GRADIENT MIRAGE: TRAINABLE YET LABEL-UNIDENTIFIABLE GRADIENTS IN LARGE LANGUAGE MODEL SPLIT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient matching attacks (GMAs) in LLM split learning (SL) rely on a critical yet underexplored assumption: the gradient exposed at the split interface is a faithful derivative of the client's full-label training objective. This gradient-label correspondence allows a curious server to recover private labels by searching for a sequence whose induced gradient explains the observation. We propose Gradient Mirage, a defense that breaks this correspondence without discarding the optimization utility of the backward signal. Our key idea is to induce the adversary to solve a misspecified inverse problem, in which no plausible label sequence in the sequence space can explain the observed gradients. Concretely, Gradient Mirage achieves this by inducing inconsistency across three dimensions: objective, direction, and scale. Selective Autoregressive Supervision derives the exposed gradient from a masked surrogate loss rather than the full-label objective assumed by the attacker; Scale Blinding then applies randomized multiplicative rescaling, obscuring the gradient's natural magnitude; and Directional Privatization further randomizes the gradient direction while preserving its magnitude through the von Mises–Fisher (vMF) mechanism under a directional metric differential privacy guarantee. Crucially, utility is preserved: the Top segment still learns from all target tokens via Dual-Track Backpropagation, the exposed gradient remains informative since each supervised token retains its complete autoregressive context, and Bottom-Gradient Recovery restores the effective gradient for Bottom-segment optimization. Extensive experiments show that Gradient Mirage provides substantially stronger protection than existing defenses under comparable fine-tuning performance, achieving a better privacy-utility trade-off.

1 INTRODUCTION

Machine Learning-as-a-Service (MLaaS) Cai et al. (2024); Lin et al. (2026) has become a dominant paradigm for deploying large-scale models, allowing data owners to outsource computation and reduce infrastructure overhead. In its training setting, commonly known as Training-as-a-Service (TaaS), Split Learning (SL) Gupta & Raskar (2018); Vepakomma et al. (2018); Poirot et al. (2019); Chen et al. (2024); Shen et al. (2025); Zhang et al. (2026) mitigates direct data exposure by partitioning model training between the client and server. We focus on LLM supervised fine-tuning under *Label-Shielded Split Learning* Pasquini et al. (2021), where the compute-intensive Trunk segment is outsourced to the server, while the Bottom and Top segments remain on the client to keep private inputs, labels, and supervised loss local.

However, prior research Chen et al. (2024) shows that this LLM-SL system fails to provide effective privacy protection against a *curious-but-honest* server. In particular, the client's private data becomes highly vulnerable to GMAs Zhu et al. (2019); Zhao et al. (2020); Deng et al. (2021); Balunovic et al. (2022); Gupta et al. (2022) at the split interface between the Trunk and Top segment. As illustrated in Fig. 1, the adversary's guiding question is essentially: *"What label sequence would yield such a gradient?"* By optimizing for a label sequence whose interface gradients match the observed ones, it can recover sensitive training content even without explicit access to labels.

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.

We term this attack, launched at the split interface between the Trunk and Top segment, GMA-SL. GMA-SL constitutes a particularly compelling variant of classical GMA: We delineate its relationship to, and key distinctions from classical GMA in Section 2.2, and provide the concrete attack instantiation and implementation details in Section 2.3.

GMA-SL poses a particularly severe gradient leakage threat in LLM-SL. Unlike classical GMA, it matches activation gradients at the split interface, which retain explicit batch structure and are closely tied to local supervision. Moreover, because the cut-layer representation is already available, the adversary primarily optimizes the label sequence rather than jointly reconstructing inputs and labels, making the attack both simpler and harder to defend against.

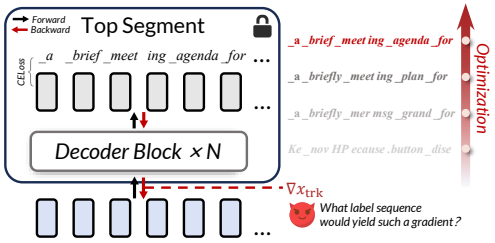


Figure 1: *Illustration of label leakage in label-shielded split learning.*

Motivated by this challenge, we propose Gradient Mirage, a defense method tailored to GMA-SL. Its key insight is that *what the model learns* and *what the exposed gradient reveals* need not coincide. Gradient Mirage deliberately decouples these two objectives. Under the constraint of minimally affecting optimization, we seek an exposed signal residing in a "gray zone" of the gradient space: to the server-side optimizer, it remains a serviceable backward signal for Trunk updates; to the adversary, it looks like a gradient, yet no suitable label sequence in the sequence space can explain it under the attacker's assumed objective. Consequently, an adversary performing gradient matching against this signal is in effect solving a misspecified inverse problem—one whose solution set is not anchored to the true labels.

Gradient Mirage realizes this design by inducing inconsistency across three dimensions. Dual-Track Backpropagation confines the full-label loss $\mathcal{L}_{\mathrm{F}}$ to local learning at the Top segment, while the transmitted gradient is generated by a masked surrogate loss $\mathcal{L}_{\mathrm{M}}$ via Selective Autoregressive Supervision, yielding *Objective Inconsistency* ($\mathcal{L}_{\mathrm{F}} \neq \mathcal{L}_{\mathrm{M}}$): making the attacker's assumed objective misspecified at its root. Directional Privatization then randomizes the gradient direction via the vMF mechanism Weggenmann & Kerschbaum (2021); von Mises (1918); Fisher (1953) with a directional metric differential privacy guarantee, inducing *Directional Inconsistency* ($\frac{g_{\text{release}}}{\|g_{\text{release}}\|} \neq \frac{\nabla_x \mathcal{L}_{\mathrm{M}}}{\|\nabla_x \mathcal{L}_{\mathrm{M}}\|}$). Scale Blinding applies randomized multiplicative rescaling, introducing *Scale Inconsistency* ($\|g_{\text{release}}\| \neq \|\nabla_x \mathcal{L}_{\mathrm{M}}\|$): even when the direction is approximately matched, the gradient's natural magnitude is no longer trustworthy. Meanwhile, optimization compatibility is supported by design: the Top segment still learns under full supervision, the vMF mechanism retains the gradient magnitude with a provable lower bound on the amplitude signal-to-noise ratio, and Bottom-Gradient Recovery restores the effective gradient for Bottom-segment optimization. Gradient Mirage thereby disrupts the label-identifiability of the exposed gradient while retaining stable and effective SL.

We present an effective defense against GMA-SL, a particularly severe gradient leakage threat in LLM-SL. As GMA-SL constitutes a specialized form of GMA in the SL setting, we adapt representative GMA defenses and a stringent sequence-level differential privacy (DP) mechanism as practical baselines, demonstrating that our method consistently achieves superior privacy-utility trade-offs.

The contributions are summarized as follows: (i) We identify gradient-label correspondence as the key vulnerability underlying GMA-SL and provide a systematic characterization of this attack in label-shielded LLM-SL, clarifying its attack surface and key distinctions from classical GMA. (ii) We propose Gradient Mirage, a defense that decouples what the model learns from what the gradient reveals, injecting objective, directional, and scale inconsistencies into the released gradient while preserving optimization utility. (iii) We conduct extensive experiments across multiple LLMs and datasets, adapting representative GMA defenses and a rigorous DP method as baselines. Gradient Mirage consistently achieves a superior privacy-utility trade-off.

2 PRELIMINARIES AND THREAT MODEL

2.1 SPLIT LEARNING

In SL setting, a neural network is partitioned between clients and a central server, with clients holding only a small model segment and most parameters residing on the server. SL typically takes two paradigms: (i) **Label-Exposed SL,** where the network is split into two parts and clients disclose labels to the server; (ii) **Label-Shielded SL,** where labels remain on-device and the network is split into three segments, with the bottom and top on clients and the trunk on the server.

In LLM supervised fine-tuning, next-token labels are obtained by shifting the input sequence by one position, so exposing labels would effectively reveal the private text. We therefore consider **Label-Shielded SL**. Given a client's private dataset D_{pri}, the overall network f_θ is partitioned into a *Bottom* f_{btm}, a *Trunk* f_{trk}, and a *Top* f_{top}. The server hosts f_{trk}, while the client retains f_{btm} and f_{top}. For a mini-batch $x \in D_{\text{pri}}$, the client transmits $x_{\text{btm}} = f_{\text{btm}}(x)$ to the server, and receives $x_{\text{trk}} = f_{\text{trk}}(x_{\text{btm}})$ from the server. The client then obtain the prediction $\hat{y} = f_{\text{top}}(x_{\text{trk}})$ and computes the training objective $\mathcal{L}(\hat{y}, y)$ with label y kept local. During backpropagation, the exchanged gradients $\partial\mathcal{L}/\partial x_{\text{trk}}$ allow the server to update the Trunk, while the client updates the Bottom and Top using $\partial\mathcal{L}/\partial x_{\text{btm}}$ and $\partial\mathcal{L}/\partial x_{\text{top}}$.

2.2 CLASSICAL GRADIENT MATCHING ATTACK

Classical GMA reconstructs private training data from shared gradients in distributed learning. Given a model snapshot $\boldsymbol{\theta}^{(t)}$, the observed gradient generated by a private minibatch $(x^{(t)}, y^{(t)})$ is $\mathbf{g}^{(t)} = \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(F\left(x^{(t)}; \boldsymbol{\theta}^{(t)}\right), y^{(t)}\right)$, where $\boldsymbol{\theta}^{(t)}$ denotes the parameter snapshot at step t, and $\left(x^{(t)}, y^{(t)}\right)$ the private minibatch that generated $\mathbf{g}^{(t)}$. The adversary introduces dummy variables (x_{dum}, y_{dum}) and solves:

$$(x^*, y^*) = \arg\min_{\tilde{x}, \tilde{y}} D\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(F(\tilde{x}; \boldsymbol{\theta}^{(t)}), \tilde{y}\right), \mathbf{g}^{(t)}\right), \tag{1}$$

where $D(\cdot, \cdot)$ is a gradient-discrepancy measure (e.g., $D(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2^2$, cosine distance, or a weighted combination). The optimization is carried out by gradient descent on the dummy variables:

$$(\tilde{x}, \tilde{y}) \leftarrow (\tilde{x}, \tilde{y}) - \eta \nabla_{(\tilde{x}, \tilde{y})} D\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(F(\tilde{x}; \boldsymbol{\theta}^{(t)}), \tilde{y}\right), \mathbf{g}^{(t)}\right), \tag{2}$$

while keeping $\boldsymbol{\theta}^{(t)}$ unchanged throughout the optimization.

2.3 GRADIENT MATCHING ATTACK IN SL

Unlike classical GMA, GMA-SL performs matching in the activation space at the split interface rather than directly on parameter gradients $\nabla_{\boldsymbol{\theta}}$, as illustrated in Fig. 2.

Let the Top segment be $F(\cdot; \theta_{\text{top}})$ and the top proxy held by the adversary $F(\cdot; \theta_{pro})$. Given label y, the client locally computes prediction $\hat{y} = F(x_{\text{trk}}; \theta_{\text{top}})$ and loss $\mathcal{L}(\hat{y}, y)$. The gradient propagated back to the cut layer is given by $\mathbf{g}_{\text{trk}} = \nabla_{x_{\text{trk}}} \mathcal{L}(F(x_{\text{trk}}; \boldsymbol{\theta}_{\text{top}}), y)$.

The attacker observes $\mathbf{g}_{\text{trk}}$ and seeks a plausible label sequence $\tilde{y}$ that reproduces it through the proxy model. The proxy-induced gradient is $\tilde{\mathbf{g}}_{\text{trk}}(\tilde{y}) = \nabla_{x_{\text{trk}}} \mathcal{L}(F(x_{\text{trk}}; \theta_{\text{pro}}), \tilde{y})$. The attack solves the following matching problem:

$$\tilde{y}^* = \arg\min_{\tilde{y}} D(\mathbf{g}_{\text{trk}}, \tilde{\mathbf{g}}_{\text{trk}}(\tilde{y})), \tag{3}$$

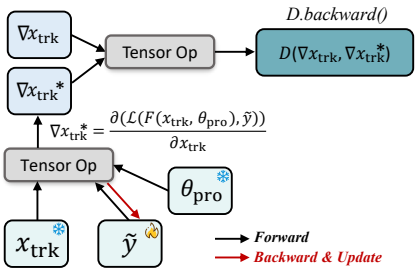


Figure 2: *Schematic computation graph of GMA-SL.*

where $D(\cdot, \cdot)$ is a discrepancy measure. The optimization is carried out by gradient descent on the dummy variables: $\tilde{y} \leftarrow \tilde{y} - \eta \nabla_{\tilde{y}} D(\mathbf{g}_{\text{trk}}, \tilde{\mathbf{g}}_{\text{trk}}(\tilde{y}))$, while θ_{pro} is held fixed during the inner-loop optimization.

Compared with classical GMA, GMA-SL yields the following advantages. (i) **Batch-separable matching.** The observed gradient $\nabla_{x_{\mathrm{trk}}}\mathcal{L} \in \mathbb{R}^{B\times L\times D}$ retains explicit batch structure, allowing samples to be matched independently except for the scalar coupling introduced by batch-averaged loss. The only coupling arises from the scalar coefficient induced by averaging the loss over the batch. (ii) **A simpler optimization objective.** With x_{trk} fixed, the attacker optimizes only the dummy label $\tilde{y}$, rather than jointly optimizing the input and label as in classical GMA. GMA-SL also introduces two challenges. (i) **Limited gradient observability.** The attacker can match only split-interface gradients (i.e., gradients with respect to activations), rather than full parameter gradients. (ii) **Semi-white-box scenario limitation.** The attacker lacks access to an exact snapshot of the victim's Top-segment parameters, making the attack non-white-box.

2.4 Threat Model

We consider a curious-but-honest adversary. Specifically, the server *faithfully* executes the split learning protocol (e.g., forward/backward computations and parameter updates) without deviating. Meanwhile, the server can log any legitimately available information and perform offline analysis or computation to reconstruct private sequences $x \in D_{\mathrm{pri}}$, i.e., a passive inference adversary.

Adversary's Capacities. We endow the curious server with the following capabilities: (i) Interface observability. The curious server can monitor all cut-layer activations (i.e., clients' smashed representations) x_{btm} and the corresponding back-propagated gradients ∇x_{trk} exchanged during the fine-tuning procedure. (ii) White-box access to the trunk. The adversary has full white-box knowledge of the server-side Trunk segment. (iii) Semi-white-box access to client-side segments, which will be explained below.

Semi-White-Box Access. Following prior work on LLM-SL Chen et al. (2024), we adopt a semi-white-box threat model. Concretely, the adversary does not have access to the victim's exact per-step fine-tuned parameters or optimizer state, but is aware of the model architecture and pre-trained checkpoint. This access level—situated between the white-box and black-box extremes—is both realistic and widely adopted in LLM-SL, since pre-trained weights are often public or available to the server as a model provider. Consequently, the attacker can mount near-white-box attacks but still faces a *fine-tuning gap* between the pre-trained proxy and the continually updated victim model.

3 Gradient Mirage

Gradient Mirage aims to induce three layers of inconsistency in GMA-SL while minimizing their impact on optimization (Fig. 3). *Objective Inconsistency* arises because the adversary assumes a full-label loss, whereas the exposed gradient is generated from a masked loss through Selective Autoregressive Supervision. *Directional Inconsistency* remains because Directional Privatization randomizes the exposed gradient direction using the von Mises–Fisher mechanism. *Scale Inconsistency* is introduced by Scale Blinding, making the gradient's natural magnitude unreliable even when its direction is approximately matched. To further improve optimization compatibility, Gradient Mirage employs Dual-Track Backpropagation to preserve full-label learning at the Top segment and Bottom-Gradient Recovery to restore the effective gradient used for Bottom-segment optimization.

