ALIASGUARD: SURROGATE-BASED DEFENSE AGAINST PASSIVE INFERENCE IN SPLIT LANGUAGE MODEL FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Split learning (SL) is often treated as a privacy-preserving way to fine-tune language models (LMs) without exposing clients' private training tokens. In split-based LM fine-tuning (SL-FT), however, an honest-but-curious (passive) server can leverage smashed data in the forward pass and gradients in the backward pass to infer these private tokens in training sequences. Under the passive inference, current defenses face a trade-off: they cannot protect these private tokens while retaining training utility. To mitigate this trade-off, we present AliasGuard, a surrogate-based defense. Specifically, AliasGuard generates surrogates from the unprotected context of a sequence and uses these surrogates to construct the training sequences and one-hot vectors for computing smashed data and gradients. We prove that AliasGuard's smashed data and gradients provide no additional information about private tokens beyond the unprotected context. We also bound changes in smashed data at unprotected-token positions. We evaluate AliasGuard under three corpora and three models. Under adaptive inference attacks, AliasGuard reduces private-token recovery by 67.4–99.6% relative to the lowest baseline rate, while retaining much of No Defense baseline's task utility.

1 INTRODUCTION

Fine-tuning (FT) adapts a pre-trained language model (LM) to a task using training sequences (Howard & Ruder, 2018; Hu et al., 2022). These sequences may contain private information, so sending them to a central server creates privacy risks (Yu et al., 2022). To mitigate the risks, federated learning (FL) (McMahan et al., 2017) and split learning (SL) (Vepakomma et al., 2018) are widely investigated. In particular, SL-based LM fine-tuning (SL-FT) places the first and last layers (the head and tail) of the pre-trained model on the client and the middle layers (the trunk) of the model on the server (Liu et al., 2025). The client sends only the head output (denoted as smashed data) and the gradient to the server during training. Although SL-FT does not share these sequences, it is unable to prevent passive inference attacks (Chen et al., 2024), where an honest-but-curious server uses the messages it generates and receives to infer the client's private information in training sequences (Zhu et al., 2025). Two properties of SL-FT help explain why these attacks are effective.

The first is the not-too-far property (Chen et al., 2024). Let X denote one client training sequence and Z the smashed data produced by the fine-tuned head (Figure 1(a)). Although the server does not have the fine-tuned head, it has the public head from the pre-trained model. The server uses public auxiliary sequence X_{aux}, drawn from a distribution similar to that of X, and the public head to produce the corresponding Z_{aux} (Figure 1(b)). It then trains a decoder D_{dec} to recover X_{aux} from Z_{aux} (Figure 1(c)). Since the fine-tuned head is distributionally close to the public head, attacks leverage this property to combine strategies like optimization with D_{dec} to infer X from Z.

The second is the autoregressive property. A causal LM predicts the next token, so the label at position i is $y_i = x_{i+1}$ (Radford et al., 2019). Let P_i be the predicted vocabulary distribution vector and Y_i the one-hot vector of y_i. For cross-entropy loss at this position, the gradient with respect to the scores before softmax is $P_i - Y_i$. Subtracting Y_i reduces only the true token's entry by one (see Figure 2(a)). Changing the label y_i therefore changes which entry in $P_i - Y_i$ is reduced. The tail uses $P_i - Y_i$ for every position i to compute the gradient Δ sent to the server (see Figure 2(b)).

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.

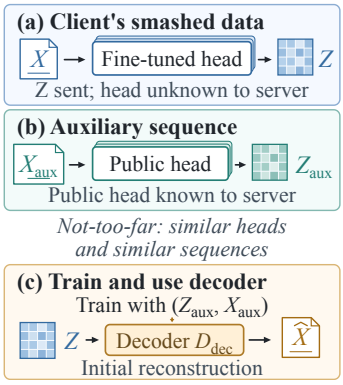

Figure 1: Not-too-far attack.

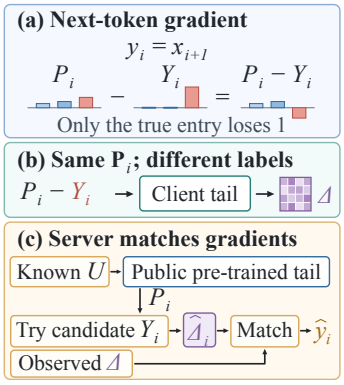

Figure 2: Backward attack.

