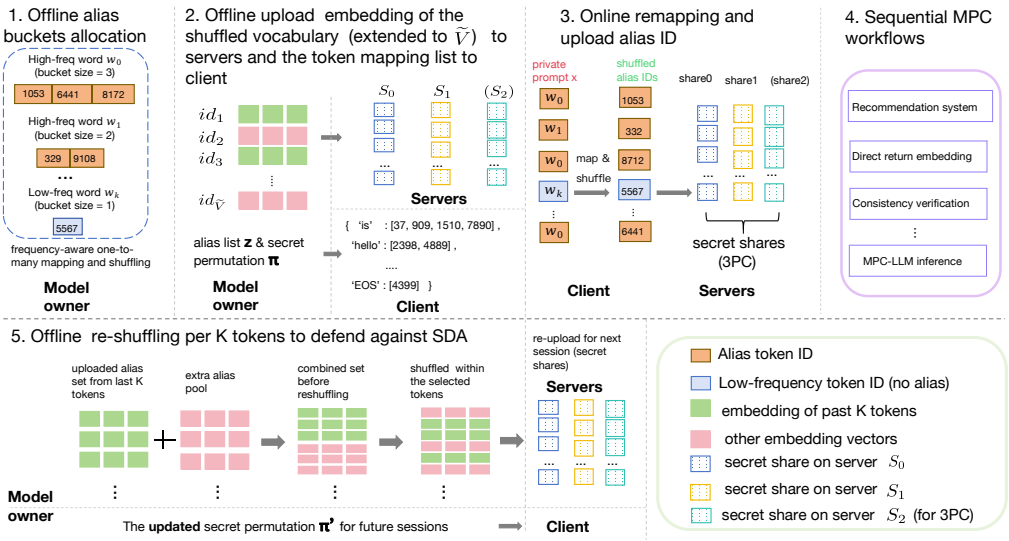

Figure 1: Workflow of AliasShare. Model owner expands and shuffles the embedding vocabulary, uploads secret-shared aliased embedding rows to MPC servers, and constructs frequency-aware alias buckets so that frequent tokens can be mapped to multiple aliases. At online inference time, the client selects one alias ID per token occurrence and sends the secretly shuffled IDs to servers. Servers perform standard embedding lookup to obtain secret shares for the sequential MPC pipeline. To defend against substitution decipherment attacks, model owner periodically re-shuffles the whole embedding table when token counts accumulate to K ($K < V$).

Figure 1 illustrates the overall workflow of **AliasShare**. The framework mainly consists of three partners: model owner, client, and servers. Model owner has full embedding parameters and keeps them secret during MPC deployment and inference on servers, in **AliasShare**, model owner also bears the responsibility to expand and shuffle the embedding table. Servers are MPC computation nodes. While client seeks to protect the confidentiality of its uploaded prompt against potentially curious or compromised servers. The security analysis of **AliasShare** is in Appendix B.

Threat model. We assume a non-collusion setting in which no two parties collude by sharing their private states or protocol transcripts. For example, model owner does not reveal information about alias allocation and permutation to servers, and servers do not exchange their locally held secret-shared embedding tables. Besides collusion, **AliasShare** allows semi-honest adversaries in common 2PC/3PC cases, and could be extended to malicious security setting under 3PC (assuming at most one malicious adversary, see Appendix F).

Algorithm 1: Greedy Marginal-Gain Alias Allocation

Input: Token frequency prior $\{p_i\}_{i=1}^{V}$ and extra alias budget B
Output: Alias counts $\{r_i\}_{i=1}^{V}$ and alias buckets $\{A(i)\}_{i=1}^{V}$

for $i \leftarrow 1$ **to** V **do**
  $r_i \leftarrow 1$;
for $b \leftarrow 1$ **to** B **do**
  Select the token with the largest marginal gain:

$$i^\star = \arg\max_{i \in [V]} p_i \log \frac{r_i + 1}{r_i}$$

  $r_{i^\star} \leftarrow r_{i^\star} + 1$;
for $i \leftarrow 1$ **to** V **do**
  Allocate r_i alias identifiers to token i;
  $A(i) \leftarrow \{a_{i,1}, a_{i,2}, \ldots, a_{i,r_i}\}$;
return $\{r_i\}_{i=1}^{V}$ *and* $\{A(i)\}_{i=1}^{V}$;

3.1 Frequency-Aware Alias Bucket Construction

Let the original embedding matrix be $W \in \mathbb{Z}_q^{V \times d}$, where V denotes the vocabulary size and d denotes the embedding dimension.

Model owner first expands the vocabulary into an enlarged vocabulary space of size $\widetilde{V}$, where $\widetilde{V} \geq V$. The additional rows correspond to alias entries used for high-frequency tokens. The resulting expanded embedding matrix is denoted by $W_E \in \mathbb{Z}_q^{\widetilde{V} \times d}$. Specifically, for a token assigned multiple aliases, **AliasShare** duplicates its embedding row into multiple positions in W_E. Therefore, multiple observable embedding vectors correspond to the same semantic token. Token frequencies approximately follow *Zipf's law*, where only a small number of tokens dominate the overall occurrence probability. **AliasShare** leverages this property to allocate aliases non-uniformly.

For each token i, **AliasShare** assigns an alias bucket $A(i) = \{a_{i,1}, a_{i,2}, \ldots, a_{i,r_i}\}$, where r_i denotes the number of aliases assigned to token i. High-frequency tokens are assigned multiple aliases, while most low-frequency tokens retain only one alias. For example, a frequent punctuation token may correspond to multiple observable token IDs: $A(i) = \{1950, 9813, 2342, ..., 9088\}$.

All aliases inside the same bucket point to duplicated embedding rows in the shuffled embedding matrix. Therefore, although different secret-shared embedding vectors are observed by the servers, they correspond to identical embedding semantics. This design distributes the probability mass of frequent tokens across multiple aliases and significantly flattens the adversary-visible token distribution, thereby mitigating Token Frequency Matching Attacks (TFMA).

Let $B = \widetilde{V} - V$ denote the *extra alias budget*, where $\widetilde{V}$ is the expanded vocabulary size. This budget directly corresponds to the additional embedding rows stored by the MPC servers and therefore represents the server-side memory overhead introduced by **AliasShare**. Given the token frequency prior p_i, **AliasShare** computes the alias counts by solving the following discrete allocation problem:

$$\max_{r_i} \sum_{i=1}^{V} p_i \log r_i \quad \text{s.t.} \quad r_i \in \mathbb{Z}_{>0}, \ \sum_{i=1}^{V} (r_i - 1) \leq B. \tag{1}$$

Since $p_i > 0$ and $\log r_i$ is increasing, the optimal solution uses the full alias budget. **AliasShare** therefore adopts the *greedy marginal-gain allocation algorithm* in Algorithm 1.

Intuitively, the algorithm repeatedly assigns the next available alias slot to the token that yields the largest entropy increase in the adversary-visible alias distribution. Since the marginal gain $p_i \log((r_i + 1)/r_i)$ decreases monotonically as r_i increases, the procedure naturally allocates more aliases to high-frequency tokens and stops expanding low-frequency tokens unless their marginal gain becomes competitive.

In particular, under the standard Zipf case, the resulting discrete allocation preserves the first-order relation $N = \Theta(B/\ln B)$ as derived under continuous relaxation, where N denotes the corresponding cutoff rank. This allows us to choose B from a perspective of balancing security and cost. The optimality of Algorithm 1, as well as its connection to the budget–cutoff relation used to estimate B, is proved in Appendix B. By default, we set the target expanded head range to $N = V/10$. Equivalently, the alias budget B is estimated from the asymptotic relation

$$B/\ln B \approx V/10. \tag{2}$$

3.2 Offline Embedding Shuffling

Vocabulary will be expanded to size $\widetilde{V}$ after construction of aliases, then model owner secretly samples a permutation $\pi : [\widetilde{V}] \to [\widetilde{V}]$ and shuffles the embedding rows as $W_\pi[j,:] = W_E[\pi(j),:]$.

The shuffled embedding matrix is then secret-shared and uploaded to the MPC servers. EIA that depends on embedding information leakage is theoretically impossible to **AliasShare** due to nature of secret share.