3.1 Dual-Track Backpropagation

In SL for autoregressive language modeling, the Top segment has access to full target labels, whereas lower segments receive only split-interface gradient, which may be exploited by GMA. This asymmetry suggests that full supervision should be confined to local learning within the Top segment.

To this end, we introduce two backward tracks with distinct roles: a *Private Learning Track* and an *Exposed Optimization Track*. For an input sequence, let x_{trk} denote the hidden representation sent from the trunk to the Top segment. We define a standard full-label autoregressive loss $\mathcal{L}_F$ over all target tokens and a masked loss $\mathcal{L}_M$ over a selected subset of tokens. After a single forward pass produces logits Z_{pre}, the two tracks perform separate backward passes. In the Private Learning Track, $\mathcal{L}_F$ is used solely to optimize the Top-segment parameters, while its gradient with respect to x_{trk} is discarded. In the Exposed Optimization Track, $\mathcal{L}_M$ produces the interface gradient transmit-

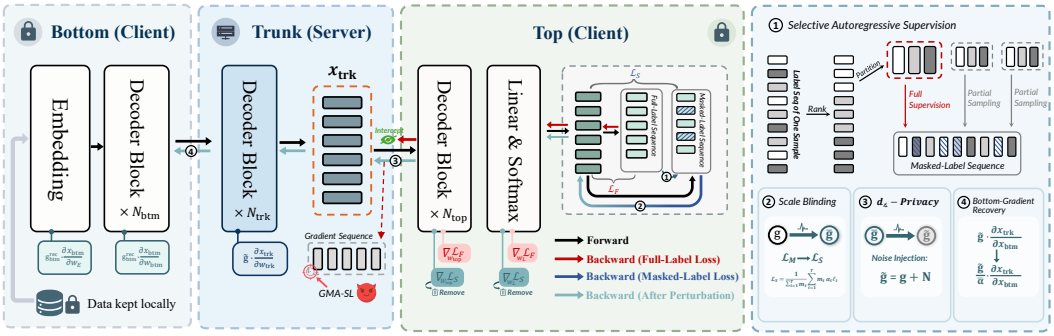


Figure 3: *Illustration of Gradient Mirage against GMA-SL.* The private track (red) remains local, while the exposed track (blue-to-cyan) is protected before transmission.

ted to the lower segments. This dual-track design preserves full-label learning at the Top segment while exposing only a controlled optimization signal across the split interface.

3.2 Selective Autoregressive Supervision

In LLM-SL, the exposed split-interface gradient depends critically on the token positions included in the loss. To this end, we propose **Selective Autoregressive Supervision (SAS)**, which supervises only a structured subset of token positions in each exposed backward pass. SAS preserves a useful optimization signal at the Top segment while inducing a mismatch between the true training gradient and the gradient assumed by an attacker.

For each sample, we construct the autoregressive target sequence by left-shifting the labels, yielding $\mathbf{y} = (y_1, y_2, \ldots, y_T)$, where T is the number of target tokens. In an initial forward pass, the Top segment produces token-wise predictive distributions $\{p_t\}_{t=1}^{T}$, from which we compute the predictive entropy $H_t = -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v)$, where $\mathcal{V}$ is the vocabulary. We then sort token positions by entropy and partition them into three strata: high-, medium-, and low-entropy. Each stratum is evenly divided into k disjoint subgroups, and one subgroup from each stratum is combined to form a token group. This yields k disjoint groups $G_1, G_2, \ldots, G_k$ satisfying $G_i \cap G_j = \emptyset \quad (i \neq j)$, $\bigcup_{i=1}^{k} G_i = \{1, 2, \ldots, T\}$. Hence, the k token groups are disjoint and jointly cover the entire label sequence.

At each training step, one token group is selected as the *full-supervision group* G_r, where all tokens receive supervision, while the remaining groups receive partial supervision. Let $m_t \in \{0, 1\}$ denote the supervision mask for position t. For G_r, we set $m_t = 1$ for all $t \in G_r$. For each remaining group G_j with $j \neq r$, we sample a subset $S_j \subseteq G_j$ with total token sampling ratio $\rho \in [0, 1]$, and set $m_t = 1$ for $t \in S_j$ and $m_t = 0$ otherwise. The masked autoregressive loss is $\mathcal{L}_M = \frac{1}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \ell_t$, $\ell_t = -\log p_t(y_t)$.

SAS serves two purposes. Fully supervising one group at each step preserves a sufficiently strong and coherent training signal, while entropy-aware partial supervision over the remaining groups makes the observed interface gradient deviate from that induced by standard full-label autoregressive loss. The hyperparameter k controls the granularity of the entropy-aware grouping and need not equal the number of fine-tuning epochs. In practice, the grouping can be periodically recomputed under the updated model state. In our experiments, we use $E = 3$ epochs and set $k = 3$.

3.3 Scale Blinding

Building on the masked loss $\mathcal{L}_M$ defined above, we introduce Scale Blinding to induce scale inconsistency by randomizing the magnitudes of token-level gradient contributions. Specifically, we replace the default unit weight of each supervised token with an independently sampled coefficient $\alpha_t > 0$, yielding the reweighted masked loss $\mathcal{L}_S = \frac{1}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \alpha_t \ell_t$, $\bar{\mathbf{g}} = \nabla_{x_{\text{trk}}} \mathcal{L}_S$. The released gradient therefore no longer reflects the natural token-wise scale induced by $\mathcal{L}_M$, hindering gradient matching and reconstruction. We parameterize the blinding distribution by its mean

scale and relative token-wise variation, and select $\mathrm{Unif}[1500, 2000]$ from a broad effective range; sensitivity analyses and scale-aware adaptive-attack results are provided in Appendix D. This randomized reweighting avoids deterministic scale patterns, incurs negligible computational overhead, and preserves supervision at all selected positions while disrupting magnitude-sensitive matching objectives such as ℓ_1, ℓ_2, and their weighted combination.

3.4 Directional Privatization

To introduce directional inconsistency, we adopt the vMF mechanism for directional perturbation. Rather than adding isotropic noise to the gradient, we perturb only its direction while preserving its magnitude, thereby obscuring direction while retaining optimization stability.

Given a nonzero interface gradient $\mathbf{g} \in \mathbb{R}^d$, let $\mu = \frac{\mathbf{g}}{\|\mathbf{g}\|_2} \in \mathbb{S}^{d-1}$. We then sample a perturbed direction $\tilde{\mu} \sim \mathrm{vMF}(\mu, \kappa)$, and release $\tilde{\mathbf{g}} = \|\mathbf{g}\|_2 \tilde{\mu}$, where the concentration parameter $\kappa \geq 0$ controls the perturbation strength. Larger κ concentrates $\tilde{\mu}$ around μ, whereas $\kappa = 0$ yields the uniform distribution on the hypersphere. The vMF mechanism satisfies Directional Metric Differential Privacy defined in Definition 1 under the angular distance $d_\angle(\mathbf{u}, \mathbf{v}) = \arccos(\mathbf{u}^\top \mathbf{v})$. Here, the concentration parameter κ determines the corresponding privacy budget ε.

Definition 1 (Directional Metric Differential Privacy). *Let $(\mathbb{S}^{d-1}, d_\angle)$ be the unit hypersphere equipped with angular metric $d_\angle$, and let $\varepsilon > 0$. A randomized mechanism $\mathcal{M} : \mathbb{S}^{d-1} \to \mathcal{P}(\mathbb{S}^{d-1})$ satisfies εd-metric differential privacy if, for any two input directions $\mathbf{u}, \mathbf{v} \in \mathbb{S}^{d-1}$ and any measurable output set $S \subseteq \mathbb{S}^{d-1}$, it holds that $\Pr[\mathcal{M}(\mathbf{u}) \in S] \leq \exp(\varepsilon d_\angle(\mathbf{u}, \mathbf{v})) \Pr[\mathcal{M}(\mathbf{v}) \in S]$.*

Compared with additive-noise DP mechanisms Dwork et al. (2006), the vMF mechanism avoids the need for a user-specified clipping threshold C, which is often difficult to estimate and tune in practice. The vMF mechanism also preserves the gradient magnitude, which uniformly bounds amplitude distortion. Specifically, for $\mathbf{n} = \tilde{\mathbf{g}} - \mathbf{g}$, we have $\|\mathbf{g}\|_2 / \|\mathbf{n}\|_2 \geq 1/2$ for every realization, with equality when $\tilde{\boldsymbol{\mu}} = -\boldsymbol{\mu}$. Hence, it provides directional privacy while preserving sufficient optimization signal for stable training.

Theorem 1 (Lower Bound on the Amplitude Signal-to-Noise Ratio). *Let $\mathbf{g} \in \mathbb{R}^d$ be a nonzero gradient and let $\mu = \frac{\mathbf{g}}{\|\mathbf{g}\|_2}$. Suppose that the vMF mechanism samples a perturbed direction $\tilde{\mu} \in \mathbb{S}^{d-1}$ and releases $\tilde{\mathbf{g}} = \|\mathbf{g}\|_2 \tilde{\mu}$.*

Define the perturbation noise as $\mathbf{n} = \tilde{\mathbf{g}} - \mathbf{g}$, and amplitude signal-to-noise ratio as $\mathrm{ASNR} = \frac{\|\mathbf{g}\|_2}{\|\mathbf{n}\|_2}$. For any realization of $\tilde{\mu}$, the amplitude signal-to-noise ratio satisfies $\mathrm{ASNR} \geq \frac{1}{2}$, where the case $\tilde{\mu} = \mu$ is interpreted as $\mathrm{ASNR} = +\infty$. The lower bound is tight and is attained when $\tilde{\mu} = -\mu$.

3.5 Bottom-Gradient Recovery

Scale Blinding deliberately amplifies token-level gradient contributions, which may distort the gradient scale used for Bottom-segment optimization. We therefore normalize the gradient reaching the Bottom segment by the expected blinding factor. Let $\bar{\alpha} = \mathbb{E}[\alpha_t]$, and let $\tilde{\mathbf{g}}$ denote the protected interface gradient. For $x_{\mathrm{trk}} = f(x_{\mathrm{btm}})$, the gradient propagated through the Trunk is $\tilde{\mathbf{g}}_{\mathrm{btm}} = J_f(x_{\mathrm{btm}})^\top \tilde{\mathbf{g}}$. Before updating the Bottom segment, we recover its effective scale as $\mathbf{g}_{\mathrm{btm}}^{\mathrm{rec}} = \frac{1}{\bar{\alpha}} \tilde{\mathbf{g}}_{\mathrm{btm}}$.

Since $\bar{\alpha}$ is the expected token weight, this normalization removes the average scale amplification introduced by Scale Blinding while retaining its token-wise randomness.

4 Experiments

Experimental Configuration.[1] We evaluate GMA-SL and its defenses on three LLMs: Llama-2-7B Touvron et al. (2023), Llama-3-8B Grattafiori et al. (2024), and DeepSeek-LLM-7B Bi et al. (2024), using CodeAlpaca Chaudhary (2023), GSM8K Cobbe et al. (2021), and PIQA Bisk et al.

[1]Due to space limitations, we present the representative results and analyses in this section. Additional results and analyses are provided in Appendix D.

Dataset		CodeAlpaca			PIQA			GSM8K		
Model	Method	RougeL-F	Meteor	PPL	RougeL-F	Meteor	PPL	RougeL-F	Meteor	PPL
Llama3 -8B	GP	1.00±.00	.99±.00	5.29±.54	.96±.01	.94±.02	11.50±2.16	1.00±.00	1.00±.01	12.23±1.13
	GD	.94±.01	.94±.01	4.78±.10	.95±.01	.91±.00	7.76±.03	.95±.01	.94±.01	12.48±.97
	LDP(10240)	.56±.03	.57±.02	4.67±.00	.80±.03	.76±.05	7.74±.06	.90±.01	.92±.01	11.04±.03
	LDP(12288)	.66±.03	.66±.04	4.61±.04	.81±.03	.77±.05	7.20±.02	.89±.02	.89±.02	11.12±.04
	Top-Only	-	-	4.83±.01	-	-	8.87±.02	-	-	12.00±.01
	Standard	1.00±.00	.99±.00	4.44±.00	.99±.00	.97±.01	7.40±.01	1.00±.00	1.00±.00	10.59±.00
	Ours(512)	**.20**±.00	**.23**±.00	4.45±.00	**.21**±.02	**.14**±.01	7.40±.02	**.11**±.01	**.07**±.01	10.64±.03
	Ours(1024)	**.35**±.02	**.34**±.02	**4.42**±.01	**.45**±.02	**.31**±.03	**7.19**±.03	**.36**±.01	**.21**±.01	**10.50**±.01
Llama2 -7B	GP	.99±.00	.99±.01	5.00±.09	1.00±.00	1.00±.01	6.50±.62	.98±.01	.98±.00	10.66±.23
	GD	.90±.02	.89±.02	6.98±1.58	.81±.02	.80±.02	7.76±1.77	.90±.01	.92±.00	107.08±163.00
	LDP(10240)	.78±.03	.74±.04	4.80±.08	.79±.04	.74±.04	6.30±.66	.38±.02	.33±.01	10.49±.15
	LDP(12288)	.82±.04	.80±.04	4.85±.13	.83±.02	.79±.02	5.85±.05	.44±.03	.40±.03	10.48±.14
	Top-Only	-	-	4.96±.04	-	-	6.27±.06	-	-	11.20±.28
	Standard	.99±.00	.98±.01	**4.73**±.02	.99±.01	.99±.01	**5.29**±.10	.98±.00	.98±.00	**9.71**±.14
	Ours(512)	**.10**±.01	**.11**±.01	4.82±.10	**.05**±.02	**.05**±.00	5.63±.13	**.03**±.01	**.04**±.01	10.43±.16
	Ours(1024)	**.17**±.02	**.15**±.01	4.77±.05	**.10**±.01	**.09**±.01	5.56±.23	**.08**±.01	**.07**±.01	10.14±.17
DeepSeek -LLM-7B	GP	.99±.00	.97±.00	5.39±.05	.94±.02	.93±.02	14.39±2.09	.79±.01	.75±.03	39.11±35.00
	GD	.68±.04	.68±.04	6.33±.33	.84±.01	.82±.03	301.78±415.97	.84±.02	.85±.03	16.10±3.46
	LDP(8192)	.60±.02	.54±.03	5.05±.05	.82±.02	.76±.02	6.71±.24	.31±.02	.21±.03	12.13±.12
	LDP(10240)	.74±.02	.71±.03	5.00±.10	.85±.01	.71±.02	6.72±.27	.34±.01	.24±.03	12.13±.12
	Top-Only	-	-	5.44±.12	-	-	7.44±.13	-	-	13.72±.11
	Standard	1.00±.00	.98±.00	**4.38**±.00	1.00±.00	.97±.01	**5.79**±.01	.98±.00	.98±.00	**10.01**±.00
	Ours(512)	**.07**±.02	**.05**±.02	5.30±.22	**.07**±.01	**.04**±.00	6.67±.16	**.03**±.00	**.01**±.00	11.99±.11
	Ours(1024)	**.11**±.01	**.06**±.01	5.15±.14	**.14**±.01	**.08**±.01	6.45±.02	**.07**±.01	**.03**±.00	12.04±.29

Table 1: *Comparison between Gradient Mirage and existing privacy-preserving methods.*

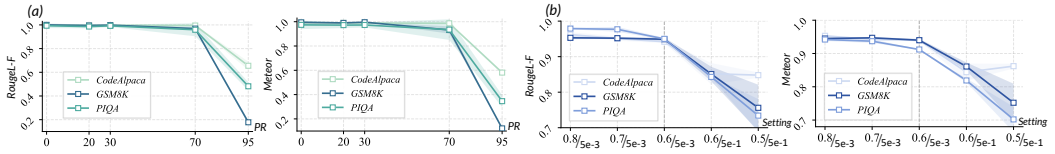


Figure 4: *Effect of different defenses against GMA-SL: (a) GP under different PRs; (b) GD under different drop rates and Gaussian noise standard deviations.*

(2020). All models and attack tensors use bfloat16. We assign six decoder layers to the Bottom segment and five to the Top segment, with the remaining layers hosted in the Trunk.