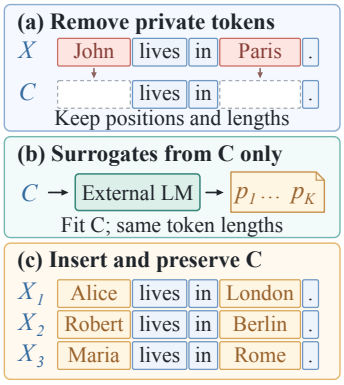

Figure 3: Surrogate & context.

Different labels with the same P_i can thus produce different gradients. The server feeds its known trunk output U into the public pre-trained tail and computes gradients for the guessed distribution of $P_i - Y_i$. It adjusts these guessed distributions to reduce the difference between its produced gradients $\hat{\Delta}$ and the observed Δ, then selects each position's most probable token, as illustrated in Figure 2(c) (Zhu et al., 2019; Chen et al., 2024). Together, using Z and Δ gives the server two sources of information to infer the private tokens in X. Recent attacks have exploited the combined observations in practice, making passive inference attacks particularly effective (Chen et al., 2024).

The central difficulty is that Z and Δ_i serve two roles: they can reveal private tokens in X, but they also carry information needed for training utility. A solution is to use secure-computation methods like multi-party computation but at the cost in training efficiency (Luo et al., 2026). To maintain training efficiency, perturbation methods add noise to smashed data or gradients, but stronger noise can compromise training utility (Li et al., 2022a; Du et al., 2023; Shen et al., 2025). To keep training utility, other defenses change the client's head and tail model to make the exchanged signals, i.e., smashed data and gradients, harder to be leveraged for inference (Liu et al., 2025). However, these defenses still keep private information in smashed data and gradients for training utility, so adaptive attacks that devise attack strategies based on corresponding defense designs can still leverage the exchanged signals for inference. The open question is how to protect private tokens under adaptive attacks while retaining training utility in a training-efficient manner.

This work proposes AliasGuard to fill this gap. The design uses surrogates and the unprotected context C to construct both smashed data and gradients. We first identify the private tokens and remove them from their sequence X, which produces an unprotected context C, as illustrated in Figure 3(a). We then build a group of surrogates for X using only C. This design ensures that, given C, these surrogates are independent of the private tokens in X (see Figure 3(b)). We generate a set of sequences for X by inserting each surrogate at the private-token positions, as illustrated in Figure 3(c). Each sequence (e.g., X_1, X_2 and X_3) preserves every unprotected token (e.g., lives in) at its original position. Using these sequences, AliasGuard makes the following contributions:

- **AliasGuard rebuilds both smashed data and one-hot vectors.** It computes smashed data for each selected surrogate sequence of X and averages the results before sending them to the server (Figure 4(a)). It averages the same selected sequences' one-hot vectors to replace those of X, preserving every unprotected next token's one-hot vector (Figure 4(b)).
- **AliasGuard provides forward and backward privacy guarantees.** We prove that smashed data and gradients in AliasGuard provide no additional information about private tokens beyond the training sequences' unprotected contexts. We bound changes in smashed data at unprotected-token positions and prove that the one-hot vectors of unprotected next tokens remain unchanged.
- **We evaluate AliasGuard under adaptive attacks.** Across three corpora and GPT-2 Small, Qwen2.5-1.5B/3B, we compare four baselines. AliasGuard reduces private-token recovery under combined forward and backward reconstruction by 67.4–99.6% relative to the lowest baseline rate, has the highest task score in three of nine settings, and adds 11.4–23.6 ms per update over No Defense (NoDef) baseline using three prepared surrogate sequences.

2 BACKGROUND

2.1 SPLIT-BASED LANGUAGE MODEL FINE-TUNING

Split learning (SL) partitions a language model (LM) into a client-side head f_ϕ, a server-side trunk g_ψ, and a client-side tail h_ω, with parameters ϕ, ψ, and ω, respectively (Chen et al., 2024; Liu et al., 2025). For a sequence $X = (x_1, \ldots, x_I)$ of I vocabulary tokens, the forward pass computes

$$Z = f_\phi(X), \qquad U = g_\psi(Z), \qquad P_i = \mathrm{softmax}\big(h_\omega(U)_i\big). \tag{1}$$

Here, $Z \in \mathbb{R}^{I \times d_h}$ is the smashed data, and $U \in \mathbb{R}^{I \times d_u}$ is the trunk output. The vector $P_i \in \mathbb{R}^V$ contains predicted probabilities for V vocabulary tokens. During each round's forward pass, the client sends Z to the server and receives U from the server.