3.3 Online Inference with Alias Remapping

During inference, the client first tokenizes the prompt sequence $x = (t_1, t_2, \ldots, t_m)$. For each token t_i, the client randomly samples one alias from its corresponding alias bucket: $z_i \sim \mathrm{Uniform}(A(t_i))$. The remapped alias sequence $z = (z_1, z_2, \ldots, z_m)$

is then secretly transformed to $\pi(z)$ and uploaded to the MPC servers. Note that the shuffling permutation π is unknown to the servers, thus the semantics x behind received $\pi(z)$ remains secret. On the other hand, the servers may still execute potential symbol frequency-based attacks such as TFMA and SDA. In practice, alias expansion and offline re-shuffling help resist SOTA statistical attacks, which is evaluated in Section 4.2.

3.4 Offline Dynamic Re-Shuffling

Although alias mapping mitigates TFMA, a fixed long-term token-alias mapping may still expose sequential statistical patterns exploitable by Substitution Decipherment Attacks (SDA).

To prevent this, **AliasShare** periodically performs offline dynamic re-shuffling.

The entire embedding table including those appearing in the past K tokens will be reshuffled by model owner and transmitted to the servers when token counts accumulate to K. The adversary must solve an additional cross-round matching problem with $A_V^K > K!$ complexity to complete beam search required by SDA. Consequently, **AliasShare** prevents the adversary from concatenating multiple inference sessions into one stable substitution cipher, significantly reducing the effectiveness of SDA-style attacks.

4 Evaluation

We use (embedding layer of) GPT-2, BERT, Llama-3 to evaluate performance, which are most popular models in current MPC-LLM systems, along with Taobao Rec. (Sanagapati, 2020) to represent the PIR scenario in recommendation. We include SST-2, QNLI, and MNLI from GLUE (Wang et al., 2018), as well as Databricks-Dolly-15K (Conover et al., 2023) and Alpaca (Taori et al., 2023) as experimental data sources. The performance baselines include direct local embedding and SOTA methods for 3PC, 2PC, and commercial recommendation system, respectively: PUMA-PIR (Dong et al., 2023), SHAFT-PIR (Kei & Chow, 2025), and GPU DPF (Lam et al., 2024). All experiments are executed on a server workstation equipped with 3 NVIDIA GeForce RTX 4090 GPUs, an Intel Xeon E-2176G 3.70 GHz processor, and 128 GB of memory.

4.1 Computation & Communication Performance

Communication settings. Accurately measuring the empirical running time requires isolating the uncontrollable fluctuations introduced by interprocess communication. According to Table 1, our

Metric	Local Emb.	GPU DPF	SHAFT -PIR	PUMA -PIR	Alias-Share
S-S Rounds	0	0	2	6	0
C-S Volume(B)	8md	128mlogV	8m	12m	8m
S-S Volume(B)	0	0	16(V+m)d	16(V+m)d	0

Table 2: Communication overhead comparison of secure embedding methods. m is input length, d is embedding dimension and V is vocabulary size.

Table 3: Peak memory (MB) and computation delay (ms) comparison of secure embedding frameworks across different models and input lengths.

Framework	m	BERT		GPT-2		LLaMA-3		Taobao Rec.	
		Memory	Delay	Memory	Delay	Memory	Delay	Memory	Delay
GPU DPF	64	554.57	278.26	667.13	516.28	2631.10	5019.99	797.11	294.05
	256	554.57	315.88	667.13	527.37	2631.10	4952.61	797.11	295.94
	1024	556.57	668.79	669.13	1061.46	2633.10	10115.48	801.11	594.20
SHAFT-PIR	64	135.15	0.19	216.42	0.30	2244.23	2.96	1058.48	3.27
	256	159.38	0.74	255.80	1.17	2346.93	11.93	1750.86	13.07
	1024	161.75	3.16	258.17	5.12	2359.52	48.25	1750.96	52.98
PUMA-PIR	64	114.03	0.23	180.59	0.36	2154.75	3.18	419.37	1.44
	256	122.43	0.80	194.05	1.27	2191.45	13.54	650.08	5.67
	1024	124.80	3.04	196.60	4.96	2204.04	56.44	650.18	22.65
AliasShare	64	143.09	0.035	265.25	0.035	3967.29	0.035	228.59	0.035
	256	143.09	0.035	265.25	0.034	3967.29	0.041	228.59	0.034
	1024	143.09	0.036	265.25	0.036	3967.29	0.117	228.59	0.034

reported results are based on the following network settings: a mobile network for client with 10 MBps uplink bandwidth, 50 MBps downlink bandwidth, and 60 ms RTT latency, while a local-area network (LAN) with (1 GBps, 1 GBps, 1ms) or a wide-area network (WAN) with (100 MBps, 100 MBps, 20 ms) for servers.

Computation latency. We evaluate the PIR inference performance of **AliasShare** against baselines under various input lengths (denoted by m) and models. The results are summarized in Table 3. **AliasShare** only requires index lookup and achieves **76**$\times$ to **755**$\times$ computation acceleration, which also indicates the throughput acceleration lower-bound as **AliasShare** incurs negligible input-dependent memory cost as indicated by Table 3.

Overall delay. We give server communication rounds and volume in Table 2, which enables the theoretical calculation of communication latency of baseline methods. The total running time is calculated as: `computation time + C-S communication volume/client uplink bandwidth + S-S communication volume/server bandwidth + RTT× rounds`. We give the C-S communication volumes, S-S communication volume and end-to-end latency performance on LAN/WAN of **AliasShare** along with other baselines in Figure 2 (m is uniformly set to 1024). **AliasShare** achieves plain-text level communication reduction with minimal C-S volume and S-S volume, and accelerates end-to-end PIR inference by **1.42 ×** to **56 ×**.

Memory usage. Table 3 reports the peak memory consumption of different embedding-only protocols. We observe that the memory footprint of **AliasShare** is almost independent of the input length m. This is because **AliasShare** removes the need to construct input-dependent one-hot vectors, equality masks, or DPF evaluation tensors over the vocabulary. For relatively narrow embedding tables such as Taobao Rec., **AliasShare** consumes less GPU memory than SHAFT-PIR and GPU DPF. For large hidden dimensions such as LLaMA-3, its resident embedding-share storage can be larger than some baselines. However, **AliasShare** simplifies the secret embedding lookup into standard table-lookup operation and does not require complex GPU-side tensor computation. Therefore, unlike GPU-DPF, SHAFT-PIR, and PUMA-PIR, **AliasShare** is naturally suitable to **CPU** deployment, which could reduce the occupation of scarce **GPU** memory.

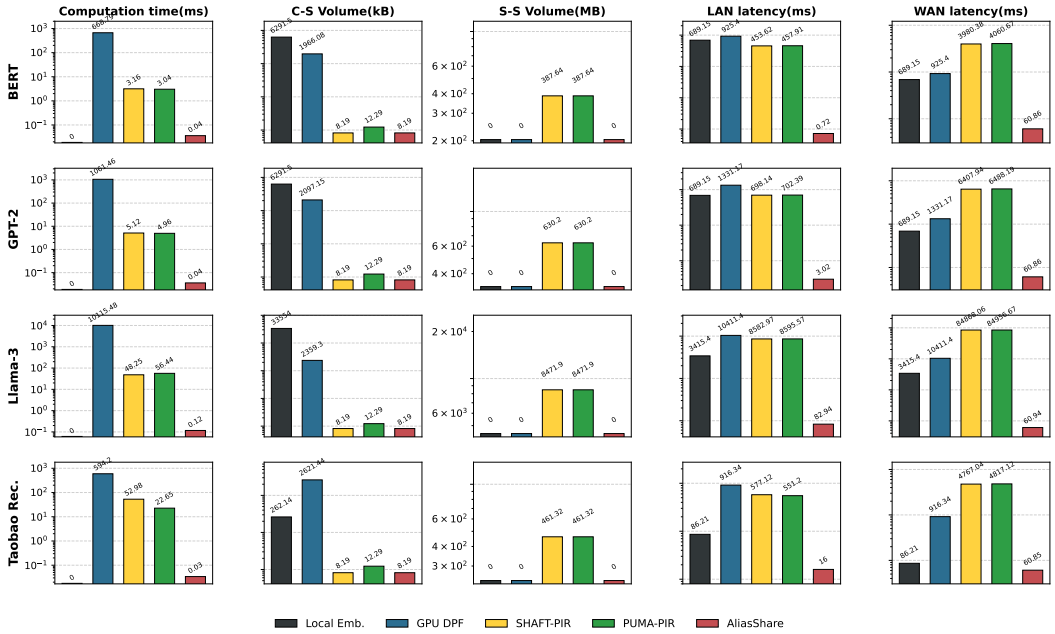

Figure 2: Comprehensive delay comparison of secure embedding methods across different models in terms of computation, communication and end-to-end latency.