Implementation Details. We evaluate the data reconstruction performance of GMA-SL using the following metrics: (i) ROUGE-based F1 scores Lin (2004), primarily ROUGE-L F1; (ii) Meteor Banerjee & Lavie (2005), as a complementary semantically aware metric. We use mean perplexity (PPL) to evaluate the fine-tuning performance of the LLMs.

Attack Setting. We perform attacks periodically during fine-tuning. Specifically, over a total of 600 fine-tuning steps, we launch one GMA-SL attack every 50 steps and evaluate the corresponding reconstruction metrics. We adopt BiSR(b) Chen et al. (2024), a state-of-the-art GMA-SL paradigm, with the optimization objective: $L(\tilde{y}) = \beta \left\|\nabla_{x_{\text{trk}}}\mathcal{L}(x_{\text{trk}}, \tilde{y}) - \nabla_{x_{\text{trk}}}\mathcal{L}(x_{\text{trk}}, y)\right\|_2 + (1-\beta)\left\|\nabla_{x_{\text{trk}}}\mathcal{L}(x_{\text{trk}}, \tilde{y}) - \nabla_{x_{\text{trk}}}\mathcal{L}(x_{\text{trk}}, y)\right\|_1$. In our experiments, β is set to 0.65. In the ablation study, we also use cosine distance as the matching objective.

4.1 Existing Privacy-Preserving Defenses

Gradient Pruning. An intuitive defense against GMA-SL is Gradient Pruning (GP) Vadera & Ameen (2022), which forces the adversary to match against an incomplete gradient. Fig. 4(a) presents the effect of GP against GMA-SL under different pruning rates (PRs), evaluated on Llama-3-8B. It can be seen that GP does not effectively defend against GMA-SL, and training becomes unstable when PR exceeds 90%.

Gradient Dropout. Gradient Dropout (GD) Sotthiwat et al. (2026) transforms the shared gradient by randomly retaining a subset of coordinates and perturbing the rest. Each gradient component is

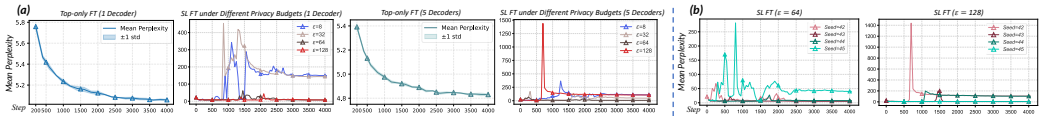


Figure 5: *(a) Comparison between Top-Only FT and SL FT under GradSeq-LDP. (b) Seed sensitivity of SL FT across random seeds. Experiments were conducted on Llama3-8B using CodeAlpaca.*

kept with probability p and rescaled by $1/p$; otherwise, it is replaced with Gaussian noise sampled from $\mathcal{N}(0, \sigma^2)$. The key idea is to preserve partial training signal while disrupting coordinate-wise gradient information, thereby hindering GMAs. Fig. 4(b) presents the effect of GD against GMA-SL under different drop rates and Gaussian noise standard deviations, evaluated on Llama-3-8B.

Gradient Sequence Local DP. Unlike DP-SGD Abadi et al. (2016), the object to be released and protected in our setting is the gradient sequence $G \in \mathbb{R}^{L\times D}$. Inspired by DP-Forward Du et al. (2023), which enforces privacy in the forward pass by clipping and perturbing client-side intermediate activations before transmission, we extend *Sequence Local DP (SeqLDP)* Du et al. (2023) to *Gradient Sequence Local DP (GradSeq-LDP)*.

Definition 2 (**GradSeq-LDP**). *For $\varepsilon \geq 0$, $0 \leq \delta \leq 1$, $\mathcal{M}$ satisfies (ε, δ)-GradSeq-LDP, if $\forall G, G' \in \mathbb{R}^{L\times D}$, and any possible output subset $\mathcal{O}$, it holds that* $\Pr[\mathcal{M}(G) \in \mathcal{O}] \leq e^{\varepsilon} \Pr[\mathcal{M}(G') \in \mathcal{O}] + \delta$.

We implement *GradSeq-LDP* using the analytic Gaussian mechanism (aGM) Balle & Wang (2018). *GradSeq-LDP* enforces privacy protection at an earlier stage than DP-SGD. Consequently, applying such a strong privacy mechanism at this location may substantially affect training. As shown in Tab. 3, a small ε indeed provides strong defense effectiveness, but leads to highly unstable training. Fig. 5(a) shows that fine-tuning under small privacy budgets is highly unstable and underperforms Top-Only Training. Even when a particular run appears stable, the training can become unstable after changing the random seed, as illustrated in Fig. 5(b).

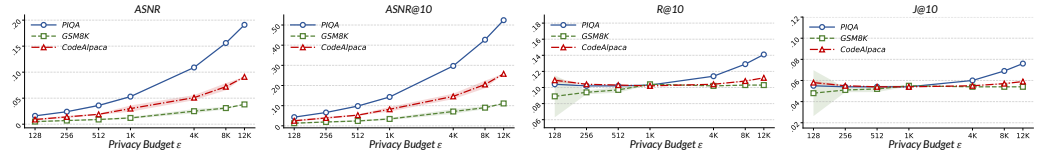


Figure 6: *Gradient utility of GradSeq-LDP under different privacy budgets across three datasets.*

As shown in Fig. 6, across all four gradient utility metrics, GradSeq-LDP preserves only limited signal strength and salient-coordinate information even at large ε. Definitions of the four metrics are provided in Appendix C.

4.2 Benchmarking Gradient Mirage

To fairly compare defense effectiveness against BiSR(b), we tune all privacy-preserving methods to achieve comparable fine-tuning performance. Results are reported in Tab. 1. For GP, we use PRs of 0.7 for the Llama-series models and 0.8 for DeepSeek to obtain fine-tuning performance of Gradient Mirage. For GD, we set the retention probability to $p = 0.6$ and Gaussian noise standard deviation to 5×10^{-3}. For *GradSeq-LDP*, the clipping coefficients C are set to 0.12, 0.02, and 0.01 for Llama-3, Llama-2, and DeepSeek, respectively, while the corresponding privacy budgets ε are specified in the tables. For Gradient Mirage, we set the SAS sampling ratio to $\rho = 0.6$ and use the directional metric privacy budget $\varepsilon \in \{512, 1024\}$ induced by the vMF mechanism for all models. All experiments are conducted with a batch size of 8 and repeated using three random seeds.

Dataset	Gradient Mirage(PPL)	Dual-Track BP (✗)		Bottom-Gradient Rec (✗)		Trunk Training (✗)	
		PPL	Δ(%)	PPL	Δ(%)	PPL	Δ(%)
CodeAlpaca	4.42±.01	4.51±.05	2.04↑	4.51±.01	2.04↑	4.48±.05	1.36↑
PIQA	7.19±.03	7.30±.03	1.53↑	7.43±.02	3.34↑	7.23±.02	0.56↑
GSM8K	10.50±.01	10.61±.05	1.05↑	10.69±.03	1.82↑	10.59±.03	0.86↑

Table 2: *Impact of Dual-Track Backpropagation, Bottom-Gradient Recovery, and Trunk Training on SL utility.*

ε	RougeL-F	Rouge2-F	Meteor
128	.02±.01	.00±.00	.01±.00
256	.03±.01	.00±.00	.02±.01
1024	.12±.05	.05±.03	.10±.05
8192	.62±.07	.48±.09	.62±.08
10240	.56±.03	.41±.02	.57±.02
12288	.66±.03	.52±.04	.66±.04

Table 3: *Defense performance of GradSeq-LDP.*

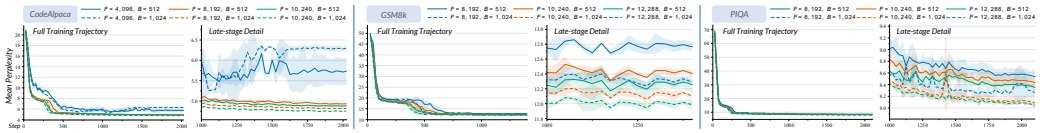


Figure 7: *Fine-tuning performance under joint forward and backward protection. F and B denote the forward- and backward-privacy budgets, respectively.*

4.3 Ablation Study

Impact of SAS and Directional Privatization. We use cosine distance as the matching objective to isolate SAS and Directional Privatization. As shown in Tab. 4, both SAS and Directional Privatization substantially reduce reconstruction performance by perturbing the positional and directional information of the exposed gradient, respectively. **Impact of Scale Blinding.** We evaluate amplitude perturbation against BiSR(b), which adopts the magnitude-sensitive objective used by TAG. As shown in Tab. 5, removing Scale Blinding increases ROUGE-L F1 from 0.35 to 0.71, indicating that SAS and Directional Privatization are insufficient.

Table 4: *Impact of SAS and Directional Privatization.*

Setting	RougeL-F	Rouge2-F	Meteor
Scale Blinding-Only	.55±.11	.25±.04	.43±.11
ρ=0.8	.42±.09	.11±.04	.28±.09
ρ=0.7	.41±.06	.11±.03	.28±.04
ρ=0.6	.37±.04	.09±.02	.25±.03
ρ=0.6,ε=8192	.29±.02	.05±.01	.19±.02
ρ=0.6,ε=1024	.16±.01	.01±.00	.11±.00
ρ=0.6,ε=512	.09±.01	.00±.00	.07±.00

Impact of Dual-Track Backpropagation. As shown in Tab. 2, removing Dual-Track Backpropagation consistently increases PPL across all datasets. This confirms that retaining full-label supervision in the *private learning track* is important for preserving fine-tuning utility.

Table 5: *Impact of Scale Blinding.*

Setting	RougeL-F	Rouge2-F	Meteor
w/o Scale Blinding	.71±.01	.51±.02	.69±.01
w/ Scale Blinding	.35±.02	.18±.02	.34±.02

Impact of Bottom-Gradient Recovery. Disabling Bottom-Gradient Recovery likewise degrades PPL across all datasets, as the blinded gradient scale directly affects Bottom-segment optimization. These results validate its role in preventing Scale Blinding from impairing training. **Impact of Trunk Training.** The gradients in the Trunk segment are subject to a certain degree of perturbation. Therefore, whether these gradients should be used to update the model is itself an important factor that warrants ablation. As shown in Tab. 2, we find that updating the trunk parameters does not compromise training stability; instead, it slightly improves training performance.

4.4 Joint Forward and Backward Protection

We combine Gradient Mirage with forward-pass protection method to protect both communication directions. As shown in Fig. 14, the joint defense provides effective protection against both forward- and backward-channel leakage. Meanwhile, Fig. 7 shows stable fine-tuning across different forward and backward privacy budgets on all three datasets, demonstrating that Gradient Mirage can be readily integrated with existing forward defenses while retaining good training utility. We adopt BiSR(f) as the forward attack, with detailed configurations provided in Appendix D.5. Additional evaluation metrics under this setting are reported in Fig. 15.

5 Conclusion

We identify gradient–label correspondence as the key vulnerability enabling GMA-SL in LLM-SL and propose Gradient Mirage, a defense that turns gradient matching into a misspecified inverse problem. By introducing objective, directional, and scale inconsistencies while preserving full-label learning, Gradient Mirage decouples model learning from gradient leakage. Experiments across multiple LLMs and datasets show substantially reduced label reconstruction with comparable fine-tuning performance, achieving a favorable privacy–utility trade-off. Extending gradient privacy defenses beyond autoregressive LLMs and split learning to other training paradigms and model architectures remains an important direction for future work.

AI USE STATEMENT

We used generative AI tools for literature exploration and language polishing. All technical ideas, method design, theoretical analysis, experimental implementation, result interpretation, and manuscript decisions were carried out and verified by the authors. The authors take full responsibility for the content of the paper.

ETHICS STATEMENT

This work studies privacy leakage and defense in LLM split learning. All experiments are conducted on publicly available models and datasets and involve no human subjects or personally identifiable information. Although gradient reconstruction techniques may have dual-use implications, they are investigated solely to characterize privacy risks and develop effective defenses. Gradient Mirage is intended to improve the privacy of distributed LLM fine-tuning against curious-but-honest servers. Any practical deployment should follow applicable privacy, data-protection, and security requirements.

REPRODUCIBILITY STATEMENT

The paper specifies the threat model, model partitioning, attack formulation, and all components of Gradient Mirage, together with their mathematical definitions and training procedures. The experimental section reports the evaluated models and datasets, attack settings, defense hyperparameters, privacy budgets, batch size, and random-seed configuration. We further provide ablation studies and additional analyses in the appendix, including detailed evaluation metrics, sensitivity analyses, and joint forward- and backward-protection results.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International conference on machine learning*, pp. 394–403. PMLR, 2018.

Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. Lamp: Extracting text from gradients with language model priors. *Advances in Neural Information Processing Systems*, 35:7641–7654, 2022.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72, 2005.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

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

Zinuo Cai, Rongbo Ma, Yicheng Fu, Weishan Zhang, Ruhui Ma, and Haibing Guan. Llmaas: Serving large-language models on trusted serverless computing platforms. *IEEE Transactions on Artificial Intelligence*, 6(2):405–415, 2024.

Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation, 2023.

Guanzhong Chen, Zhenghan Qin, Mingxin Yang, Yajie Zhou, Tao Fan, Tianyu Du, and Zenglin Xu. Unveiling the vulnerability of private fine-tuning in split-based frameworks for large language models: A bidirectionally enhanced attack. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 2904–2918, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, eighth workshop on syntax, semantics and structure in statistical translation*, pp. 103–111, 2014.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

Jieren Deng, Yijue Wang, Ji Li, Chenghong Wang, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. Tag: Gradient attack on transformer-based language models. In *Findings of the association for computational linguistics: EMNLP 2021*, pp. 3600–3610, 2021.

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*, pp. 2665–2679, 2023.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.

Ronald Aylmer Fisher. Dispersion on a sphere. *Proceedings of the royal society of London. Series A. Mathematical and physical sciences*, 217(1130):295–305, 1953.

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

Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. Recovering private text in federated learning of language models. *Advances in neural information processing systems*, 35:8130–8143, 2022.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Yu Lin, Qizhi Zhang, Wenqiang Ruan, Daode Zhang, Jue Hong, Ye Wu, Hanning Xia, Yunlong Mao, and Sheng Zhong. Towards privacy-preserving llm inference via covariant obfuscation (technical report). *arXiv preprint arXiv:2603.01499*, 2026.

Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pp. 2113–2129, 2021.

Maarten G Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar. Split learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115*, 2019.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, 2017.