Causal LM uses the next token $y_i = x_{i+1}$ as the label at position i. Let $Y_i = \mathrm{onehot}(y_i)$ be its one-hot label vector. Let U_i denote row i of U. For $I \geq 2$, the backward pass computes

$$\begin{gathered} \mathcal{L} = -\frac{1}{I-1} \sum_{i=1}^{I-1} Y_i^\top \log P_i, \qquad \Delta_i = \frac{\partial \mathcal{L}}{\partial U_i}, \\ G_Z = \left(\frac{\partial U}{\partial Z}\right)^\top \Delta, \quad \nabla_\psi \mathcal{L} = \left(\frac{\partial U}{\partial \psi}\right)^\top \Delta, \quad \nabla_\phi \mathcal{L} = \left(\frac{\partial Z}{\partial \phi}\right)^\top G_Z. \end{gathered} \tag{2}$$

Here, $\Delta = (\Delta_1, \ldots, \Delta_I)$ and $G_Z = \partial \mathcal{L} / \partial Z$; matrix entries are arranged as vectors in these derivatives. The client uses $\nabla_\omega \mathcal{L}$ to update the tail and sends Δ. The server uses $\nabla_\psi \mathcal{L}$ to update the trunk and returns G_Z. The client uses $\nabla_\phi \mathcal{L}$ to update the head.

2.2 THREAT MODEL, PRIVACY OBJECTIVE AND CONCEPTS

Let $\mathcal{X}$ denote the client's training dataset. We consider an honest-but-curious (passive) server that follows the training workflow described above and knows the defense, the public pre-trained model, and an auxiliary dataset $\mathcal{X}'$ drawn from a distribution similar with $\mathcal{X}$. It can adapt its inference strategy using the exchanged signals (Z and Δ) and its own model states across training rounds.

We divide the tokens in each sequence X into private tokens and unprotected tokens. Let T_p and T_u be their respective position sets in X. Private tokens are the tokens selected for protection, such as person names or code identifiers. Privacy means that the exchanged signals provide no additional information about private tokens beyond C. We further use *training utility* to mean the ability of LM to learn the intended task and retain task performance after fine-tuning.

2.3 RELATED WORK

Inference attacks. Active SL attacks manipulate training (Pasquini et al., 2021); passive attacks infer labels (Li et al., 2022a; Liu et al., 2024) or client inputs (Zhu et al., 2019; Geiping et al., 2020), such as text (Deng et al., 2021; Balunović et al., 2022) or images (Zhu et al., 2025). Others support embedding-based (Morris et al., 2023) and LM outputs-based inversion (Carlini et al., 2021).

Reducing private information. For smashed data, NoPeek reduces dependence on inputs (Vepakomma et al., 2020), while DP-Forward (Du et al., 2023), SAP (Shen et al., 2025), and PrivTune (Liu et al., 2026a) add noise. For gradients, Soteria prunes entries in FL (Sun et al., 2021), and Marvell adds noise in SL (Li et al., 2022a). DP-SGD clips and perturbs per-sequence parameter gradients (Abadi et al., 2016); PE-SGD uses public data to guide gradient directions (Zou et al., 2026). These changes can remove task information along with private information, creating a privacy–utility trade-off (Yu et al., 2022; Li et al., 2022b; Rehn et al., 2026).

Making inference harder. DualGuard changes client head and tail parameters (Liu et al., 2025). MIXGUARD (Chen et al., 2026) inserts extra tokens, mixes smashed data, and perturbs gradients. Gradient Mirage (Miao et al., 2026) selects token labels and changes gradient length and direction. Adaptive reconstruction can exploit private information that remains in smashed data and gradients (Chen et al., 2024). Our experiments demonstrate this limitation for DualGuard (Section 4.2).

Secure computation. Secure multi-party computation (MPC) allows training to use private inputs without revealing them, but requires extra computation and communication (Luo et al., 2026).