Table 4: Defense performance against TFMA. We follow the public/private dataset setting of ObfusLM. Lower values indicate stronger defense.

Private/Public Dataset	Defense	Private Prop.	Top-1(%)	Top-3(%)	ROUGE-L
SST-2/QNLI	ObfusLM	50%	2.13	11.03	2.94
		100%	23.67	92.13	17.54
	AliasShare	50%	**0.13**	**0.57**	**0.67**
		100%	**0.15**	**0.84**	**1.02**
Alpaca/Dolly	ObfusLM	50%	10.53	26.60	7.71
		100%	40.16	98.73	32.10
	AliasShare	50%	**0.22**	**0.94**	**1.33**
		100%	**0.48**	**1.87**	**2.19**

4.2 Security & Utility Performance

Defense against TFMA. TFMA assumes that the attacker has access to a public corpus whose distribution is close to the client's private corpus. The attacker estimates token-frequency statistics from the public corpus and matches the most frequent observed obfuscated identifiers to the most frequent plaintext tokens. Following the setting of ObfusLM, we test two private/public dataset pairs: QNLI is used as the public dataset while SST-2 is used as the client's private dataset, and Databricks-Dolly is used as the public dataset while Alpaca is used as the client's private dataset.

For TFMA, we report Top-1, Top-3, and ROUGE-L recovery scores. Top-1 and Top-3 measure whether the original token appears in the attacker's top recovered candidates, while ROUGE-L measures the sequence-level similarity between the recovered text and the private text. Lower values indicate stronger privacy protection. As shown in Table 4, **AliasShare** substantially reduces frequency-based recovery ability. This is because high-frequency tokens are split into multiple aliases, which dilutes their empirical frequencies and breaks the direct frequency correspondence.

Defense against SDA. Substitution Decipherment Attacks (SDA) exploit the sequential structure of the adversary-visible identifiers. The attacker treats the observed identifier sequence as a substi-

Table 5: Defense performance against SDA measured by ROUGE-L recovery score. Lower ROUGE-L indicates stronger defense.

	SST-2	QNLI	MNLI	Alpaca		SST-2	QNLI	MNLI	Alpaca
ObfusLM	73.69	27.52	27.47	35.16	**AliasShare**	**2.11**	**0.32**	**0.24**	**0.61**

Table 6: Utility performance comparison. AliasShare can be losslessly integrated into MPC workflows, while obfuscation-based methods trade model utility for privacy.

	Plaintext	DP-Forward	SANTEXT+	CUSTEXT+	ObfusLM	AliasShare+PUMA	AliasShare+SHAFT
Type	Plain inference	Obfuscation	Obfuscation	Obfuscation	Obfuscation	MPC	MPC
SST-2 Acc%	92.88	52.57	83.72	79.13	88.07	**92.31**	**91.97**
QNLI Acc%	90.74	49.49	81.16	77.76	86.52	**90.70**	**90.31**

tution cipher and searches for a decoded token sequence with high language-model likelihood. We follow the attack methodology used in ObfusLM. The attacker first constructs a candidate token set for each observed obfuscated token and then performs beam search over possible plaintext sequences. Each candidate sequence is scored by a neural sequence scorer. Specifically, we fine-tune a `bert-base-uncased` model on SST-2 as a binary sequence classification model, where original SST-2 sentences are treated as positive samples and corrupted sentences generated by randomly replacing tokens with candidate aliases are treated as negative samples. During attack inference, the fine-tuned BERT scorer is used to rank candidate plaintext sequences generated by beam search.

Both input length m and offline re-shuffling window size K are set to 1024 in evaluation. We evaluate SDA on prompts from SST-2, MNLI, QNLI, and Alpaca and report ROUGE-L between the recovered sequence and the original private sequence. As shown in Table 5, **AliasShare** achieves lower recovery similarity than ObfusLM under the same SDA setting. This indicates that alias splitting and offline re-shuffling successfully weaken the stable semantic substitution relationship.

Utility Performance. We further evaluate whether **AliasShare** affects the model utility when integrated into existing MPC-based LLM inference workflows. Unlike obfuscation-based defenses, **AliasShare** does not perturb token embeddings or modify the Transformer computation. Therefore, once the correct aliased embedding is selected, the following MPC inference pipeline remains identical to the original one.

We integrate **AliasShare** with two representative MPC workflows, PUMA and SHAFT, and evaluate them on GPT-2 with baselines depending on obfuscation. Obfuscation-based methods protect privacy by injecting noise, which inevitably creates a trade-off between privacy strength and task utility. Stronger obfuscation generally improves security but also makes the model harder to fine-tune or infer accurately. As shown in Table 6, obfuscation-based methods suffer from different degrees of utility degradation because their privacy protection relies on modifying the model input representation. For example, DP-Forward (Du et al., 2023) significantly reduces the task accuracy, while SANTEXT+ (Yue et al., 2021) and CUSTEXT+ (Chen et al., 2023) also introduce non-negligible accuracy loss. ObfusLM achieves better utility by synthesizing obfuscated embeddings, but its accuracy is still lower than **AliasShare**.

5 Conclusion

We presented AliasShare, a plug-and-play module for efficient embedding operations in MPC workflows. By combining frequency-aware alias allocation, vocabulary shuffling, and dynamic re-shuffling, **AliasShare** avoids costly secure embedding lookup while reducing client-side upload communication. **AliasShare** provides theoretical security protection without introducing noise or degrading model accuracy. It is compatible with existing MPC modules. Experiments show that **AliasShare** reduces client upload by $32\times$ to $288\times$, accelerates secret embedding lookup computation by over $76\times$, and improves end-to-end PIR inference by $1.42\times$ to $56\times$.

AI USE STATEMENT

We have reviewed all AI-assisted work. For the manuscript, all AI-assisted text improvement was carefully reviewed and revised by the authors to ensure its accuracy, clarity, and consistency with the actual methods and experimental results. We also verified that the cited claims and descriptions accurately reflect the corresponding literature and our experimental findings. We take responsibility for the final content of this work, including all text, mathematical claims, proofs, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made substantial efforts to ensure the reproducibility of the results presented in this work. For the theoretical results, all assumptions are explicitly stated in the main text and the appendix, and complete proofs are provided in Appendix B. Specifically, Theorem 1 and its proof establish the information-theoretic security of the shuffling permutation against Embedding Inversion Attacks (Appendix B.1); the resistance to Token Frequency Matching Attacks is analyzed via the KL-divergence formulation and the greedy alias-allocation optimality proof in Appendix B.2; and the resistance to Substitution Decipherment Attacks is formalized through the cross-round matching complexity in Appendix B.3. The continuous-relaxation and integer-optimality arguments for the alias allocation problem, including the budget–cutoff relation $N = \Theta(B/\ln B)$, are also fully derived in Appendix B.

For the experimental setup, the models, datasets, and baselines used in Section 4 are described in detail in the main text, including GPT-2, BERT, Llama-3, and Taobao Rec., as well as SST-2, QNLI, MNLI, Databricks-Dolly-15K, and Alpaca. All datasets used in the experiments are publicly available, and the exact baseline configurations, training hyperparameters, and privacy parameters are reported in Appendix A. The network and hardware settings are specified in Section 4.1 and Table 1, and the memory estimation procedure is detailed in Appendix C. The security assumptions underlying our theoretical guarantees, including the non-collusion and semi-honest/malicious settings, are stated in Appendix E, and the integration with 3PC backends under both security models is illustrated in Appendix F.