Xicong Shen, Yang Liu, Yi Liu, Peiran Wang, Huiqi Liu, Jue Hong, Bing Duan, Zirui Huang, Yunlong Mao, Ye Wu, et al. Sap: Privacy-preserving fine-tuning on language models with split-and-privatize framework. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 502–510, 2025.

Ekanut Sotthiwat, Chi Zhang, Xiaokui Xiao, and Liangli Zhen. Safeguarding federated learning from data reconstruction attacks via gradient dropout. *IEEE Transactions on Information Forensics and Security*, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Sunil Vadera and Salem Ameen. Methods for pruning deep neural networks. *Ieee Access*, 10: 63280–63300, 2022.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Richard von Mises. Über die "ganzzahligkeit" der atomgewichte und verwandete fragen. *Physikalische Zeitschrift*, 19:490, 1918.

Benjamin Weggenmann and Florian Kerschbaum. Differential privacy for directional data. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1205–1222, 2021.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv e-prints*, pp. arXiv–2412, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qinbo Zhang, Yanhang Shi, Ziyi Zhang, Hao Wang, Sai Qian Zhang, and Jian Li. Stealing split learning bottom models by recovering embedding geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20660–20669, June 2026.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

SUPPLEMENTARY MATERIAL

The supplementary material provides additional details about our work. Specifically, we provide:

- A. Explanation of the batch-separable matching property of GMA-SL.
- B. More details of the vMF Mechanism.
- C. More details of Selective Autoregressive Supervision.
- D. Additional experimental details and analyses.
- E. Visualization of the effectiveness of Gradient Mirage.
- F. Limitation and Future Work.

A BATCH-SEPARABLE MATCHING OF GMA-SL

As discussed in Section 2.3, GMA-SL benefits from the explicit batch structure preserved in the exposed gradient, enabling sample-wise matching within a batch. The only coupling between samples comes from the scalar coefficient introduced by batch-averaged loss. Here, we provide a detailed analysis of this batch-separable matching property. An illustration of GMA-SL is shown in Fig. 8.

A.1 LOSS IN AUTO-REGRESSIVE DECODER STRUCTURE

We first formalize the autoregressive training loss used throughout the paper and clarify how the loss is computed by a benign client and the adversary under our threat model.

For a mini-batch of size B and sequence length L, let $\mathbf{H} \in \mathbb{R}^{B\times L\times D}$ denote the input feature tensor (token-wise hidden states) at the input of the language-modeling head, and let $\mathbf{M} \in \{0,1\}^{B\times L}$ be the attention mask, where $\mathbf{M}_{b,\ell} = 1$ indicates that position ℓ in example b is a valid token contributing to training, and $\mathbf{M}_{b,\ell} = 0$ indicates padding or an ignored position. The model produces next-token logits

$$\mathbf{L} = f_{\mathrm{LM}}(\mathbf{H};\boldsymbol{\theta}) \in \mathbb{R}^{B\times L\times V}, \tag{4}$$

where V is the vocabulary size and $\boldsymbol{\theta}$ are model parameters.

Let $\mathbf{Y} \in \{1,\dots,V\}^{B\times L}$ be the token id matrix for the same sequences. With the standard one-step shift, the training target for position ℓ is $\mathbf{Y}_{b,\ell+1}$. Define the per-token cross-entropy

$$\ell_{b,\ell}(\boldsymbol{\theta}) = -\log \mathrm{Softmax}(\mathbf{L}_{b,\ell,:})_{\mathbf{Y}_{b,\ell+1}}. \tag{5}$$

We mask out invalid positions and (optionally) the final position, yielding the masked autoregressive loss

$$\mathcal{L}_{\mathrm{AR}}(\boldsymbol{\theta}) = \frac{1}{\sum_{b=1}^{B}\sum_{\ell=1}^{L-1}\mathbf{M}_{b,\ell+1}} \sum_{b=1}^{B}\sum_{\ell=1}^{L-1}\mathbf{M}_{b,\ell+1}\,\ell_{b,\ell}(\boldsymbol{\theta}). \tag{6}$$

For the clients, they perform the computation in Eq. 6 locally at the end of the Top segment. For the adversary, they execute:

$$\begin{aligned} \tilde{\mathbf{L}} &= F(x_{\mathrm{trk}};\theta_{\mathrm{pro}}) \in \mathbb{R}^{B\times L\times V}, \\ \tilde{\ell_{b,\ell}}(\boldsymbol{\theta}) &= -\log \mathrm{Softmax}\Big(\tilde{\mathbf{L}}_{b,\ell,:}\Big)_{\tilde{\mathbf{Y}}_{b,\ell+1}}, \\ \mathcal{L}_{\mathrm{AR}}(\boldsymbol{\theta}) &= \frac{1}{\sum_{b=1}^{B}\sum_{\ell=1}^{L-1}\mathbf{M}_{b,\ell+1}} \sum_{b=1}^{B}\sum_{\ell=1}^{L-1}\mathbf{M}_{b,\ell+1}\,\tilde{\ell_{b,\ell}}(\boldsymbol{\theta}). \end{aligned} \tag{7}$$

The adversary fully knows the normalization factor $N = \sum_{b=1}^{B}\sum_{\ell=1}^{L-1}\mathbf{M}_{b,\ell+1}$. This is because, in addition to the cut-layer representation x_{btm}, the attention mask $\mathbf{M}$ must also be transmitted to the server: the server-side trunk requires $\mathbf{M}$ to construct attention score matrices and to carry out the forward pass. Since this information is a functional prerequisite for executing the protocol, even the weakest adversarial assumptions in the LLM-SL setting typically grant the server access to $\mathbf{M}$, and hence to N.

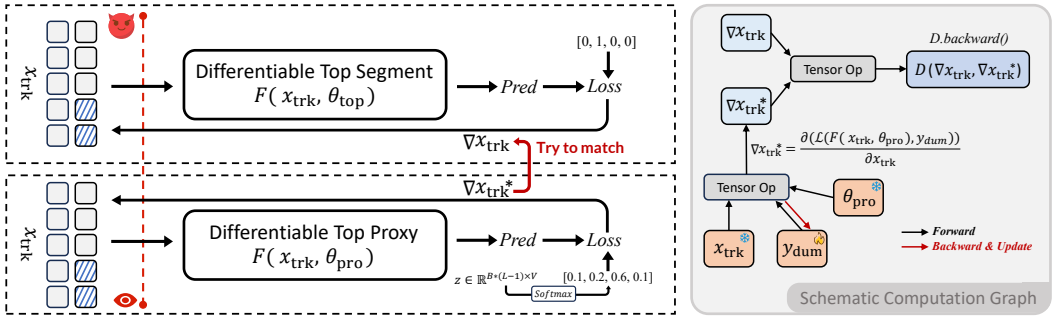


Figure 8: *Gradient matching attack in split learning (GMA-SL).* The adversary observes the gradient with respect to the cut-layer representation ∇x_{trk}, produced by the victim's Top segment, then optimizes a dummy label y_{dum} using a differentiable proxy top model $F(\cdot; \boldsymbol{\theta}_{\text{pro}})$, such that the proxy-induced gradient $\nabla_{x_{\text{trk}}} \mathcal{L}\big(F(x_{\text{trk}}; \boldsymbol{\theta}_{\text{pro}}), \tilde{y}\big)$ matches the observed one.

A.2 Batch-Separable Matching

We elaborate on advantage (i) of the adversary in GMA-SL, as introduced in Section 2.3, namely why samples within the same mini-batch do not interfere with one another during gradient matching. We begin with the computational details of gradient matching in SL.

Let B be the batch size, L the sequence length, and V the vocabulary size. The server observes (i) the smashed data $x_{\text{trk}} \in \mathbb{R}^{B\times L\times D}$, (ii) the attention mask $\mathbf{M} \in \{0,1\}^{B\times L}$, and (iii) the back-propagated gradient w.r.t. the smashed data $\mathbf{G} = \mathbf{g}_{\text{trk}} \in \mathbb{R}^{B\times L\times D}$, where $\mathbf{g}_{\text{trk}} = \nabla_{x_{\mathbf{trk}}} \mathcal{L}_{\text{AR}}(x_{\mathbf{trk}}; \theta_{\text{top}})$. The attacker optimizes a dummy soft label tensor $\tilde{\mathbf{Z}} \in \mathbb{R}^{B\times L\times V}$ (logits in label space), which is transformed into a distribution via temperature-scaled softmax:

$$\mathbf{P} \;=\; \text{Softmax}\Big(\tilde{\mathbf{Z}}/\tau\Big) \in [0,1]^{B\times L\times V}, \tag{8}$$

where $\tau > 0$ is a temperature hyperparameter. For the *next-token soft target distribution*, the per-token cross-entropy is

$$\ell_{b,\ell} \;=\; -\sum_{v=1}^{V} \mathbf{P}_{b,\ell+1,v}\, \log \text{Softmax}\Big(\tilde{\mathbf{L}}_{b,\ell,:}\Big)_v . \tag{9}$$

Masking is applied to ignore padding positions (and effectively excludes the final token by using the $\ell+1$ target):

$$\mathcal{L}_{\text{soft}}(x_{\text{trk}}, \tilde{\mathbf{Z}}; \boldsymbol{\theta}_{\text{pro}}) \;=\; \frac{1}{N} \sum_{b=1}^{B} \sum_{\ell=1}^{L-1} \mathbf{M}_{b,\ell+1}\, \ell_{b,\ell}, \tag{10}$$

where $N = \sum_{b=1}^{B} \sum_{\ell=1}^{L-1} \mathbf{M}_{b,\ell+1}$. From $\mathcal{L}_{\text{soft}}$, the attacker computes the gradient with respect to the observed smashed data x_{trk}:

$$\tilde{\mathbf{g}}_{\text{trk}}(\tilde{\mathbf{Z}}) = \nabla_{x_{\mathbf{trk}}} \mathcal{L}_{\text{soft}}(x_{\text{trk}}, \tilde{\mathbf{Z}}; \theta_{\text{pro}}) \in \mathbb{R}^{B\times L\times D}. \tag{11}$$

The attack then minimizes a discrepancy between the induced gradient $\tilde{\mathbf{G}}(\tilde{\mathbf{Z}})$ and the observed gradient $\mathbf{G}$, **where $\tilde{\mathbf{G}}(\tilde{\mathbf{Z}}) \triangleq \tilde{\mathbf{g}}_{\text{trk}}(\tilde{\mathbf{Z}})$ is exactly the gradient defined in Eq. 11; we adopt the uppercase notation hereafter to emphasize its role as a tensor being matched against $\mathbf{G}$.** Taking TAG as an illustrative example, we define the gradient-matching objective as

$$\mathcal{L}_{\text{TAG}}(\tilde{\mathbf{Z}}) = \sum_{b=1}^{B} \sum_{\ell=1}^{L} \mathbf{M}_{b,\ell} \Big(\beta \big\| \tilde{\mathbf{G}}_{b,\ell,:}(\tilde{\mathbf{Z}}) - \mathbf{G}_{b,\ell,:} \big\|_2^2 + (1-\beta) \big\| \tilde{\mathbf{G}}_{b,\ell,:}(\tilde{\mathbf{Z}}) - \mathbf{G}_{b,\ell,:} \big\|_1 \Big), \tag{12}$$

where $\beta \in [0,1]$ controls the trade-off between the ℓ_2 and ℓ_1 discrepancies. The key optimization variable is $\tilde{\mathbf{Z}}$, but the matching loss depends on $\tilde{\mathbf{Z}}$ only through the intermediate gradient $\mathbf{G}(\tilde{\mathbf{Z}}) =$

$\nabla_{x_{\mathrm{trk}}}\mathcal{L}_{\mathrm{soft}}$. Therefore, updating $\tilde{\mathbf{Z}}$ requires differentiating *through a gradient*, i.e., computing a mixed second-order derivative:

$$\begin{aligned}\nabla_{\tilde{\mathbf{Z}}}\mathcal{L}_{\mathrm{TAG}}(\tilde{\mathbf{Z}}) &= \frac{\partial \mathcal{L}_{\mathrm{TAG}}}{\partial \tilde{\mathbf{G}}} \cdot \underbrace{\frac{\partial \tilde{\mathbf{G}}}{\partial \tilde{\mathbf{Z}}}}_{\text{requires 2nd-order terms}} \\ &= \frac{\partial \mathcal{L}_{\mathrm{TAG}}}{\partial \tilde{\mathbf{G}}} \cdot \frac{\partial}{\partial \tilde{\mathbf{Z}}}\Big(\nabla_{x_{\mathrm{trk}}}\mathcal{L}_{\mathrm{soft}}(x_{\mathrm{trk}}, \tilde{\mathbf{Z}}; \boldsymbol{\theta}_{\mathrm{pro}})\Big) \\ &= \frac{\partial \mathcal{L}_{\mathrm{TAG}}}{\partial \tilde{\mathbf{G}}} \cdot \nabla^2_{\tilde{\mathbf{Z}},\, x_{\mathbf{trk}}}\mathcal{L}_{\mathrm{soft}}(x_{\mathrm{trk}}, \tilde{\mathbf{Z}}; \boldsymbol{\theta}_{\mathrm{pro}}).\end{aligned} \tag{13}$$

In the implementation, we obtain $\mathbf{G}(\tilde{\mathbf{Z}})$ via automatic differentiation while retaining the computational graph (`create_graph=True`)[2], which enables backpropagation from $\mathcal{L}_{\mathrm{TAG}}$ to $\tilde{\mathbf{Z}}$.

Finally, the adversary decodes discrete token predictions by $\tilde{\mathrm{Y}}_{b,\ell} = \arg\max_{v\in[V]} \tilde{\mathbf{Z}}_{b,\ell,v}$, and ignores positions where $\mathbf{M}_{b,\ell} = 0$.

We now return to the main theme of this section: when optimizing the dummy label logits $\tilde{\mathbf{Z}}$, samples within the same mini-batch are *batch-separable* and hence do not affect each other, except through the global normalization constant N, which is known to the adversary.