AliasGuard constructs smashed data and averages of one-hot vectors from surrogates generated only from C. It aims to mitigate the privacy–utility trade-off under adaptive attacks while keeping training time close to NoDef.

3 ALIASGUARD

Before elaborating AliasGuard, we first present surrogates. As shown in Figure 3(a), the unprotected context C contains all unprotected tokens of its sequence X in their original order. It also records the positions and token length of each private datum item in X. Let $\mathcal{D} = \{d_1, d_2, \ldots\}$ be the set of private datum items in X, such as John and Paris (see Figure 3(a)). A private datum item may contain multiple private tokens. AliasGuard uses an LM to generate surrogates for the private datum items in X using only the unprotected context C (Figure 3(b)). Each surrogate contains one replacement for every private datum item in $\mathcal{D}$, and each replacement has the same token length as the private datum item it replaces. Appendix H.5 describes how the surrogates are generated, checked, and selected.

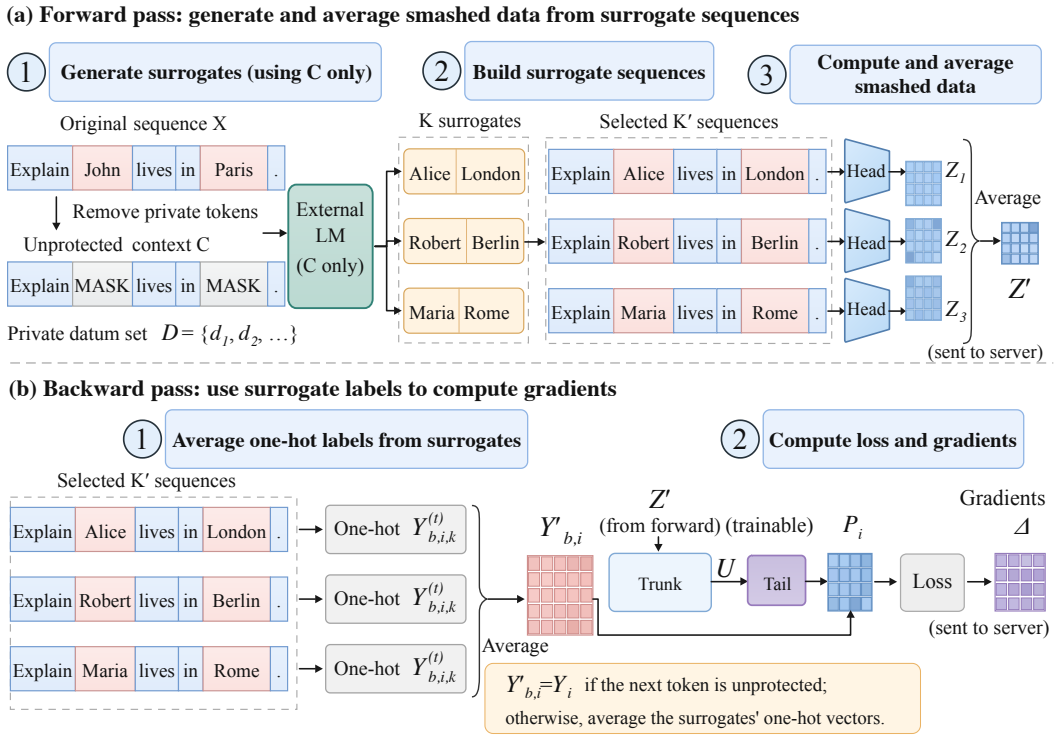

Figure 4: AliasGuard's forward and backward passes.

3.1 ALIASGUARD'S FORWARD PASS

Figure 4 summarizes AliasGuard's forward and backward passes. In the forward pass, AliasGuard rebuilds smashed data Z' and has two goals. For the first goal, observing Z' should leave the server's inference probabilities for private datum d unchanged after it knows C:

$$\Pr(d \mid C, Z') = \Pr(d \mid C). \tag{3}$$

For the second goal, we seek an upper bound on how much the values in Z' differ from those in Z at the positions of unprotected tokens in X.

Because $Z = f_\phi(X)$ is computed from a sequence that contains d, the smashed data Z can reveal information about d. AliasGuard removes the private datum items from X. Specifically, at round $t = 1, 2, \ldots$, let $X_b^{(t)}$ be sequence $b \in \{1, \ldots, B\}$ in a batch of size B, with unprotected context $C_b^{(t)}$. The client performs the following three operations.