To facilitate reproduction, the implementation of AliasShare is made available as anonymized source code at `https://anonymous.4open.science/r/AliasShare-main-BC51`. This code includes the frequency-aware alias bucket construction (Algorithm 1), the offline embedding shuffling and dynamic re-shuffling procedures, the online alias remapping and MPC-compatible embedding lookup, and the scripts used to reproduce the communication, computation, and security evaluations in Section 4. The repository also contains the scripts that generate Tables 2–6 and Figures 2–3, so that the reported numbers can be regenerated end-to-end. The code is organized to match the structure of the main paper and appendix, and does not require access to any private data or proprietary models beyond the publicly available checkpoints listed in Section 4.

REFERENCES

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Hao Chen et al. Simple encrypted arithmetic library - seal v2.1. In *Financial Cryptography Workshops*, 2017.

Huimin Chen, Fengran Mo, Yanhao Wang, Cen Chen, Jian-Yun Nie, Chengyu Wang, and Jamie Cui. A customized text sanitization mechanism with differential privacy. In *Findings of ACL*, 2023.

Shiyu Chen, Zhiwei Zeng, et al. On compressing neural ranking models for recommender systems. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.

Ke Cheng, Yuheng Xia, Anxiao Song, Jiaxuan Fu, Wenjie Qu, Yulong Shen, and Jiaheng Zhang. Mosformer: Maliciously secure three-party inference framework for large transformers. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 4124–4138, 2025.

Ashish Choubey et al. Gpu-accelerated private information retrieval. In *International Conference on Applied Cryptography and Network Security (ACNS)*, 2021.

Vanya Cohen and Aaron Gokaslan. Opengpt-2: Open language models and implications of generated text. *XRDS: Crossroads, The ACM Magazine for Students*, 27(1):26–30, 2020.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 4 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm. Dataset: databricks-dolly-15k.

Henry Corrigan-Gibbs and Dan Boneh. Pir with compressed queries and amortized query processing. In *IEEE Symposium on Security and Privacy (SP)*, 2019.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2 edition, 2006.

Ye Dong, Wen-jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Cheng. Puma: Secure inference of llama-7b in five minutes. *arXiv preprint arXiv:2307.12533*, 2023.

Minxin Du, Xiang Yue, Sherman SM Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 2665–2679. ACM, 2023. doi: 10.1145/3576915.3616592. URL https://dl.acm.org/doi/10.1145/3576915.3616592.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yong Gong, Zhuoran Jiang, Kuang Zhao, Qiang Liu, and Wenwu Ou. Edgerec: Recommender system on edge in mobile taobao. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2505–2512, 2020.

Udit Gupta, Shen Hsia, Vikram Saraph, Xiaodong Wang, Brandon Reagen, Gu-Yeon Wei, Hsien-Hsin S Lee, David Brooks, and Carole-Jean Wu. Deeprecsys: A system for optimizing end-to-end at-scale neural recommendation inference. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 782–796, 2020.

Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. In *NeurIPS*, 2022.

Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Wen jie Lu, Cheng Hong, and Kui Ren. CipherGPT: Secure two-party GPT inference. Cryptology ePrint Archive, Paper 2023/1147, 2023. URL https://eprint.iacr.org/2023/1147.

Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th annual international symposium on computer architecture*, pp. 1–14, 2023.

Nishant Kambhatla, Anahita Mansouri Bigvand, and Anoop Sarkar. Decipherment of substitution ciphers with neural language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 869–874, 2018.

Andes YL Kei and Sherman SM Chow. Shaft: Secure, handy, accurate and fast transformer inference. In *NDSS*, 2025.

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 ACM Symposium on Operating Systems Principles (SOSP 23)*, pp. 611–626, 2023.

Maximilian Lam, Jeff Johnson, Wenjie Xiong, Kiwan Maeng, Udit Gupta, Yang Li, Liangzhen Lai, Ilias Leontiadis, Minsoo Rhu, Hsien-Hsin S Lee, et al. Gpu-based private information retrieval for on-device machine learning inference. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 197–214, 2024.

Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P Xing, and Hao Zhang. Mpcformer: fast, performant and private transformer inference with mpc. *arXiv preprint arXiv:2211.01452*, 2022.

Yu Lin, Ruining Yang, Yunlong Mao, Qizhi Zhang, Jue Hong, Quanwei Cai, Ye Wu, Huiqi Liu, Zhiyu Chen, Bing Duan, and Sheng Zhong. Obfuslm: Privacy-preserving language model service against embedding inversion attacks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1160–1174, 2025.

Zirui Liu et al. Privacy-preserving neural network inference via private weighted correlation. In *ACM Conference on Computer and Communications Security (CCS)*, 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. *Cryptology ePrint Archive*, 2023.

Jinglong Luo, Yehong Zhang, Zhuo Zhang, Jiaqi Zhang, Xin Mu, Hui Wang, Yue Yu, and Zenglin Xu. Secformer: Fast and accurate privacy-preserving inference for transformer models via smpc. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 13333–13348, 2024.

Dheevatsa Mudigere, Yuxin Hao, Jianyu Huang, Zhenhao Jia, Andrew Tulloch, Srinivas Sridharan, Xiaodong Liu, Mustafa Ozdal, Jiang Nie, Jongsoo Park, et al. Software-hardware co-design for fast and scalable training of deep learning recommendation models. In *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–15, 2022.

nPerf. Barometer of Fixed Internet Connections in India. `https://media.nperf.com/files/publications/IN/IN-Barometre-Fixed-connections-nPerf-2026_7953.pdf`, 2026a. Fixed Internet measurements in India from Apr. 1, 2025 to Mar. 31, 2026.

nPerf. Barometer of Mobile Internet Connections in India. `https://media.nperf.com/files/publications/IN/IN-Barometre-Mobile-connections-nPerf-2026_7928.pdf`, 2026b. Mobile Internet measurements in India from Apr. 1, 2025 to Mar. 31, 2026.

nPerf. nPerf Internet Speed Test. `https://www.nperf.com/en/`, 2026c. Defines download bitrate, upload bitrate, and latency measurement methodology.

Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. Bolt: Privacy-preserving, accurate and efficient inference for transformers. *Cryptology ePrint Archive*, 2023.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Richard Plant, Dimitra Gkatzia, and Valerio Giuffrida. Cape: Context-aware private embeddings for private language learning. In *EMNLP*, 2021.

Pavan Sanagapati. Taobao ad display click dataset. Kaggle Dataset, 2020. https://www.kaggle.com/datasets/pavansanagapati/ad-displayclick-data-on-taobaocom.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning (ICML)*, pp. 31094–31116, 2023.

Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 377–390, 2020.

Zhichuang Sun, Ruimin Sun, Changming Liu, Amrita Roy Chowdhury, Long Lu, and Somesh Jha. Shadownet: A secure and efficient on-device model inference system for convolutional neural networks. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1596–1612. IEEE, 2023.

SweetPricing. Average mobile application file size statistics. Online Report, 2020. https://sweetpricing.com/blog/index.html#:~:text=Average%20Android%20and%20iOS%20file%20size.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: an instruction-following llama model (2023). *URL https://github.com/tatsu-lab/stanford_alpaca*, 1(9), 2023.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Pengli Wang, Bingyou Dong, Yifeng Cai, Zheng Zhang, Junlin Liu, Huanran Xue, Ye Wu, Yao Zhang, and Ziqi Zhang. Game of arrows: On the ({In-) Security} of weight obfuscation for {On-Device}{TEE-Shielded}{LLM} partition algorithms. In *Proceedings of the 34th USENIX Conference on Security Symposium*, pp. 279–298, 2025.

Xiang Yue, Minxin Du, Tianhao Wang, Yaliang Li, Huan Sun, and Sherman SM Chow. Differential privacy for text analytics via natural text sanitization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3841–3851. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.337. URL https://aclanthology.org/2021.findings-acl.337/.

Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 363–375, 2020.

Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. Secure transformer inference made non-interactive. Cryptology ePrint Archive, Paper 2024/136, 2024a. URL https://eprint.iacr.org/2024/136.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, 52(1):1–38, 2019.