The key observation comes directly from the computational graph. For each sample $b \in [B]$, the server-side proxy Top segment and the shifted soft-target loss induce a per-sample subgraph

$$\begin{aligned} x^{(b)}_{\mathrm{trk}},\, \mathbf{M}^{(b)} \;&\to \tilde{\mathbf{L}}^{(b)} \;=\; F\big(x^{(b)}_{\mathrm{trk}},\, \mathbf{M}^{(b)};\, \theta_{\mathrm{pro}}\big), \\ \tilde{\mathbf{Z}}^{(b)} \;&\to \mathbf{P}^{(b)} \;=\; \mathrm{Softmax}\big(\tilde{\mathbf{Z}}^{(b)}/\tau\big), \\ \big(\tilde{\mathbf{L}}^{(b)},\, \mathbf{P}^{(b)},\, \mathbf{M}^{(b)}\big) \;&\to\; \mathcal{L}^{(b)}_{\mathrm{soft}} \;\to\; \tilde{\mathbf{g}}^{(b)}_{\mathrm{trk}} \;=\; \nabla_{x^{(b)}_{\mathrm{trk}}}\mathcal{L}_{\mathrm{soft}}. \end{aligned} \tag{14}$$

and these subgraphs are disjoint across different b's because modern LLM implementations execute the forward/backward pass independently per example and only stack tensors along the batch dimension for efficiency. Concretely, the masked shifted loss decomposes as

$$\mathcal{L}_{\mathrm{soft}}(x_{\mathrm{trk}}, \tilde{\mathbf{Z}}; \theta_{\mathrm{pro}}) = \frac{1}{N}\sum_{b=1}^{B}\sum_{\ell=1}^{L-1} \mathbf{M}_{b,\ell+1}\, \ell_{b,\ell} = \frac{1}{N}\sum_{b=1}^{B} \mathcal{L}^{(b)}_{\mathrm{soft}}, \tag{15}$$

where $\mathcal{L}^{(b)}_{\mathrm{soft}} \triangleq \sum_{\ell=1}^{L-1} \mathbf{M}_{b,\ell+1}\, \ell_{b,\ell}$ depends only on $(\mathbf{x}^{(b)}_{\mathrm{trk}}, \mathbf{M}^{(b)}, \tilde{\mathbf{Z}}^{(b)})$. As a result, the dummy gradient tensor factorizes across the batch:

$$\tilde{\mathbf{g}}_{\mathrm{trk}}(\tilde{\mathbf{Z}}) = \nabla_{x_{\mathrm{trk}}}\mathcal{L}_{\mathrm{soft}}(x_{\mathrm{trk}}, \tilde{\mathbf{Z}}; \theta_{\mathrm{pro}}) = \frac{1}{N}\Big[\nabla_{x^{(1)}_{\mathrm{trk}}}\mathcal{L}^{(1)}_{\mathrm{soft}}, \ldots, \nabla_{x^{(B)}_{\mathrm{trk}}}\mathcal{L}^{(B)}_{\mathrm{soft}}\Big], \tag{16}$$

and there are no cross-sample Jacobian terms. More explicitly, for any $b \neq b'$,

$$\frac{\partial\, \tilde{\mathbf{g}}^{(b)}_{\mathrm{trk}}(\tilde{\mathbf{Z}})}{\partial\, \tilde{\mathbf{Z}}^{(b')}} = \mathbf{0}, \;\; \text{equivalently} \;\; \nabla^2_{\tilde{\mathbf{Z}}^{(b')},\, x^{(b)}_{\mathrm{trk}}}\, \mathcal{L}_{\mathrm{soft}} = \mathbf{0}.$$

Therefore, the gradient-matching objective (e.g., $\mathcal{L}_{\mathrm{TAG}}$) decomposes into a sum of per-sample objectives (up to the shared scalar $1/N$):

$$\begin{aligned}\mathcal{L}_{\mathrm{TAG}}(\tilde{\mathbf{Z}}) &= \sum_{b=1}^{B} \mathcal{L}^{(b)}_{\mathrm{TAG}}(\tilde{\mathbf{Z}}^{(b)}; \mathbf{G}^{(b)}) \\ \Rightarrow \nabla_{\tilde{\mathbf{Z}}^{(b)}}\mathcal{L}_{\mathrm{TAG}}(\tilde{\mathbf{Z}}) &= \nabla_{\tilde{\mathbf{Z}}^{(b)}}\mathcal{L}^{(b)}_{\mathrm{TAG}}(\tilde{\mathbf{Z}}^{(b)}; \mathbf{G}^{(b)}),\end{aligned} \tag{17}$$

meaning that optimizing $\tilde{\mathbf{Z}}^{(b)}$ for one sample cannot change the matching loss terms of any other sample $b' \neq b$. The only batch-level coupling is the normalization $N = \sum_{b=1}^{B}\sum_{\ell=1}^{L-1} \mathbf{M}_{b,\ell+1}$,

[2] In PyTorch, setting `create_graph=True` in `torch.autograd.grad` constructs a computation graph for the differentiation operation itself, i.e., it treats the returned gradient as a differentiable output of an autograd-tracked operator.

which is fully known to the adversary because $\mathbf{M}$ must be provided to the server to execute attention in the trunk/top forward pass. Consequently, GMA-SL admits *batch-separable* gradient matching: the attacker may conceptually solve B independent subproblems (one per sample) within each mini-batch, without cross-sample interference.

B Details of the vMF Mechanism

This section provides additional details on the vMF mechanism. Sections B.1 and B.2 describe our implementation of vMF sampling, including the tangent-normal decomposition and the rejection sampling procedure, respectively. Section B.3 provides the proof of the theoretical result stated in the main text.

B.1 Tangent-Normal Decomposition in vMF Sampling

The von Mises-Fisher privacy mechanism adopts the von Mises-Fisher (vMF) distribution, named after von Mises and Fisher. This distribution is defined over the unit hypersphere S^{n-1}.

Definition 3 (The vMF Distribution). *The vMF distribution on $\mathbb{S}^{n-1}$ with mean direction $\mu \in \mathbb{S}^{n-1}$ and concentration parameter $\kappa \geq 0$ is defined by the following density:* $\mathrm{vMF}(\mu, \kappa)[\mathbf{x}] = C_{\mathrm{vMF}}(n, \kappa) \cdot \exp\left(\kappa \cdot \mu^\top \mathbf{x}\right)$, *where* $\mathbf{x} \in \mathbb{S}^{n-1}$ *and* $C_{\mathrm{vMF}}(n, \kappa)$ *is the normalization constant.*

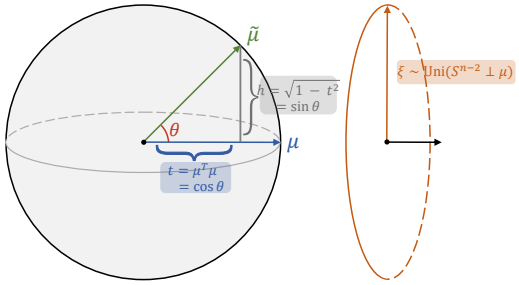


Figure 9: *Tangent-normal decomposition in vMF sampling.* The sampled direction $\tilde{\mu}$ is decomposed into an axial component $t = \tilde{\mu}^\top \mu = \cos\theta$ along μ and a tangential component of magnitude $\sqrt{1 - t^2} = \sin\theta$, where $\xi \sim \mathrm{Uni}(\mathbb{S}^{n-2} \perp \mu)$.

To sample a new direction vector $\tilde{\mu} \in \mathbb{S}^{n-1}$ under the vMF mechanism centered at $\mu \in \mathbb{S}^{n-1}$, we use the tangent-normal decomposition. As illustrated in Fig. 9, the sampled vector $\tilde{\mu}$ is decomposed into a component along μ and a component in the subspace orthogonal to μ. Let $t = \tilde{\mu}^\top \mu = \cos\theta$, where θ is the angle between $\tilde{\mu}$ and μ. Then the orthogonal component has magnitude $h = \sqrt{1 - t^2} = \sin\theta$. If $\xi \sim \mathrm{Unif}(\mathbb{S}^{n-2} \perp \mu)$ is a unit vector uniformly sampled from the subsphere orthogonal to μ, the sampled direction can be written as $\tilde{\mu} = t\mu + \sqrt{1 - t^2}\,\xi$ or equivalently $\tilde{\mu} = \cos\theta\,\mu + \sin\theta\,\xi$. Therefore, vMF sampling is separated into two parts: sampling the axial projection t (or equivalently the angle θ) and sampling the tangential direction ξ uniformly on the orthogonal subsphere.

B.2 Rejection Sampling for t

After the tangent-normal decomposition, vMF sampling reduces to drawing the axial variable $t \in [-1, 1]$. In our implementation, t is generated by a rejection sampling scheme with a Beta proposal. Here, $p = n$ denotes the ambient dimension of the hypersphere. Specifically, we first sample

$$y \sim \mathrm{Beta}\left(\frac{p-1}{2}, \frac{p-1}{2}\right), \tag{18}$$

and transform it into a candidate $t = \frac{1-(1+b_0)y}{1-(1-b_0)y}$, where $b_0 = \frac{-2\kappa + \sqrt{4\kappa^2 + (p-1)^2}}{p-1}$. Following the standard construction, we further define

$$a = \frac{p - 1 + 2\kappa + \sqrt{4\kappa^2 + (p-1)^2}}{4}, \; d = \frac{4ab_0}{1 + b_0} - (p-1)\log(p-1). \tag{19}$$

For each proposed sample, let $s = \frac{2ab_0}{1-(1-b_0)y}$. The candidate is accepted if $\log u \leq (p-1)\log s - s + d$, where $u \sim \mathrm{Unif}(0, 1)$. Otherwise, the sample is rejected and the procedure is repeated. This yields a valid sample for t, which is then combined with the tangential direction to form the final vMF sample.

B.3 Proofs related to the vMF Mechanism

Theorem 2 (The vMF Mechanism Satisfies εd_2-Privacy). *Let $\mathcal{M}_{\mathrm{vMF}}$ be the vMF mechanism, $\mathcal{M}_{\mathrm{vMF}}$ satisfies εd_2-metric differential privacy, where*

$$d_2(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2.$$

For any $\mathbf{u}, \mathbf{v} \in \mathbb{S}^{d-1}$ and any measurable set $S \subseteq \mathbb{S}^{d-1}$,

$$\Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{u}) \in S] \leq \exp\big(\varepsilon\, d_2(\mathbf{u}, \mathbf{v})\big)\, \Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{v}) \in S].$$

Proof of Theorem 2. Let $\mathbf{u}, \mathbf{v} \in \mathbb{S}^{d-1}$ be arbitrary unit vectors, and let $S \subseteq \mathbb{S}^{d-1}$ be any measurable set. For any $\mathbf{z} \in S$, the vMF density satisfies

$$\frac{p_{\mathbf{u}}(\mathbf{z})}{p_{\mathbf{v}}(\mathbf{z})} = \frac{C_d(\varepsilon) \exp(\epsilon\, \mathbf{u}^\top \mathbf{z})}{C_d(\epsilon) \exp(\epsilon\, \mathbf{v}^\top \mathbf{z})} = \exp\big(\epsilon(\mathbf{u} - \mathbf{v})^\top \mathbf{z}\big).$$

By the Cauchy–Schwarz inequality,

$$(\mathbf{u} - \mathbf{v})^\top \mathbf{z} \leq \|\mathbf{u} - \mathbf{v}\|_2\, \|\mathbf{z}\|_2.$$

Since $\mathbf{z} \in \mathbb{S}^{d-1}$, we have $\|\mathbf{z}\|_2 = 1$. Hence,

$$\frac{p_{\mathbf{u}}(\mathbf{z})}{p_{\mathbf{v}}(\mathbf{z})} \leq \exp\big(\varepsilon \|\mathbf{u} - \mathbf{v}\|_2\big) = \exp\big(\varepsilon d_2(\mathbf{u}, \mathbf{v})\big).$$

Integrating both sides over $\mathbf{z} \in S$, we obtain

$$\begin{aligned}\Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{u}) \in S] &= \int_S p_{\mathbf{u}}(\mathbf{z})\, d\mathbf{z} \leq \exp\big(\varepsilon d_2(\mathbf{u}, \mathbf{v})\big) \int_S p_{\mathbf{v}}(\mathbf{z})\, d\mathbf{z} \\ &= \exp\big(\varepsilon d_2(\mathbf{u}, \mathbf{v})\big)\, \Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{v}) \in S].\end{aligned}$$

Therefore, $\mathcal{M}_{\mathrm{vMF}}$ satisfies εd_2-metric differential privacy. □

Theorem 3 (The vMF Mechanism Satisfies $\varepsilon d_\angle$-Privacy). *$\mathcal{M}_{\mathrm{vMF}}$ also satisfies $\varepsilon d_\angle$-metric differential privacy, where*

$$d_\angle(\mathbf{u}, \mathbf{v}) = \arccos(\mathbf{u}^\top \mathbf{v}).$$

For any $\mathbf{u}, \mathbf{v} \in \mathbb{S}^{d-1}$ and any measurable set $S \subseteq \mathbb{S}^{d-1}$,

$$\Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{u}) \in S] \leq \exp\left(\varepsilon d_\angle(\mathbf{u}, \mathbf{v})\right) \Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{v}) \in S].$$

Proof of Theorem 3. From Theorem 2, the vMF mechanism satisfies

$$\Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{u}) \in S] \leq \exp\left(\varepsilon \|\mathbf{u} - \mathbf{v}\|_2\right) \Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{v}) \in S].$$

It remains to show that $\|\mathbf{u} - \mathbf{v}\|_2 \leq d_\angle(\mathbf{u}, \mathbf{v})$. Let $\theta = d_\angle(\mathbf{u}, \mathbf{v}) \in [0, \pi]$. Since $\mathbf{u}$ and $\mathbf{v}$ are unit vectors,

$$\|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{2 - 2\mathbf{u}^\top \mathbf{v}} = \sqrt{2 - 2\cos\theta} = 2 \sin\frac{\theta}{2}.$$

For any $\theta \in [0, \pi]$, we have $2 \sin\frac{\theta}{2} \leq \theta$. Therefore, $\|\mathbf{u} - \mathbf{v}\|_2 \leq d_\angle(\mathbf{u}, \mathbf{v})$. Substituting this inequality into the result of Theorem 2 yields

$$\exp\left(\varepsilon \|\mathbf{u} - \mathbf{v}\|_2\right) \leq \exp\left(\varepsilon d_\angle(\mathbf{u}, \mathbf{v})\right).$$

Hence, $\Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{u}) \in S] \leq \exp\left(\varepsilon d_\angle(\mathbf{u}, \mathbf{v})\right) \Pr[\mathcal{M}_{\mathrm{vMF}}(\mathbf{v}) \in S]$. Thus, the mechanism satisfies $\varepsilon d_\angle$-metric differential privacy. □

Proof of Theorem 1. Since $\mathbf{g} = \|\mathbf{g}\|_2 \mathbf{u}$ and $\tilde{\mathbf{g}} = \|\mathbf{g}\|_2 \tilde{\mathbf{u}}$, the perturbation noise can be written as

$$\mathbf{n} = \tilde{\mathbf{g}} - \mathbf{g} = \|\mathbf{g}\|_2 (\tilde{\mathbf{u}} - \mathbf{u}).$$

Therefore, $\|\mathbf{n}\|_2 = \|\mathbf{g}\|_2 \|\tilde{\mathbf{u}} - \mathbf{u}\|_2$. Because both $\mathbf{u}$ and $\tilde{\mathbf{u}}$ are unit vectors, their Euclidean distance is at most 2:

$$\|\tilde{\mathbf{u}} - \mathbf{u}\|_2 \leq \|\tilde{\mathbf{u}}\|_2 + \|\mathbf{u}\|_2 = 2.$$

Thus, $\|\mathbf{n}\|_2 \leq 2\|\mathbf{g}\|_2$. It follows that

$$\mathrm{ASNR} = \frac{\|\mathbf{g}\|_2}{\|\mathbf{n}\|_2} \geq \frac{\|\mathbf{g}\|_2}{2\|\mathbf{g}\|_2} = \frac{1}{2}.$$

The equality holds when $\|\tilde{\mathbf{u}} - \mathbf{u}\|_2 = 2$, which occurs if and only if $\tilde{\mathbf{u}} = -\mathbf{u}$. Hence, the bound is tight. □

Algorithm 1 Selective Autoregressive Supervision

Require: Label sequence $\mathbf{y} = (y_1, \ldots, y_T)$; predictive distributions $\{p_t\}_{t=1}^{T}$ from the first forward pass; number of token groups k; sampling ratio ρ
Ensure: Supervision mask $\mathbf{m} = (m_1, \ldots, m_T)$

Construct autoregressive targets by left-shifting $\mathbf{y}$
for $t = 1$ **to** T **do**
  // Compute token entropy
  $H_t \leftarrow -\sum_{v \in \mathcal{V}} p_t(v) \log p_t(v)$
end for
Sort token positions $\{1, \ldots, T\}$ by H_t in descending order
Partition the sorted positions into three strata: $\mathcal{H}^{\text{high}}$, $\mathcal{H}^{\text{mid}}$, and $\mathcal{H}^{\text{low}}$
Randomly and evenly split each stratum into k disjoint subgroups: $\{\mathcal{H}_i^{\text{high}}\}_{i=1}^{k}$, $\{\mathcal{H}_i^{\text{mid}}\}_{i=1}^{k}$, and $\{\mathcal{H}_i^{\text{low}}\}_{i=1}^{k}$
for $i = 1$ **to** k **do**
  $\mathcal{G}_i \leftarrow \mathcal{H}_i^{\text{high}} \cup \mathcal{H}_i^{\text{mid}} \cup \mathcal{H}_i^{\text{low}}$
end for
Sample r uniformly from $\{1, \ldots, k\}$ and designate $\mathcal{G}_r$ as the full-supervision group
// Full supervision
Initialize $m_t \leftarrow 0$ for all $t = 1, \ldots, T$
for $t \in \mathcal{G}_r$ **do**
  $m_t \leftarrow 1$
end for
for $j = 1$ **to** k **do**
  if $j \neq r$ **then**
    Randomly sample $\mathcal{S}_j \subseteq \mathcal{G}_j$ with sampling ratio ρ
    // Partial supervision
    for $t \in \mathcal{S}_j$ **do**
      $m_t \leftarrow 1$
    end for
  end if
end for
return $\mathbf{m}$

C Details of Selective Autoregressive Supervision

C.1 Algorithmic Implementation of Selective Autoregressive Supervision

To complement the description in the main text, we provide the detailed pseudocode implementation of Selective Autoregressive Supervision in Algorithm 1.