First, for $X_b^{(t)}$, the client uses only $C_b^{(t)}$ to generate K surrogates and randomly selects K' of them, where $1 \le K' \le K$. The private tokens in $X_b^{(t)}$ are not used to generate or select the surrogates.

For each $k \in \{1, \dots, K'\}$ and the k-th selected surrogate, let $p_{b,k}^{(t)}$ list the tokens of the k-th selected surrogate. The client performs the second operation:

$$X_{b,k}^{(t)} = X_b^{(t)}[T_{p,b}^{(t)} \leftarrow p_{b,k}^{(t)}], \tag{4}$$

where $T_{p,b}^{(t)}$ contains the private-token positions in $X_b^{(t)}$. This substitution preserves every unprotected token and replaces the private tokens in $X_b^{(t)}$ with the selected surrogates.

For the third operation, the client passes these K' sequences through the head at round t, with parameters $\phi^{(t)}$, and averages their smashed data:

$$Z_{b,k}^{(t)} = f_{\phi^{(t)}}(X_{b,k}^{(t)}), \qquad Z_b'^{(t)} = \frac{1}{K'} \sum_{k=1}^{K'} Z_{b,k}^{(t)}. \tag{5}$$

Theorem 1 (Appendix A) proves Eq. (3) for each sequence.

Although unprotected tokens in $X_b^{(t)}$ remain unchanged, replacing private tokens with surrogates can affect their head outputs through attention (Vaswani et al., 2017). For the second goal, we compare each $Z_{b,k}^{(t)}$ with $Z_b^{(t)} = f_{\phi^{(t)}}(X_b^{(t)})$ at the unprotected token positions $T_{u,b}^{(t)}$ of $X_b^{(t)}$. We measure this difference using the Frobenius norm $\|\cdot\|_F$. Theorem 2 in Appendix A.2 proves that if each difference has norm at most $\varepsilon \ge 0$, the difference after averaging also has norm at most ε.

3.2 AliasGuard's backward pass

Section 3.1 achieves the two goals at the smashed data side, while AliasGuard also has two goals at the backward side. First, at a fixed model state and training round, the gradient Δ should offer no further evidence about a private datum d:

$$\Pr(d \mid C, Z', \Delta) = \Pr(d \mid C, Z') = \Pr(d \mid C). \tag{6}$$

Second, it aims to retain training utility during the backward pass.

Section 2.1 defines the label vector $Y_i = \mathrm{onehot}(y_i)$, where $y_i = x_{i+1}$. When x_{i+1} is private, Y_i identifies that token. Thus, Y_i can affect the gradients Δ through the logit gradient $P_i - Y_i$. To achieve the first goal, Y_i should not be determined by its corresponding private token. To this end, for the original sequence $X_b^{(t)}$, the client uses its K' selected sequences as the forward pass. For the $(i+1)$-th token in the k-th sequence $X_{b,k}^{(t)}$, denoted as $(X_{b,k}^{(t)})_{i+1}$, the client averages these vectors:

$$Y_{b,i}'^{(t)} = \frac{1}{K'} \sum_{k=1}^{K'} Y_{b,i,k}^{(t)}. \tag{7}$$

where $Y_{b,i,k}^{(t)} = \mathrm{onehot}\big((X_{b,k}^{(t)})_{i+1}\big), 1 \le i < I$. AliasGuard uses $Y_{b,i}'^{(t)}$ to replace the original Y_i. If the next token at position $i+1$ is unprotected, $Y_{b,i}'^{(t)} = Y_i$. If the next token is private, $Y_{b,i}'^{(t)}$ is the average of the one-hot vectors of the selected surrogate tokens at position $i+1$. We call these averaged vectors surrogate labels. Thus, the surrogate labels depend only on C and corresponding LM and prompts, independent of the removed private tokens.