Zheng Zhang, Na Wang, Ziqi Zhang, Yao Zhang, Tianyi Zhang, Jianwei Liu, and Ye Wu. Groupcover: a secure, efficient and scalable inference framework for on-device model protection based on tees. In *Forty-first international conference on machine learning*, 2024b.

Wei Zhao, Dong Xie, Jianyu Jia, Yi Qian, Ruobing Ding, Minghuang Sun, and Peng Li. Distributed hierarchical gpu parameter server for massive scale deep learning ads systems. In *Proceedings of Machine Learning and Systems*, volume 2, pp. 154–173, 2020.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, et al. Deep interest network for click-through rate prediction. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

Xin Zhou, Yi Lu, Ruotian Ma, Tao Gui, Yuran Wang, Yong Ding, Yibo Zhang, Qi Zhang, and Xuanjing Huang. Textobfuscator: Making pre-trained language model a privacy protector via obfuscating word representations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 5459–5473. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.336.

George Kingsley Zipf. *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio books, 2016.

A Evaluation Detail on Obfuscation-style Baselines

We compare AliasShare with four representative obfuscation-style baselines in Section 4.2: ObfusLM (Lin et al., 2025), DP-Forward (Du et al., 2023), SANTEXT+ (Yue et al., 2021), and CUSTEXT+ (Chen et al., 2023). These methods protect client-side textual inputs by perturbing token identifiers, embedding vectors, or intermediate representations before they are exposed to the server. Unlike AliasShare, which preserves the original embedding semantics through aliasing and secret-shared lookup, obfuscation-style methods introduce explicit perturbations and therefore naturally involve a privacy–utility trade-off.

ObfusLM. ObfusLM is an embedding-level obfuscation framework designed for privacy-preserving language-model services against embedding inversion attacks. It first shuffles the vocabulary and the corresponding embedding/model-head weights, and then synthesizes obfuscated embeddings within local embedding clusters. Specifically, ObfusLM clusters semantically similar embeddings using cosine similarity and constructs each obfuscated embedding as a noisy weighted combination of embeddings in the same cluster. The method is based on (k, ϵ)-anonymity, where k controls the anonymity-set size and ϵ controls the noise strength used in embedding synthesis. Following the default evaluation setting in ObfusLM, we set $k = 10$, $\epsilon = 0.1$, and $\beta = 0.99$.

DP-Forward. DP-Forward is an embedding-level differential-privacy baseline that perturbs hidden representations during the model forward pass. Instead of replacing discrete tokens, it injects calibrated noise into model embeddings or intermediate activations so that the server observes noisy representations rather than exact client-side inputs. This design can be directly applied to standard language-model pipelines, but the injected noise may distort semantic information and reduce downstream task utility. Following the ObfusLM evaluation protocol, we search the privacy parameter ϵ over $\{1, 3, 8\}$ and use the selected default setting $\epsilon = 8$.

SANTEXT+. SANTEXT+ is a token-level text sanitization method based on differential privacy. Instead of sending the original token sequence to the server, the client locally replaces tokens with semantically similar substitutes. The replacement probability is determined by embedding-space similarity, so that closer tokens are sampled with higher probability while still providing randomized privacy protection. Following ObfusLM's evaluation setup, we use GloVe embeddings (Pennington et al., 2014) for token replacement, search the privacy parameter ϵ over $\{1, 2, 3\}$, and adopt the selected default value $\epsilon = 3$. For SANTEXT+, we set the sensitive-word percentage to $w = 0.9$ and the sanitization probability for non-sensitive words to $p = 0.3$.

CUSTEXT+. CUSTEXT+ is another token-level differential-privacy baseline. Different from SANTEXT+, it constructs a customized candidate output set for each input token and samples the sanitized token from this local candidate set. This customized design restricts replacement to semantically related tokens and is intended to improve the privacy–utility balance. Following the ObfusLM appendix setting, we use GloVe embeddings for token replacement, search ϵ over $\{1, 2, 3\}$, and use the selected default value $\epsilon = 3$. We set the customized candidate size to $\text{top-}k = 20$.

For fair comparison, we follow the training hyperparameters reported by ObfusLM for these baselines. Specifically, the learning rate is set to 2×10^{-5} and the batch size is set to 32 for ObfusLM, DP-Forward, SANTEXT+, and CUSTEXT+.

B Security Analysis

B.1 Resistance to EIA

Embedding Inversion Attacks (EIA) aim to exploit leaked information in adversary's view to recover embedding. A trivial property is that the secret-shared embedding vectors leak no information about the original vectors. We then show that the uploaded embedding shares do not reveal the model owner-side shuffling parameter π, even when the expanded embedding matrix is known to the servers. This captures the share-level security of **AliasShare**, where model owner uploads secret-shared shuffled embeddings rather than the shuffled embedding matrix itself.

Theorem 1. *Let $W_E \in \mathbb{Z}_q^{\widetilde{V} \times d}$ be the expanded embedding matrix, which is known to all servers. Let π be a secret permutation over $[\widetilde{V}]$ chosen by model owner, and define the permuted embedding matrix W_π row-wise by $W_\pi[j, :] = W_E[\pi(j), :]$. Model owner secret-shares W_π into three matrices*

$W_\pi^{(0)}, W_\pi^{(1)}, W_\pi^{(2)} \in \mathbb{Z}_q^{\widetilde{V}\times d}$ *such that* $W_\pi = W_\pi^{(0)} + W_\pi^{(1)} + W_\pi^{(2)} \pmod q$. *For any single server* S_i*, even if* S_i *knows the full matrix* W_E*, its local share view is statistically independent of* π*, namely,*

$$I(\pi; \mathsf{View}_i \mid W_E) = 0.$$

Thus, the shuffling parameter π *is information-theoretically hidden from any single server.*

Proof. Let $\mathcal{M} = \mathbb{Z}_q^{\widetilde{V}\times d}$. Model owner samples two matrices $W_\pi^{(0)}, W_\pi^{(1)} \xleftarrow{\$} \mathcal{M}$ independently and uniformly at random, and sets $W_\pi^{(2)} = W_\pi - W_\pi^{(0)} - W_\pi^{(1)} \pmod q$.

First consider a server that receives a single additive share, say $W_\pi^{(0)}$. For any fixed W_E, any permutation π, and any matrix $R \in \mathcal{M}$, we have $\Pr[W_\pi^{(0)} = R \mid W_E, \pi] = |\mathcal{M}|^{-1}$. This distribution is uniform over $\mathcal{M}$ and does not depend on π. Hence $W_\pi^{(0)}$ is independent of π conditioned on W_E. The same argument applies to $W_\pi^{(1)}$. Moreover, $W_\pi^{(2)}$ is also uniformly distributed over $\mathcal{M}$, because it is masked by the sum of two independent uniform matrices. Therefore, any single additive share reveals no information about π.

We next consider the replicated-share case, where one server may hold two shares. For example, if a server holds $(W_\pi^{(0)}, W_\pi^{(1)})$, then for any $R_0, R_1 \in \mathcal{M}$, any fixed W_E, and any permutation π, we have $\Pr[(W_\pi^{(0)}, W_\pi^{(1)}) = (R_0, R_1) \mid W_E, \pi] = |\mathcal{M}|^{-2}$. This distribution also does not depend on π, and therefore $I(\pi; (W_\pi^{(0)}, W_\pi^{(1)}) \mid W_E) = 0$.

The same conclusion holds for the other replicated views. Consider $(W_\pi^{(1)}, W_\pi^{(2)})$. For any fixed $R_1, R_2 \in \mathcal{M}$, the condition $W_\pi^{(1)} = R_1$ and $W_\pi^{(2)} = R_2$ uniquely determines the missing share as $W_\pi^{(0)} = W_\pi - R_1 - R_2 \pmod q$. Since $W_\pi^{(0)}$ was sampled uniformly over $\mathcal{M}$, we have $\Pr[(W_\pi^{(1)}, W_\pi^{(2)}) = (R_1, R_2) \mid W_E, \pi] = |\mathcal{M}|^{-2}$, which is again independent of π. The argument for $(W_\pi^{(2)}, W_\pi^{(0)})$ is identical.