C.2 Last-k Tokens in Selective Autoregressive Supervision

We next discuss a practical implementation detail of SAS from the client's perspective.

While Selective Autoregressive Supervision is designed to perturb the supervision structure exposed through the split-interface gradient, its implementation should also preserve the stealthiness of the defense.

In particular, if the last k consecutive target tokens are all excluded from the loss computation, then the corresponding gradient signals on a contiguous block of k token representations in x_{trk} become identically zero. Such a structured zero-gradient suffix can serve as an explicit artifact of the defense and may reveal the existence of selective masking to an adversarial server.

To see this more formally, consider the autoregressive loss with a binary supervision mask $m_t \in \{0, 1\}$:

$$\mathcal{L}_M = \frac{1}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \ell_t, \ \ell_t = -\log p_t(y_t), \tag{20}$$

where p_t denotes the predictive distribution for the target token y_t. Let h_i denote the split-interface representation at position i, i.e., the i-th token representation in x_{trk}.

Due to the causal structure of autoregressive decoding, the representation h_i can only influence the losses of future or aligned target positions. Abstractly, we may write

$$\frac{\partial \mathcal{L}_M}{\partial h_i} = \frac{1}{\sum_{t=1}^{T} m_t} \sum_{t \in \mathcal{D}(i)} m_t \frac{\partial \ell_t}{\partial h_i}, \tag{21}$$

where $\mathcal{D}(i)$ denotes the set of supervised prediction positions whose computation depends on h_i. For the last few token positions, this dependency set contains only a small number of subsequent losses. Therefore, if the last k target positions are all masked out, namely $m_{T-k+1} = m_{T-k+2} = \cdots = m_T = 0$, then for the corresponding suffix positions whose only downstream supervised losses lie in this masked region, we obtain $\frac{\partial \mathcal{L}_M}{\partial h_i} = 0$. Consequently, the gradient returned to the server-side Trunk segment contains a contiguous zero-gradient block over the suffix of x_{trk}. This phenomenon is illustrated in Fig. 10.

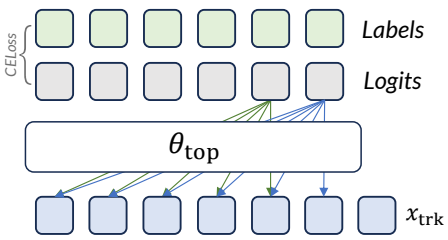


Figure 10: *Illustration of the zero-gradient suffix induced by masking the last k target tokens in selective autoregressive supervision.* The arrows indicate the direction of gradient propagation.

This artifact is undesirable because it weakens the concealment of the defense. From the client's perspective, the goal of SAS is not only to introduce a supervision mismatch against gradient-matching attacks, but also to avoid leaving obvious structural signatures in the exposed gradient. A consecutive zero-gradient suffix is particularly problematic because it is independent of token content and can be detected directly from the norm pattern of the interface gradient. An adversary may observe that $\left\| \frac{\partial \mathcal{L}_M}{\partial h_i} \right\|_2 = 0$ for several consecutive positions near the end of the sequence, and infer that these positions were systematically removed from supervision.

To avoid this issue, SAS should ensure that the last k token positions are not simultaneously excluded from the masked loss. This constraint guarantees that at least one suffix target token contributes to the loss, thereby preventing a fully zero-gradient suffix from appearing in x_{trk}. To prevent such a zero-gradient suffix, we enforce the last k tokens to be valid supervised positions in our implementation. We set $k = 5$ throughout our experiments.

D More Detailed Experiments and Analyses

D.1 Definitions of Gradient Utility Metrics

To quantify the utility of privatized gradients, we use four complementary metrics: the Amplitude Signal-to-Noise Ratio (ASNR), ASNR@10%, Recall@10% (R@10%), and Jaccard@10% (J@10%). Let $g \in \mathbb{R}^{LD}$ denote the original gradient and $\tilde{g} \in \mathbb{R}^{LD}$ the privatized gradient. The injected noise is defined as $n = \tilde{g} - g$.

ASNR. The Amplitude Signal-to-Noise Ratio measures the overall gradient magnitude relative to the perturbation magnitude:

$$\mathrm{ASNR}(g, \tilde{g}) = \frac{\|g\|_2}{\|n\|_2}. \tag{22}$$

ASNR@10%. Let $S_{10}(g)$ denote the index set of the top 10% entries of g ranked by absolute value. ASNR@10% is computed on this subset:

$$\mathrm{ASNR@10\%}(g, \tilde{g}) = \frac{\|g_{S_{10}(g)}\|_2}{\|n_{S_{10}(g)}\|_2}. \tag{23}$$

Recall@10%. Let $S_{10}(g)$ and $S_{10}(\tilde{g})$ be the top 10% index sets of g and $\tilde{g}$, respectively. Recall@10% measures the fraction of salient coordinates in the original gradient that are retained after

privatization:

$$\mathrm{R@10\%}(g, \tilde{g}) = \frac{|S_{10}(g) \cap S_{10}(\tilde{g})|}{|S_{10}(g)|}. \tag{24}$$

Jaccard@10%. Jaccard@10% measures the overlap between the two top-10% index sets:

$$\mathrm{J@10\%}(g, \tilde{g}) = \frac{|S_{10}(g) \cap S_{10}(\tilde{g})|}{|S_{10}(g) \cup S_{10}(\tilde{g})|}. \tag{25}$$

ASNR and ASNR@10% evaluate magnitude preservation, while R@10% and J@10% measure support preservation.

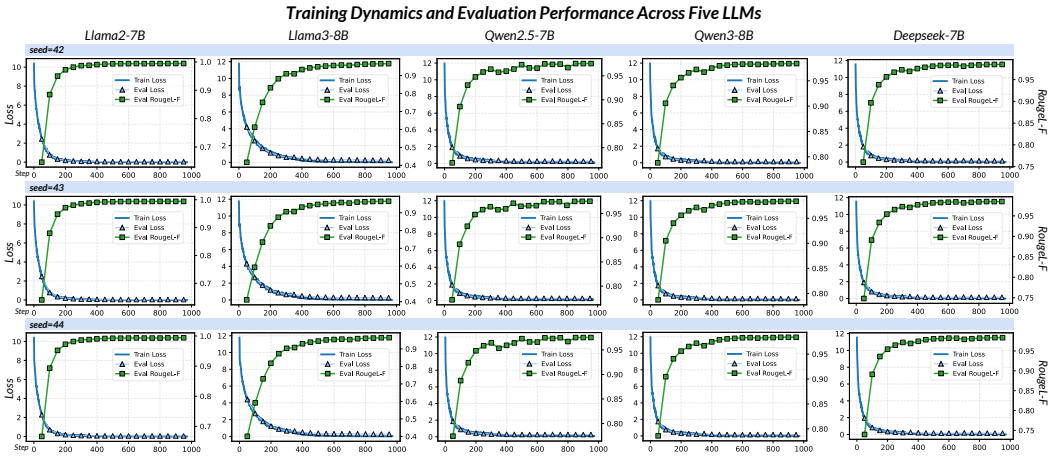


Figure 11: *Training dynamics of the learning-based inversion decoder across five LLMs.* Shown are the training loss, evaluation loss, and evaluation RougeL-F over training steps under three random seeds.

D.2 Training Details of the Learning-based Inversion Decoder

Here we present the training details of the learning-based inversion decoder used in the semi-white-box Forward Inversion Paradigm (SIP) Chen et al. (2024), which corresponds to the "SIP-only" setting in our experiments. The goal of this decoder is to directly reconstruct the original input token sequence from the smashed data produced by the client-side Bottom segment.

Following the setting of the original work, we use the open-source CNN-DailyMail News Text Summarization dataset as the auxiliary training data See et al. (2017). Specifically, the training data is derived from the CNN-DailyMail news summarization corpus.

The inversion decoder takes the smashed data generated by the Bottom segment as input. In our setting, the Bottom segment contains 6 decoder layers. Given the hidden representations with embedding dimension D, the inverter uses a single-layer unidirectional GRU Cho et al. (2014) with hidden size 4096 to model the sequential dependency of the intermediate activations. The GRU output at each token position is then passed through a linear projection layer, which maps the hidden state to the vocabulary space and produces token-level logits. A dropout rate of 0.1 is applied after the GRU layer during training.

The inverter is trained in an auto-encoder-style manner. The Bottom segment of the target language model is used as a frozen encoder to generate intermediate representations from the auxiliary text data, while only the inversion decoder is optimized. The training objective is token-level cross-entropy loss between the decoder output and the original input token sequence. In this way, the decoder learns to map the smashed data back to the corresponding textual tokens.

We train the inversion decoder for 3 epochs with a batch size of 32. The model is optimized using AdamW with a learning rate of 1e-3 and a weight decay of 1e-5. After training, the learned inverter

can be directly applied to the smashed data observed during split fine-tuning to produce an initial reconstruction of the client's private input sequence. Fig. 11 illustrates the training dynamics of the decoder on five LLMs Touvron et al. (2023); Grattafiori et al. (2024); Yang et al. (2024; 2025); Bi et al. (2024).

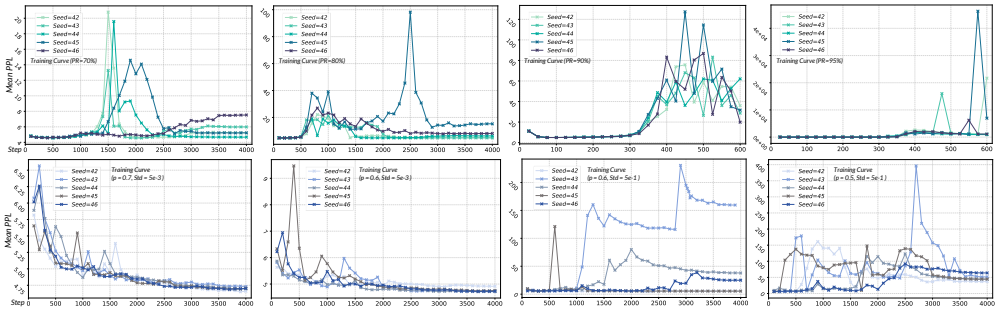


Figure 12: *Training Curves of Gradient Pruning and Gradient Dropout.*

D.3 Training Curves of GP and GD

Fig. 12 illustrates the training curves of GP and GD. It can be observed that, to achieve strong privacy protection with GP or GD, the training process becomes extremely unstable, making it challenging to achieve a favorable privacy-utility trade-off.

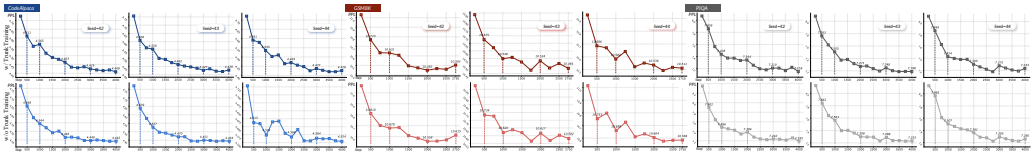


Figure 13: *Comparison between the training dynamics with and without Trunk segment updates on Llama3-8B across three datasets.* Updating the Trunk segment generally leads to lower validation perplexity across random seeds.

D.4 Ablation Details of Trunk Training

Fig. 13 illustrates the differences in training dynamics with and without Trunk segment updates, evaluated on Llama3-8B across three datasets. For each dataset, the upper panel corresponds to training with Trunk segment updates, which consistently yields better performance.

D.5 Extended Defense Experiments

In the main text, we evaluate defense performance using only ROUGE-L-F and METEOR. Here, we provide additional results for ROUGE-1-F, ROUGE-2-F, and Token Recovery Rate (TRR), as well as results over more training steps. Tab. 6, 7, and 8 report the results for Llama-3-8B, Llama-2-7B, and DeepSeek-LLM-7B, respectively. We further report the defense performance against the BiSR(b) paradigm when cosine distance is used as the matching objective, as shown in Tab. 9.

Dataset	Method	Final PPL	PPL@500steps	PPL@1000steps	RougeL-F	Rouge1-F	Rouge2-F	Meteor	TRR
Code-Alpaca	Grad Pruning	5.292±.538	4.595±.001	4.934±.017	.995±.001	.995±.001	.981±.001	.987±.001	.990±.000
	GradSeq-LDP(10240)	4.670±.002	4.962±.055	4.782±.043	.564±.032	.567±.037	.413±.020	.570±.016	.634±.007
	GradSeq-LDP(12288)	4.611±.037	4.915±.022	4.804±.050	.657±.029	.661±.030	.517±.042	.664±.042	.717±.041
	Grad Dropout	4.782±.104	5.341±.158	4.954±.041	.942±.008	.942±.008	.915±.012	.940±.007	.943±.013
	Gradient Mirage(512)	4.451±.001	4.671±.022	4.610±.008	**.202±.002**	**.208±.003**	**.108±.001**	**.226±.000**	**.255±.002**
	Gradient Mirage(1024)	4.421±.008	4.603±.008	4.552±.013	**.351±.022**	**.361±.022**	**.177±.020**	**.341±.017**	**.371±.009**
	Top-Only Training	4.831±.005	5.190±0.006	4.970±.004	-	-	-	-	-
	Standard Training	4.444±.004	4.504±.012	4.447±.005	1.000±.000	1.000±.000	.989±.002	.991±.002	.992±.001
PIQA	Grad Pruning	11.500±2.161	9.229±.598	13.624±.095	.959±.011	.959±.011	.908±.023	.935±.017	.938±.013
	GradSeq-LDP(10240)	7.738±.056	7.443±.025	7.212±.017	.799±.034	.804±.033	.613±.039	.760±.048	.758±.028
	GradSeq-LDP(12288)	7.202±.019	7.726±.040	7.416±.021	.805±.033	.810±.032	.626±.056	.769±.049	.762±.039
	Grad Dropout	7.758±.029	9.111±.265	8.316±.145	.950±.006	.950±.006	.867±.002	.912±.003	.906±.008
	Gradient Mirage(512)	7.404±.020	8.120±.042	7.894±.100	**.209±.017**	**.215±.017**	**.040±.008**	**.137±.013**	**.183±.013**
	Gradient Mirage(1024)	7.191±.033	7.824±.043	7.456±.035	**.448±.020**	**.455±.019**	**.145±.014**	**.306±.025**	**.374±.007**
	Top-Only Training	8.874±.018	9.613±.027	9.269±.029	-	-	-	-	-
	Standard Training	7.396±.013	7.338±.036	7.106±.017	.993±.004	.993±.004	.969±.010	.975±.007	.978±.005
GSM8K	Grad Pruning	12.227±1.131	10.769±.059	10.973±.301	.997±.003	.997±.006	.994±.003	.996±.012	.996±.002
	GradSeq-LDP(10240)	11.038±.025	11.712±.152	11.431±.005	.904±.014	.904±.014	.843±.013	.923±.007	.914±.007
	GradSeq-LDP(12288)	11.122±.039	11.669±.114	11.423±.058	.885±.019	.886±.018	.821±.024	.910±.013	.901±.013
	Grad Dropout	12.477±.972	12.556±.153	12.365±.172	.949±.010	.949±.011	.904±.015	.940±.010	.927±.011
	Gradient Mirage(512)	10.643±.031	10.958±.051	10.806±.055	**.112±.008**	**.119±.007**	**.011±.005**	**.070±.006**	**.079±.008**
	Gradient Mirage(1024)	10.500±.011	10.710±.042	10.587±.034	**.363±.012**	**.387±.009**	**.087±.007**	**.213±.013**	**.243±.017**
	Top-Only Training	11.999±.007	12.283±.005	12.075±.004	-	-	-	-	-
	Standard Training	10.588±.003	10.471±.059	10.416±.016	1.000±.000	1.000±.000	.993±.001	.995±.001	.996±.000

Table 6: *Comparison between Gradient Mirage and existing privacy-preserving methods with Llama3-8B.*

Dataset	Method	Final PPL	PPL@500steps	PPL@1000steps	RougeL-F	Rouge1-F	Rouge2-F	Meteor	TRR
Code-Alpaca	Grad Pruning	5.000±.089	4.864±.088	4.765±.044	.993±.003	.993±.003	.988±.006	.989±.005	.995±.003
	GradSeq-LDP(10240)	4.804±.083	5.141±.149	4.991±.155	.775±.033	.776±.034	.656±.048	.741±.043	.886±.018
	GradSeq-LDP(12288)	4.853±.131	5.132±.143	5.024±.172	.820±.037	.821±.037	.725±.052	.795±.043	.909±.018
	Grad Dropout	6.979±1.575	7.329±.929	82.223±129.115	.902±.017	.902±.016	.859±.026	.890±.016	.932±.008
	Gradient Mirage(512)	4.817±.098	5.186±.124	5.066±.068	**.095±.012**	**.097±.012**	**.016±.002**	**.112±.005**	**.199±.010**
	Gradient Mirage(1024)	4.772±.049	5.031±.136	5.061±.124	**.174±.020**	**.177±.019**	**.042±.007**	**.146±.008**	**.297±.004**
	Top-Only Training	4.961±.040	5.231±.158	5.025±.083	-	-	-	-	-
	Standard Training	4.731±.018	4.776±.108	4.669±.035	.990±.003	.990±.003	.981±.006	.982±.005	.992±.003
PIQA	Grad Pruning	6.501±.615	5.849±.055	5.576±.188	.996±.004	.996±.004	.991±.009	.995±.005	.997±.003
	GradSeq-LDP(10240)	6.302±.656	6.325±.248	6.280±.208	.792±.038	.793±.038	.635±.057	.744±.043	.861±.008
	GradSeq-LDP(12288)	5.851±.049	6.147±.067	6.172±.256	.826±.017	.826±.017	.691±.026	.786±.023	.883±.005
	Grad Dropout	7.757±1.772	11.069±.524	8.452±1.022	.813±.020	.815±.017	.743±.026	.798±.019	.859±.008
	Gradient Mirage(512)	5.631±.130	5.770±.095	5.767±.052	**.047±.015**	**.047±.016**	**.001±.002**	**.046±.001**	**.115±.007**
	Gradient Mirage(1024)	5.561±.232	5.748±.061	5.607±.065	**.102±.012**	**.103±.011**	**.016±.003**	**.088±.013**	**.203±.016**
	Top-Only Training	6.272±.059	6.661±.129	6.465±.208	-	-	-	-	-
	Standard Training	5.285±.095	5.380±.113	5.339±.060	.994±.006	.994±.006	.988±.012	.989±.011	.994±.006
GSM8K	Grad Pruning	10.659±.230	10.520±.089	10.498±.164	.978±.005	.978±.005	.970±.005	.982±.002	.991±.001
	GradSeq-LDP(10240)	10.487±.152	10.621±.052	10.337±.134	.375±.021	.390±.019	.203±.019	.333±.013	.629±.014
	GradSeq-LDP(12288)	10.483±.143	10.608±.062	10.338±.134	.440±.028	.456±.027	.264±.028	.404±.029	.680±.016
	Grad Dropout	107.082±162.999	48.571±61.245	73.177±103.329	.897±.011	.897±.011	.880±.004	.921±.003	.947±.005
	Gradient Mirage(512)	10.430±.159	10.760±.207	10.302±.055	**.031±.009**	**.031±.009**	**.002±.001**	**.042±.006**	**.082±.012**
	Gradient Mirage(1024)	10.142±.169	10.387±.257	10.237±.300	**.075±.012**	**.078±.012**	**.005±.002**	**.071±.011**	**.154±.024**
	Top-Only Training	11.198±.279	11.644±.090	11.295±.172	-	-	-	-	-
	Standard Training	9.714±.138	9.934±.289	9.786±.168	.977±.001	.977±.001	.968±.001	.981±.000	.991±.000

Table 7: *Comparison between Gradient Mirage and existing privacy-preserving methods with Llama2-7B.*

Dataset	Method	Final PPL	PPL@500steps	PPL@1000steps	RougeL-F	Rouge1-F	Rouge2-F	Meteor	TRR
Code-Alpaca	Grad Pruning	5.389±.050	5.305±.027	5.475±.094	.991±.001	.991±.001	.968±.004	.967±.001	.976±.002
	GradSeq-LDP(8192)	5.045±.049	5.319±.125	5.358±.099	.598±.015	.605±.015	.376±.028	.540±.026	.738±.015
	GradSeq-LDP(10240)	5.000±.098	5.327±.100	5.252±.160	.741±.020	.746±.018	.637±.024	.709±.030	.844±.014
	Grad Dropout	6.328±.333	6.371±.575	7.167±1.046	.681±.040	.682±.040	.584±.041	.682±.041	.718±.044
	Gradient Mirage(512)	5.304±.223	5.397±.158	5.404±.028	**.070±.023**	**.071±.024**	**.030±.013**	**.045±.015**	**.163±.030**
	Gradient Mirage(1024)	5.149±.135	5.260±.160	5.292±.024	**.105±.007**	**.107±.008**	**.028±.008**	**.062±.009**	**.204±.011**
	Top-Only Training	5.438±.121	5.801±.042	5.636±.045	-	-	-	-	-
	Standard Training	4.378±.001	4.495±.001	4.418±.001	.998±.000	.998±.000	.984±.001	.977±.001	.983±.001
PIQA	Grad Pruning	14.387±2.094	13.592±.867	14.741±1.939	.942±.016	.944±.015	.882±.026	.928±.024	.931±.019
	GradSeq-LDP(8192)	6.710±.235	6.921±.032	6.798±.105	.819±.020	.822±.019	.659±.039	.763±.027	.831±.005
	GradSeq-LDP(10240)	6.717±.274	6.870±.022	6.749±.073	.845±.010	.848±.009	.694±.008	.712±.019	.827±.014
	Grad Dropout	301.776±415.971	11.223±2.122	118.565±154.226	.844±.013	.854±.024	.773±.017	.820±.025	.848±.008
	Gradient Mirage(512)	6.673±.164	6.917±.162	6.923±.295	**.071±.005**	**.072±.005**	**.008±.005**	**.036±.004**	**.135±.009**
	Gradient Mirage(1024)	6.451±.015	7.089±.081	6.940±.195	**.142±.009**	**.148±.010**	**.021±.008**	**.081±.014**	**.221±.012**
	Top-Only Training	7.444±.130	7.935±.064	7.688±.008	-	-	-	-	-
	Standard Training	5.794±.008	6.022±.002	5.870±.016	.999±.002	.999±.002	.979±.009	.968±.012	.966±.007
GSM8K	Grad Pruning	39.113±35.004	7.430±.183	7.766±.136	.792±.013	.796±.015	.695±.024	.753±.029	.792±.024
	GradSeq-LDP(8192)	12.126±.119	13.091±.384	12.667±.296	.312±.017	.332±.015	.131±.023	.214±.027	.475±.009
	GradSeq-LDP(10240)	12.130±.115	12.981±.326	12.507±.313	.344±.014	.364±.018	.156±.021	.241±.025	.513±.008
	Grad Dropout	16.104±3.461	17.533±3.572	17.125±3.352	.843±.024	.852±.025	.784±.040	.848±.030	.909±.011
	Gradient Mirage(512)	11.989±.105	12.796±.149	12.327±.165	**.028±.002**	**.030±.001**	**.000±.000**	**.012±.001**	**.078±.013**
	Gradient Mirage(1024)	12.037±.294	13.006±.181	12.659±.100	**.074±.005**	**.077±.006**	**.004±.000**	**.032±.003**	**.163±.012**
	Top-Only Training	13.719±.105	14.407±.478	13.873±.390	-	-	-	-	-
	Standard Training	10.007±.002	10.173±.021	10.094±.008	.978±.002	.978±.002	.962±.003	.976±.001	.986±.001

Table 8: *Comparison between Gradient Mirage and existing privacy-preserving methods with DeepSeek-LLM-7B.*

Dataset		CodeAlpaca			PIQA			GSM8K		
Model	Method	RougeL-F	Rouge2-F	Meteor	RougeL-F	Rouge2-F	Meteor	RougeL-F	Rouge2-F	Meteor
Llama3-8B	GP(0.7)	.488±.006	.181±.002	.347±.002	.691±.003	.444±.007	.601±.001	.473±.008	.179±.012	.307±.011
	GD(0.6, 5e-3)	.089±.016	.009±.009	.061±.010	.237±.010	.088±.012	.186±.007	.066±.012	.000±.000	.039±.003
	LDP(10240)	.103±.012	.010±.005	.071±.010	.309±.007	.106±.003	.234±.008	.073±.006	.003±.002	.046±.003
	LDP(12288)	.132±.005	.018±.007	.093±.004	.322±.012	.114±.005	.247±.011	.080±.002	.003±.001	.046±.003
	Standard	.557±.012	.254±.015	.416±.013	.716±.007	.448±.009	.625±.009	.469±.016	.177±.016	.317±.016
	Ours(512)	.094±.005	.002±.001	.068±.003	.121±.017	.013±.002	.082±.004	.103±.004	.003±.001	.066±.005
	Ours(1024)	.161±.011	.008±.002	.106±.003	.246±.019	.040±.009	.156±.012	.136±.015	.012±.002	.105±.010
Llama2-7B	GP(0.7)	.478±.017	.285±.029	.416±.027	.629±.017	.464±.021	.573±.022	.393±.015	.202±.015	.334±.016
	GD(0.6, 5e-3)	.016±.007	.001±.001	.008±.003	.037±.014	.003±.002	.021±.008	.012±.001	.000±.000	.008±.001
	LDP(10240)	.092±.017	.016±.003	.054±.007	.225±.010	.069±.011	.152±.016	.026±.002	.002±.000	.015±.001
	LDP(12288)	.103±.011	.019±.004	.061±.007	.235±.016	.077±.014	.159±.019	.033±.002	.003±.001	.019±.003
	Standard	.471±.008	.287±.012	.414±.016	.617±.022	.444±.021	.570±.012	.420±.004	.226±.005	.371±.008
	Ours(512)	.066±.009	.006±.006	.041±.006	.085±.007	.007±.005	.061±.004	.046±.006	.001±.001	.030±.003
	Ours(1024)	.110±.017	.018±.007	.066±.003	.131±.005	.014±.005	.082±.011	.072±.007	.003±.002	.040±.001
DeepSeek-LLM-7B	GP(0.8)	.381±.018	.192±.014	.289±.022	.549±.021	.347±.024	.446±.023	.361±.009	.175±.007	.249±.008
	GD(0.6, 5e-3)	.002±.001	.000±.000	.002±.001	.001±.001	.000±.000	.002±.001	.003±.001	.000±.000	.002±.000
	LDP(8192)	.033±.002	.003±.001	.016±.001	.180±.011	.051±.011	.109±.005	.020±.003	.001±.000	.009±.002
	LDP(10240)	.032±.002	.003±.001	.015±.001	.187±.010	.057±.007	.113±.005	.025±.005	.000±.000	.011±.002
	Standard	.434±.017	.250±.015	.358±.013	.644±.006	.473±.023	.576±.022	.383±.022	.211±.015	.307±.016
	Ours(512)	.015±.002	.001±.001	.007±.000	.072±.016	.005±.003	.034±.007	.018±.004	.000±.000	.007±.002
	Ours(1024)	.057±.008	.005±.001	.028±.003	.166±.020	.021±.007	.078±.012	.066±.005	.002±.001	.030±.004

Table 9: *Comparison of defense performance against the BiSR(b) attack paradigm using cosine distance as the matching objective across three models.*

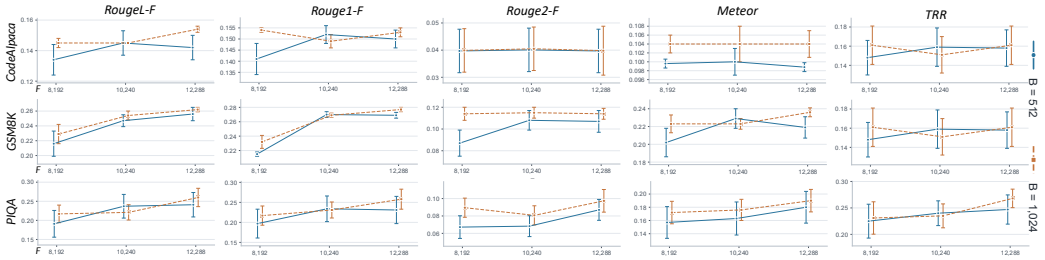


Figure 14: *Forward-pass defense performance of Gradient Mirage combined with a Directional Metric DP-based forward defense under BiSR(f).*

Additionally, we evaluate the effectiveness of combining Gradient Mirage with a Directional Metric DP-based forward defense in mitigating forward-pass leakage. As shown in Fig. 14, Gradient Mirage can be effectively combined with the forward defense to mitigate forward-pass leakage. We adopt BiSR(f) Chen et al. (2024) as the forward attack, using 800 matching epochs and a learning

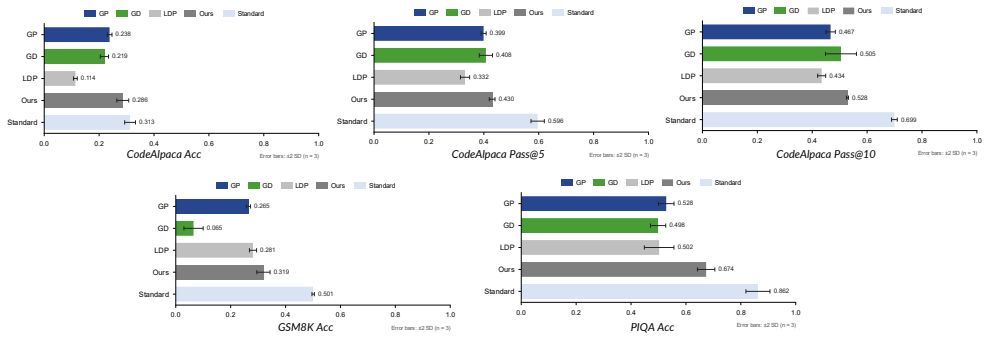


Figure 15: *ask-specific performance under joint forward-and-backward protection, including Acc, Pass@5, and Pass@10 across CodeAlpaca, GSM8K, and PIQA.*

Matching Objective	m						
	100	500	1000	1500	1750	2500	4000
TAG	.382±.026	.246±.018	.211±.012	.195±.016	.202±.002	.170±.002	.168±.010
Cosine Distance	.089±.008	.088±.006	.087±.008	.090±.003	.094±.005	.100±.004	.088±.007
Adaptive TAG	.360±.024	.248±.018	.213±.010	.197±.021	.208±.016	.187±.005	.184±.013

Table 10: *Effect of the mean scale m on reconstruction quality under different matching objectives.*

rate of 5×10^{-4}, and launch the attack every 50 steps over 600 fine-tuning steps. Consistent with the original method, we use cosine distance as the matching objective.