Therefore, for every single server S_i, the distribution of View_i is identical for all possible permutations. Equivalently, for any two permutations π_0 and π_1, $\mathsf{View}_i(\pi_0) \equiv \mathsf{View}_i(\pi_1)$, where $\equiv$ denotes identical probability distributions rather than computational indistinguishability. Hence, even an unbounded adversary with full knowledge of W_E cannot distinguish which permutation was used. Consequently, $I(\pi; \mathsf{View}_i \mid W_E) = 0$, which proves the theorem. □

According to Theorem 1, there is no *leakage of information* about item indices of the embedding matrix for a single MPC server. Thus EIAs are not applicable to **AliasShare** even if the original embedding table is public to the servers.

B.2 Resistance to TFMA

Token Frequency Matching Attacks (TFMA) exploit the skewness of natural-language token distributions. Even if token identifiers are randomly permuted, an adversary may still collect the empirical frequencies of the observed identifiers and match them against a public token-frequency prior. Therefore, the information exploited by TFMA is not the identifier value itself, but the frequency distribution exposed to the adversary.

AliasShare mitigates this leakage by combining secret token shuffling with a one-to-many alias mapping for frequent tokens. Instead of mapping each real token to a single observable identifier, **AliasShare** assigns multiple aliases to high-frequency tokens and randomly selects one alias whenever the token appears. As a result, the probability mass originally concentrated on one identifier is split across multiple aliases, making the observed distribution closer to uniform.

Let the original vocabulary be $\mathcal{V} = \{1, 2, \ldots, V\}$, where the rank implies token frequency, and let the real token distribution be $p_i = \Pr[X = i]$. **AliasShare** assigns r_i aliases to token i, denoted by $A(i) = \{a_{i,1}, a_{i,2}, \ldots, a_{i,r_i}\}$. When the real token is i, the client samples one alias uniformly from $A(i)$, i.e., $\Pr[Z = a_{i,j} \mid X = i] = 1/r_i$. Therefore, the distribution observed by the adversary is

$$q_{i,j} = \Pr[Z = a_{i,j}] = p_i \cdot \frac{1}{r_i} = \frac{p_i}{r_i}. \tag{3}$$

Let the expanded vocabulary size be $\widetilde{V} = \sum_{i=1}^{V} r_i$. Thus, **AliasShare** transforms the original token probability p_i into the alias-level probabilities $q_{i,j} = p_i/r_i$. In other words, the probability mass of token i is no longer exposed through a single observable identifier, but is evenly distributed over r_i aliases.

To quantify the distributional leakage available to TFMA, we use the uniform distribution over the expanded vocabulary as the ideal security baseline. Let u denote the uniform distribution, where $u_a = 1/\widetilde{V}$. If $q = u$, then any secret permutation of the aliases leaves the observed distribution unchanged. Therefore, different token-alias mappings are indistinguishable from frequency statistics alone, and a frequency-matching adversary obtains no useful distributional signal.

Accordingly, we define the TFMA distributional leakage as the KL divergence between the observed alias distribution q and the uniform distribution u:

$$\mathcal{L}_{\mathrm{TFMA}}(q) = D_{\mathrm{KL}}(q\|u). \tag{4}$$

This choice is natural from an information-theoretic perspective. KL divergence (Cover & Thomas, 2006), also known as relative entropy, measures the information by which one distribution differs from a reference distribution. Since TFMA is fundamentally a distribution-matching attack, $D_{\mathrm{KL}}(q\|u)$ captures how much non-uniform structure remains in the adversary-visible distribution. Moreover, when the reference distribution is uniform, KL divergence is exactly the gap between the maximum possible entropy and the entropy of the observed distribution:

$$D_{\mathrm{KL}}(q\|u) = \sum_{a} q_a \log \frac{q_a}{1/\widetilde{V}} = \log \widetilde{V} - H(q). \tag{5}$$

Hence, $\mathcal{L}_{\mathrm{TFMA}}(q) = \log \widetilde{V} - H(q)$. This expression shows that minimizing TFMA leakage is equivalent to maximizing the entropy of the adversary-visible alias distribution. When $q = u$, the leakage is zero; the more q deviates from uniformity, the more frequency structure is available for TFMA. Since $q_{i,j} = p_i/r_i$, we have

$$H(q) = -\sum_{i=1}^{V}\sum_{j=1}^{r_i} \frac{p_i}{r_i} \log \frac{p_i}{r_i} = H(p) + \sum_{i=1}^{V} p_i \log r_i. \tag{6}$$

Substituting Eq. equation 6 into Eq. equation 4 gives

$$\mathcal{L}_{\mathrm{TFMA}}(q) = \log \widetilde{V} - H(p) - \sum_{i=1}^{V} p_i \log r_i. \tag{7}$$

When the expanded vocabulary size $\widetilde{V}$ is fixed, $\log \widetilde{V}$ is constant. Since the original token distribution p is also fixed, $H(p)$ is constant as well. Therefore, minimizing TFMA leakage is equivalent to maximizing $\sum_{i=1}^{V} p_i \log r_i$.

AliasShare thus formulates alias allocation as a constrained integer optimization problem in Eq. equation 1.

Since $p_i > 0$ and $\log r_i$ is strictly increasing, the optimal solution always uses the full expanded vocabulary budget, i.e., $\sum_i r_i = \widetilde{V}$.

We first consider the continuous relaxation with $r_i \geq 1$. The Lagrangian is

$$\mathcal{J} = \sum_{i=1}^{V} p_i \log r_i - \lambda \left(\sum_{i=1}^{V} r_i - \widetilde{V} \right) + \sum_{i=1}^{V} \mu_i (r_i - 1). \tag{8}$$

For tokens whose lower-bound constraint is inactive, i.e., $r_i > 1$, the KKT condition gives $\partial \mathcal{J}/\partial r_i = p_i/r_i - \lambda = 0$, and hence $r_i = p_i/\lambda$. Taking the lower bound into account, the continuous optimum has the water-filling form

$$r_i^\star = \max\left(1, \frac{p_i}{\lambda}\right), \tag{9}$$

where λ is chosen such that $\sum_i r_i^\star = \widetilde{V}$. This shows that frequent tokens should receive aliases approximately proportional to their probabilities, while sufficiently rare tokens remain at one alias.

However, the actual allocation must be integral. The integer problem is a separable concave resource allocation problem and can be solved exactly by marginal-gain allocation. Suppose token i currently has $r_i = k$ aliases. Increasing it to $k+1$ aliases improves the objective by

$$\Delta_i(k) = p_i\big[\log(k+1) - \log k\big] = p_i \log \frac{k+1}{k}. \tag{10}$$

For each token, $\Delta_i(k)$ is monotonically decreasing in k, because $\log((k+1)/k)$ decreases as k grows. Therefore, the exact integer optimum is obtained by initializing $r_i = 1$ for all tokens and repeatedly assigning one remaining alias slot to the token with the largest marginal gain:

$$i^\star = \arg\max_i p_i \log \frac{r_i + 1}{r_i}, \qquad r_{i^\star} \leftarrow r_{i^\star} + 1. \tag{11}$$

This process is repeated until $\sum_i r_i = \widetilde{V}$. Its optimality follows from the fact that the objective can be decomposed into discrete marginal gains:

$$p_i \log r_i = \sum_{k=1}^{r_i - 1} p_i \log \frac{k+1}{k}. \tag{12}$$

Thus, maximizing the total objective is equivalent to selecting the largest $\widetilde{V} - V$ marginal gains among all candidate increments. Since the marginal gains of each token are decreasing, selecting a later increment for a token necessarily implies that all earlier increments for the same token have no smaller gain and should also be selected. Hence, the *greedy marginal-gain procedure* in Algorithm 1 satisfies the ordering constraint and yields the exact integer optimum.

Assume tokens are ranked by decreasing frequency and follow Zipf's law $p_i = i^{-\alpha}/H_{V,\alpha}$, where $H_{V,\alpha} = \sum_{k=1}^{V} k^{-\alpha}$. Under the continuous relaxation, tokens satisfying $p_i > \lambda$ receive multiple aliases, while tokens with $p_i \le \lambda$ keep one alias. Therefore, there exists a cutoff rank N such that the top N frequent tokens are expanded. Around the cutoff, $p_N \approx \lambda$, which gives $\lambda \approx N^{-\alpha}/H_{V,\alpha}$. For $i \le N$, substituting the Zipf form into the continuous optimum gives

$$r_i^\star \approx \frac{i^{-\alpha}/H_{V,\alpha}}{N^{-\alpha}/H_{V,\alpha}} = \left(\frac{N}{i}\right)^{\alpha}. \tag{13}$$

Thus, the Zipf-aware allocation is not an independent heuristic, but a corollary of the constrained entropy-maximization problem.

However, Eq. equation 13 is derived from a continuous relaxation, while **AliasShare** uses the exact integer greedy allocation in Eq. equation 11. We show that the continuous solution serves as the asymptotic envelope of the greedy integer solution. Consider the standard Zipf case $\alpha = 1$, where $p_i = c/i$ for the normalization constant c. In the greedy algorithm, the k-th additional alias assigned to token i has marginal gain $\Delta_i(k) = p_i \log((k+1)/k)$. Since

$$\frac{1}{k+1} \le \log \frac{k+1}{k} \le \frac{1}{k}, \tag{14}$$

the marginal gains are sandwiched between two harmonic allocation rules. Consequently, for a fixed marginal threshold, the number of aliases selected for token i differs from the continuous value N/i only by a constant-level rounding error. Equivalently, the greedy integer allocation satisfies

$$r_i = \max\left(1, \left\lfloor \frac{N}{i} \right\rfloor\right) + O(1) \quad \text{for } i \le N, \tag{15}$$

and $r_i = 1$ for $i > N$. Therefore, the integer greedy solution and the continuous Zipf allocation induce the same first-order budget relation.

Let $B = \widetilde{V} - V$ denote the extra alias budget. Since only the top-N tokens receive extra aliases, the continuous Zipf envelope gives

$$B = \sum_{i=1}^{V} (r_i - 1) = \sum_{i=1}^{N} \left(\frac{N}{i} - 1\right) + O(N). \tag{16}$$

Using the harmonic-number approximation $H_N = \log N + \gamma + o(1)$, where γ is the Euler–Mascheroni constant, we obtain

$$B = NH_N - N + O(N) = N(\log N + \gamma - 1) + O(N). \tag{17}$$

Therefore, the exact integer allocation produced by the marginal-gain rule and the continuous Zipf relaxation induce the same first-order budget–cutoff relation. In particular,

$$N = \Theta\left(\frac{B}{\ln B}\right). \tag{18}$$

B.3 Resistance to SDA

Substitution Decipherment Attacks (SDA) exploit the sequential structure of the adversary-visible identifiers. The attacker treats the observed identifier sequence as a substitution cipher and searches for a decoded token sequence with high language-model likelihood. This is consistent with language-model-based decipherment, where the objective is to find a substitution function ϕ that maximizes the likelihood of the decoded plaintext sequence, e.g., $\hat{\phi} = \arg\max_\phi p(\phi(f_1)\cdots\phi(f_N))$. Prior decipherment attacks also use beam search and neural language models to score candidate plaintext sequences globally.

Without re-shuffling, a fixed secret mapping π induces a stable substitution channel:

$$Z_t = \pi(X_t). \tag{19}$$

Therefore, all observations can be concatenated into one long decipherment instance, allowing the adversary to accumulate frequency and sequential evidence.

AliasShare breaks this assumption by applying an independent offline re-shuffle in each round. Let π_t be the secret shuffle used in round t. Then the adversary observes

$$Z^{(t)} = \pi_t(X^{(t)}), \tag{20}$$

where π_t changes across refresh rounds. To link observations from two consecutive rounds, the adversary must recover the transition permutation

$$\Gamma_t = \pi_{t+1} \circ \pi_t^{-1}. \tag{21}$$

Assume each re-shuffle is sampled independently and uniformly over an anonymity set of size K. Then Γ_t is uniformly distributed over all A_V^K permutations:

$$\Pr[\Gamma_t = \gamma] = \frac{1}{A_V^K}. \tag{22}$$

Let $\widetilde{V}_t$ denote the adversary's public view across update rounds t and $t+1$, excluding the secret shuffle. Since the re-shuffled results are still secret-shared embedding vectors independent of the public view, we have

$$I(\Gamma_t; \widetilde{V}_t) = 0. \tag{23}$$

Thus, the public observation gives no information about the cross-round matching. Consequently, offline re-shuffling prevents the adversary from concatenating multiple rounds into a single long substitution cipher. SDA is reduced from one long-sequence decipherment problem to many short independent decipherment problems, plus a cross-round matching problem with A_V^K recovery complexity per refresh. Therefore, **AliasShare** mitigates SDA by destroying the stable substitution structure required for language-model-based decipherment.

C Extra memory estimation

We plot the solution map of equation 2 in Figure 3, from which B can be approximately solved. Consequently, **AliasShare** only expands about 54k items for GPT-2, and 153k, 183k, and 325k items for LLaMA-3, Qwen, and Gemma, respectively. For long-context scenarios, the peak extra memory usage introduced by **AliasShare** remains of the same order as previous methods relying on matrix multiplication as shown in Table 3.

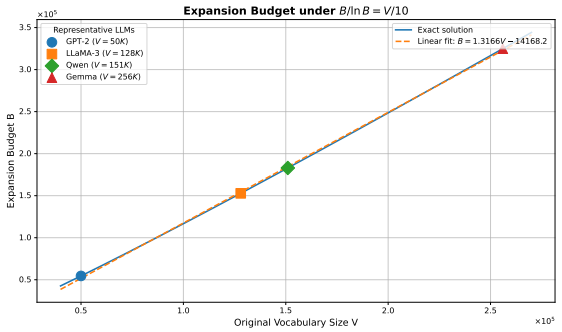

Figure 3: Expansion budget B as a function of the original vocabulary size V under the default setting $N = V/10$. The exact solution of $B/\ln B = V/10$ (blue) is closely approximated by a linear model (red dashed), indicating that the additional server-side embedding memory required for alias allocation scales approximately linearly with vocabulary size. Representative vocabularies from GPT-2, LLaMA-3, Qwen, and Gemma are shown for reference.

D Complete Related Work

D.1 Settings of MPC-LLMs

Existing MPC-based large language model (MPC-LLM) inference systems are primarily built upon either 2PC or 3PC frameworks. The common objective of both settings is to collaboratively execute Transformer inference without revealing user inputs, intermediate activations, or model parameters.

Most MPC-based LLM inference systems perform computations over a finite ring $\mathbb{Z}_q$, where $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$. All arithmetic operations are executed modulo q. Floating-point LLM parameters are usually transformed into fixed-point integer representations before secure inference: $\hat{x} = \lfloor x \cdot 2^f \rfloor$, where f denotes the scaling factor. The resulting integer values are then securely processed over $\mathbb{Z}_q$ throughout the MPC protocol.

Given secret-shared matrices $[A]$ and $[B]$, linear operations such as matrix addition can be evaluated locally: $[C] = [A] + [B]$. In contrast, secure multiplication requires interaction among participating servers. Given secret-shared values $[a]$ and $[b]$, many MPC systems employ Beaver Triples (Luo et al., 2024) to evaluate secure multiplication efficiently. A Beaver Triple consists of correlated random shares $([u], [v], [w])$ satisfying $w = u \cdot v$. The servers first compute masked differences: $d = a - u$ and $e = b - v$, followed by secure reconstruction of d and e. The multiplication result can then be obtained as $ab = w + d \cdot v + e \cdot u + d \cdot e$. This technique transforms expensive online multiplication into relatively lightweight linear operations plus limited interaction.

Based on these arithmetic primitives, most Transformer operations can be securely evaluated over secret shares. For example, matrix multiplication in attention and feed-forward layers can be represented as $[C] = [A] \cdot [B]$, while attention computation can be abstracted as $\mathrm{Softmax}(QK^\top)V$. Since non-linear functions such as Softmax and LayerNorm are difficult to evaluate exactly over secret shares, practical MPC-LLM systems typically employ polynomial approximation, lookup-table approximation, or mixed-protocol techniques to reduce online complexity.