Under the joint forward-and-backward protection setting, we further evaluated task-specific metrics beyond perplexity (PPL) across different datasets, as shown in Fig. 15. Here, Acc denotes accuracy obtained using greedy decoding without sampling, while Pass@5 and Pass@10 are computed with nucleus sampling using (p=0.95) and a temperature of 0.8. For CodeAlpaca, code generation performance is evaluated on the HumanEval benchmark Chen et al. (2021).

D.6 Analysis of Scale Blinding Parameterization

D.6.1 Parameterization

To disentangle the effect of the absolute scaling magnitude from token-wise randomness, we parameterize the blinding coefficient as

$$\alpha_t = m(1 + u_t), u_t \sim \mathrm{Unif}[-r, r], \tag{26}$$

where $m = \mathbb{E}[\alpha_t]$ controls the mean amplitude shift and r controls the relative token-wise variation. Our default choice $\alpha_t \sim \mathrm{Unif}[1500, 2000]$ corresponds to $m = 1750$ and $r = 1/7$, with a coefficient of variation of approximately 0.082.

D.6.2 Scale-Aware Adaptive Attack

We additionally consider an attacker that jointly estimates an optimal nuisance scale during reconstruction. Specifically, the attacker optimizes both the dummy label $\mathbf{y}_{\mathrm{dum}}$ and a learnable raw-scale parameter s_{raw} using the following scale-aware gradient-matching objective: $\min_{y_{\mathrm{dum}},\, s_{\mathrm{raw}}} \|\mathbf{g}_{\mathrm{release}} - s\, \mathbf{g}(y_{\mathrm{dum}})\|_2^2$, $s = \mathrm{softplus}(s_{\mathrm{raw}}) + \epsilon$, where $\mathbf{g}_{\mathrm{release}}$ denotes the released gradient, $\mathbf{g}(y_{\mathrm{dum}})$ denotes the dummy gradient generated from y_{dum}, and $\epsilon_{\mathrm{num}} = 10^{-8}$ is a small constant introduced for numerical stability. The softplus parameterization ensures that the estimated scale remains strictly positive throughout optimization. Both y_{dum} and s_{raw} are jointly updated during reconstruction.

D.6.3 Effect of the Mean Scale m

To study the impact of the mean amplitude shift, we fix the relative token-wise variation as $r = 1/7$ and evaluate different mean scales: $m \in \{100, 500, 1000, 1500, 1750, 2500, 4000\}$. The corre-

r	m						
	100	500	1000	1500	1750	2500	4000
1/5	.341±.021	.212±.014	.191±.012	.187±.012	.183±.011	.143±.013	.133±.014
1/7	.360±.024	.248±.018	.213±.010	.197±.021	.208±.016	.187±.005	.184±.013
1/10	.391±.017	.250±.011	.219±.013	.199±.023	.208±.007	.186±.011	.189±.009
1/25	.397±.024	.258±.017	.221±.023	.237±.015	.219±.010	.194±.010	.193±.010
1/50	.404±.024	.263±.017	.227±.016	.232±.022	.239±.008	.198±.005	.204±.009

Table 11: *Effect of the relative token-wise variation r on reconstruction quality under different mean scales m.*

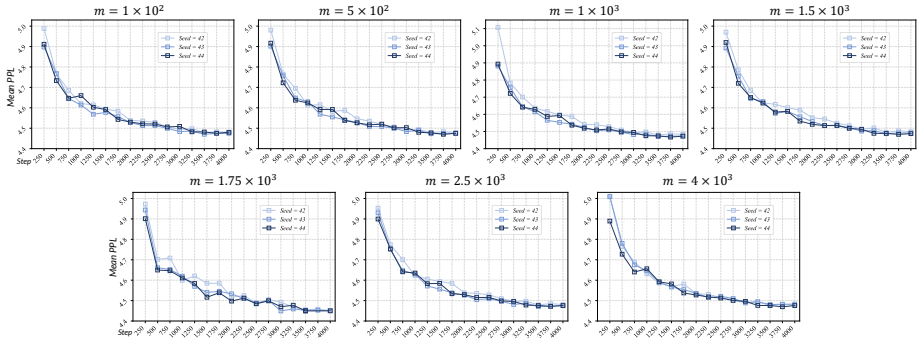


Figure 16: *Fine-tuning performance under different mean scales m with fixed relative token-wise variation $r = 1/7$. The results are obtained on Llama-3-8B with the CodeAlpaca dataset.*

sponding defense effectiveness and fine-tuning performance are reported in Tab. 10 and Fig. 16, respectively. Tab. 10 reports ROUGE-L-F as the reconstruction metric. Throughout the Scale Blinding experiments in this appendix, the directional DP budget is fixed at $\varepsilon = 512$.

As the mean scale increases, the reconstruction quality decreases rapidly in the low-scale regime and gradually stabilizes, while the fine-tuning utility remains largely unaffected across different settings. This indicates that increasing the mean scale effectively disrupts magnitude-sensitive gradient matching, but the privacy gain becomes less significant once the scale reaches a sufficiently large range.

Based on these observations, we choose $m = 1750$ (corresponding to $\alpha_t \sim \mathrm{Unif}[1500, 2000]$) as our default setting. Although this value is not necessarily the globally optimal choice, it provides a favorable privacy-utility trade-off while avoiding unnecessarily large gradient scaling.

D.6.4 Effect of Relative Token-wise Variation r

We further investigate the impact of the relative token-wise variation r, which controls the randomness introduced into the token-level scaling coefficients. We evaluate different values of r under multiple mean scales m, and the reconstruction results are summarized in Tab. 11. The experiments are conducted on the CodeAlpaca dataset using the Llama-3-8B model. ROUGE-L-F is reported as the reconstruction metric.

When r is close to zero, the scaling coefficients become concentrated around a similar value, causing Scale Blinding to degenerate toward deterministic global rescaling. Such a transformation can be partially compensated by an adaptive attacker and provides limited disruption to the relative magnitude pattern among token-level gradients.

As shown in Table 11, introducing token-wise variation consistently improves the effectiveness of Scale Blinding compared with small variation settings. However, the benefit does not increase monotonically with r. Excessively large variation may introduce overly heterogeneous gradient contributions across tokens, which can increase optimization disturbance without providing proportional privacy gains.

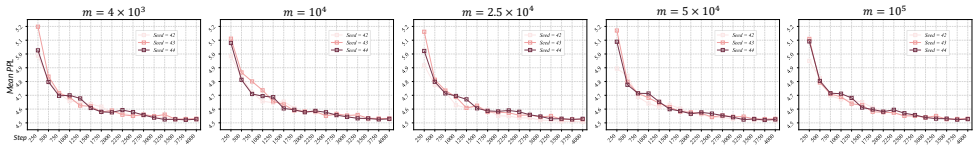


Figure 17: *Fine-tuning performance under extremely large mean scales m with the Trunk-Frozen Training strategy, evaluated on Llama-3-8B with the CodeAlpaca dataset.* The proposed strategy maintains stable convergence across all tested scales and random seeds.

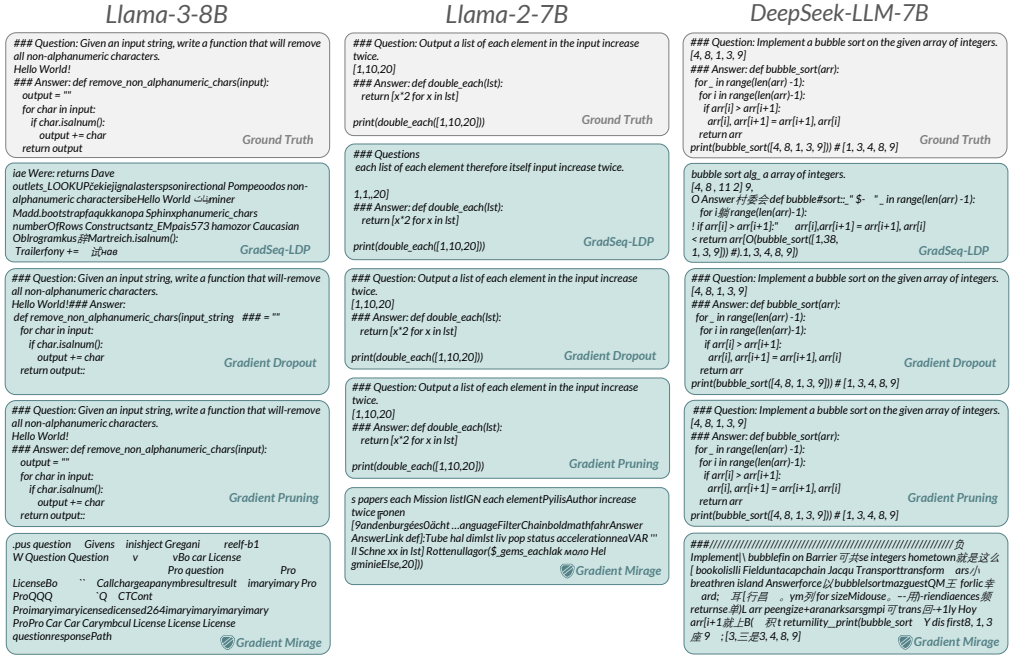


Figure 18: *Qualitative comparison of reconstructed inputs under different privacy-preserving methods across Llama-3-8B, Llama-2-7B, and DeepSeek-LLM-7B.* Compared with existing defenses, Gradient Mirage yields substantially more distorted and incoherent reconstructions.

Considering both reconstruction resistance and optimization stability, we choose $r = 1/7$ as the default setting. This value introduces sufficient token-wise randomness while avoiding excessive perturbation of the gradient structure.

D.6.5 Robust Training under Extreme Scale Blinding

While the default mean scale $m = 1750$ provides a favorable privacy-utility trade-off, we further investigate whether Gradient Mirage remains trainable under extremely large scaling factors. As m becomes excessively large, the amplified gradients introduced by Scale Blinding may cause optimization instability, particularly for the Trunk segment whose gradients are directly affected by the perturbed objective.

To address this issue, we adopt a simple Trunk-Frozen Training strategy: the gradients of the Trunk segment are discarded, while the Top segment continues to be optimized normally and the Bottom segment is updated using the recovered gradients produced by Bottom-Gradient Recovery. Since the recovered Bottom gradients remove the expected amplification introduced by Scale Blinding, they remain relatively stable and informative for optimization.

We evaluate this strategy under significantly larger mean scales $m \in \{4 \times 10^3, 10^4, 2.5 \times 10^4, 5 \times 10^4, 10^5\}$. As shown in Fig. 17, the proposed training strategy maintains stable convergence across all tested scales and consistently outperforms Top-Only Training.

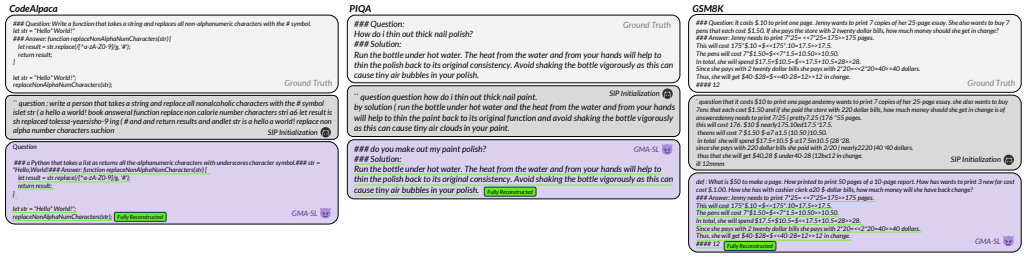


Figure 19: *Qualitative results of GMA-SL under instruction fine-tuning across CodeAlpaca, PIQA, and GSM8K.* Although the query tokens are excluded from supervision during user-side training, GMA-SL successfully reconstructs the supervised answer portion.

E Visualization of Gradient Mirage Performance

Fig. 18 presents representative defense performance across three LLMs. Compared with existing defense methods, Gradient Mirage produces substantially more distorted and incoherent reconstructions, revealing considerably less information about the ground-truth input. These examples visually demonstrate the effectiveness of Gradient Mirage in defending against GMA-SL.

F Limitation and Future Work

F.1 Limitation

The current Scale Blinding design is intentionally simple and importance-agnostic, assigning randomized weights to supervised tokens without considering their relative contribution to model learning. A more refined design could incorporate token-importance estimates to assign larger weights to informative tokens while retaining sufficient randomized rescaling to disrupt the scale information exploited by GMA-SL. Such an importance-aware blinding strategy may further improve the privacy–utility trade-off and is an interesting direction for future work.

F.2 Future Work

Beyond the settings evaluated in our main experiments, we additionally find that GMA-SL remains effective in the instruction fine-tuning setting. Under all attack settings considered for GMA-SL, the labels used in user-side training are constructed by left-shifting the entire sequence by one token, meaning that supervision is applied to all tokens. However, we further find that under instruction fine-tuning, where the query part is not supervised, GMA-SL still poses a substantial threat to the answer part.

GMA-SL matches the interface gradients computed under full-sequence supervision. However, we apply the same attack assumption to the instruction fine-tuning setting, where only the answer tokens are supervised. Let the sequence be decomposed as $x = (x^q, x^a)$, where x^q and x^a denote the query and answer parts, respectively. The optimization objective can be written as $\mathcal{L}_{\text{ans}} = \sum_{t \in \mathcal{I}_a} \ell(f_\theta(x)_t, y_t)$, where $\mathcal{I}_a$ is the set of positions belonging to the answer part. Even under this mismatch, namely, attacking the instruction fine-tuning scenario with the full-supervision assumption, we find that the reconstruction quality for the answer part remains remarkably high, whereas the reconstruction quality for the query part is relatively poor. Representative examples on CodeAlpaca, GSM8K, and PIQA are shown in Fig. 19.

Looking forward, we plan to extend Gradient Mirage to the study of gradient privacy attacks and defenses in multimodal large language models. Several deeper questions also warrant further investigation, including the underlying mechanisms of GMA-SL and, in particular, why it remains effective under instruction fine-tuning despite supervision being restricted to the answer portion. Moreover, unlike supervised learning, reinforcement learning does not rely on explicit token-level labels. In RL-based SL, it remains an open question how to formally define the privacy leakage of reward models, as well as how an adversary could effectively extract private information of reward models.