D.2 Autoregressive Inference in MPC-LLMs

Modern LLMs predominantly adopt an autoregressive inference paradigm (Kwon et al., 2023; Sheng et al., 2023). Given an input prompt $x = (t_1, \ldots, t_n)$, the model sequentially generates tokens according to $p(t_{i+1} \mid t_{\le i}) = \mathrm{Softmax}(z_i)$, where $z_i = W_{\mathrm{lm}} h_i$ denotes the logits vector produced from hidden state h_i.

Autoregressive inference generally consists of two stages. During the prefill stage, the model processes the entire prompt in parallel: $t_1, \ldots, t_n \to h_1, \ldots, h_n$, while constructing KV caches $\mathrm{KV}_{1:n} = \{(K_j, V_j)\}_{j=1}^{n}$. During the decoding stage, the model only processes the newly generated token at each iteration: $t_i \to E(t_i) \to h_i \to z_i \to t_{i+1}$, where previously computed KV caches are reused.

In MPC-based LLM inference, all computations are performed over secret-shared values: $[t_i] \to [E(t_i)] \to [h_i] \to [z_i]$. For a 2PC protocol with additive sharing, $x = x^{(0)} + x^{(1)} \pmod q$ and

Table 7: Embedding table sizes for popular public datasets and models spanning across language and recommendation.

Application	# of Embedding Entries	Entry Size	Embedding Table Size
Criteo Rec.	$\sim$ 45,000,000	$\sim$ 128B	$\sim$ 5 GB
FastText Emb. (Language Model)	$\sim$ 2,000,000	$\sim$ 1024B	$>$ 1.9 GB
Taobao Rec.	$\sim$ 900,000	$\sim$ 128B	$\sim$ 109 MB
WikiText2 (Language Model)	$\sim$ 131,000	$\sim$ 512B	$\sim$ 64 MB
Criteo 1 TB Rec.	4,000,000,000	$\sim$ 128B	476 GB

$[x] = (x^{(0)}, x^{(1)})$. For a 3PC protocol with replicated secret sharing (RSS), $x = x^{(0)} + x^{(1)} + x^{(2)} \pmod q$, where $S_0 : (x^{(0)}, x^{(1)})$, $S_1 : (x^{(1)}, x^{(2)})$, and $S_2 : (x^{(2)}, x^{(0)})$.

At each decoding iteration, the servers collaboratively evaluate $[z_i] = \mathrm{MPCForward}([t_i], [\mathrm{KV}_{<i}])$. The logits shares are then returned to the client for reconstruction: $z_i = \mathrm{Recon}([z_i])$. The client locally performs token sampling: $t_{i+1} \sim \mathrm{Sample}(\mathrm{Softmax}(z_i))$, and secret-shares the sampled token again: $t_{i+1} \to [t_{i+1}]$. Therefore, autoregressive MPC inference forms an iterative protocol: $[t_i] \xrightarrow{\mathrm{MPC}} [z_i] \xrightarrow{\mathrm{Recon}} z_i \xrightarrow{\mathrm{Sample}} t_{i+1} \xrightarrow{\mathrm{Share}} [t_{i+1}]$.

Among all components, the embedding layer is particularly critical in MPC-LLMs. During autoregressive decoding, each generated token requires secure embedding processing: $[t_i], [E] \to [E[t_i]]$, where $E \in \mathbb{R}^{V \times d}$. Thus, secure embedding lookup becomes a repeated privacy-preserving index retrieval problem over a large vocabulary table.

Client-side communication in MPC-based autoregressive LLM inference is highly asymmetric. For an input prompt of length m and an output sequence of length n, the client downloads logits only during the n decoding steps, yielding $O(n)$ downstream communication. However, it must upload the m prompt tokens and possibly the n generated tokens, yielding $O(m + n)$ upstream communication. Since typical MPC workloads satisfy $m \gg n$, client upload dominates the communication cost, making uplink bandwidth the main bottleneck.

D.3 Other Applications of PIR

Besides end-to-end LLM inference, secure embedding lookup service is of great importance in ML applications, especially for large commercial recommendation systems. Recommendation models use several user and product input features to predict whether a user is likely to interact (e.g., click or purchase) with the product (Zhang et al., 2019; Zhou et al., 2018). These models may use user data such as the list of products the user recently purchased (Zhou et al., 2018). As the number of products can be on the order of millions, the corresponding embedding table can reach several GB to TB in size (Mudigere et al., 2022). Compressing the table is difficult for many real-world models, as it leads to significant accuracy drop (Chen et al., 2022). Recommendation models represent our primary target use case given their reliance on large server-side tables.

Table 7 summarizes the size of the embedding tables of some popular datasets/models. The size ranges from several MBs to hundreds of GBs. On the other hand, the size of the mobile application is on average 34MB and rarely exceeds 200MB even in extreme cases (SweetPricing, 2020). Embedding tables, especially for recommendation models, can easily exceed this range, which makes deploying them on-device impractical.

E Semi-Honest and Malicious Security Assumptions

MPC systems are typically analyzed under either the semi-honest or malicious adversarial model. The security assumption determines whether participating servers are assumed to follow the protocol faithfully during inference.

Under the semi-honest (honest-but-curious) assumption, each party correctly executes the protocol but may attempt to infer additional information from observed messages and local shares. Formally, for a protocol execution transcript τ_i observed by server S_i, the adversary is allowed to analyze τ_i together with its local shares $x^{(i)}$ to infer private information about user inputs or model parameters. However, the adversary does not arbitrarily deviate from the protocol execution procedure.

Most existing LLM systems adopt the semi-honest setting due to its relatively low communication and computation overhead.

In contrast, the malicious adversarial model allows at most one corrupted party to arbitrarily deviate from the protocol. A malicious server may: *modify shares*, *send inconsistent messages*, *return incorrect outputs*, or attempt to *bias the inference result intentionally*.

To defend against such behaviors, malicious-secure MPC protocols typically introduce additional verification mechanisms, including consistency checking, message authentication, share verification, and result confirmation.

Note that all 2PC/3PC protocols are based on non-collusion assumptions: security is not guaranteed if two or more servers collude and combine their local shares.

F Extending to Malicious Adversary Scenario

AliasShare is designed as a general embedding-layer acceleration primitive and can be seamlessly integrated into mainstream MPC-LLM frameworks.

For 2PC systems such as SHAFT and SIGMA, **AliasShare** directly replaces the original secure embedding lookup module without modifying downstream Transformer execution.

For 3PC systems such as MPCFormer, **AliasShare** naturally integrates with RSS workflows by additional cyclic communication. While in malicious-security settings such as Mosformer, the servers additionally perform consistency verification on locally retrieved embedding shares before entering the downstream Transformer pipeline. We compare the online operations between the different security models in Figure 4.

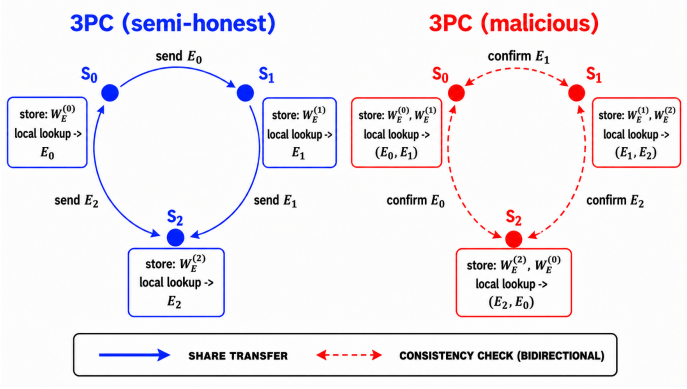

Figure 4: AliasShare integration with 3PC backends. Left: in the semi-honest setting, servers locally retrieve additive embedding components and perform one cyclic redistribution to obtain RSS-form inputs. Right: in the malicious setting, replicated embedding shares are predistributed, allowing each server to locally retrieve two shares and verify consistency with the others.

Since **AliasShare** only modifies the embedding stage while preserving the standard MPC computation graph afterward, it can also generalize to secure recommendation systems, cloud-hosted embedding retrieval, and mobile ML applications.