AliasGuard uses $Y_{b,i}'^{(t)}$ to compute each sequence's loss $\mathcal{L}_{\mathrm{AG},b}^{(t)}$. The batch loss is $\mathcal{L}_{\mathrm{AG}}^{(t)} = B^{-1} \sum_{b=1}^{B} \mathcal{L}_{\mathrm{AG},b}^{(t)}$. The loss and gradient are:

$$\mathcal{L}_{\mathrm{AG},b}^{(t)} = -\frac{1}{I-1} \sum_{i=1}^{I-1} (Y_{b,i}'^{(t)})^\top \log P_{b,i}^{(t)}, \quad \Delta_{b,i}^{(t)} = \frac{\partial \mathcal{L}_{\mathrm{AG}}^{(t)}}{\partial U_{b,i}^{(t)}} = \frac{1}{B} \frac{\partial \mathcal{L}_{\mathrm{AG},b}^{(t)}}{\partial U_{b,i}^{(t)}}. \tag{8}$$

where B is the batch size and $P_{b,i}^{(t)}$ is from Eq. (1). The head, trunk, and tail are updated using this loss and its gradients. Theorem 3 (Appendix A.3) proves that if two training sequences use the same

unprotected contexts C, then changing only the private tokens in the two sequences does not change the exchanged signals observed by the server. This achieves the first goal of backward pass.

At a position whose next token is unprotected, every selected surrogate sequence keeps that next token unchanged. Their one-hot vectors are therefore identical, and averaging them gives the original one-hot vector. Theorem 4 (Appendix A.4) proves this property, supporting the second goal.

4 Experimental Evaluation

We ask five questions: **RQ1**, how does AliasGuard compare with other state-of-the-art methods in privacy and utility? **RQ2**, why does AliasGuard reduce private-token recovery, and why can recovery remain nonzero? **RQ3**, when does protecting private tokens affect task utility? **RQ4**, how do surrogate sequences and surrogate labels each reduce private-token recovery? **RQ5**, how does K' affect privacy, utility, and training time?

4.1 Setup

We evaluate CodeAlpaca-H4 (Chaudhary, 2023), DialogSum (Chen et al., 2021), and GSM8K (Cobbe et al., 2021) with GPT-2 small (Radford et al., 2019) and Qwen2.5-1.5B/3B (Yang et al., 2024). We compare No Defense (NoDef), DP-forward (Du et al., 2023), DP-SGD*, DualGuard (Liu et al., 2025), and AliasGuard. DP-forward adds noise to smashed data; DP-SGD* adds noise to the exchanged gradient. At each random seed, methods train on one sequence per update: 154 updates for GPT-2 and 169 for Qwen. We evaluate the model after its last update. For privacy, H4 protects code identifiers; the other tasks protect person names. Recovery is the percentage of private-token positions guessed correctly in 128 training sequences. We use adaptive BiSR (Chen et al., 2024): F uses smashed data and the trunk's output, B uses gradients from the backward pass, and BF uses both (Appendix B.3). A BF procedure can sometimes guess fewer private tokens correctly than F or B; Appendix I explains why. For training utility, H4 lexical F1 penalizes missing and extra words relative to reference code. DialogSum ROUGE-L measures matching word sequences in generated and reference summaries (Lin, 2004). GSM8K reports the percentage of correct answers. Appendix B gives the full evaluation settings.

4.2 Privacy Protection and Training Utility (RQ1)

Table 1(a) compares nine dataset–model settings. AliasGuard's BF recovery is 0.35–30.68%, versus 83.30–100% for the four baselines (baseline recovery is 3.07–282 times AliasGuard's; Appendix C.1). Its task score exceeds DP-forward's and DualGuard's in all nine settings, and exceeds DP-SGD*'s and NoDef's in four each.

Table 1(b) compares repeated training. Across three GSM8K seeds, AliasGuard lowers BF recovery in all nine DualGuard comparisons and raises task scores in all six Qwen comparisons. Appendix D gives the additional training-seed results.

Table 1(c) reports the highest recovery in four additional tests. AliasGuard remains 51.11–84.85 percentage points below the lower NoDef/DualGuard rate. Its recovery advantage persists in these four tests. Appendix C lists the other results; Appendix E reports tests under other settings.

4.3 Why Does AliasGuard Reduce Private-Token Recovery? (RQ2)

We further evaluate whether private tokens can change what the server can see when C and other settings are fixed. The idea is to construct three sequences; they have the same unprotected context C but different private tokens. In four dataset–model settings, we test 64 groups. Each group contains an original sequence and two alternatives with different private tokens but the same C. With all other inputs and random choices fixed, AliasGuard uses the same selected surrogates and labels for all three sequences in the same group. A separate check replaced private tokens 1,206 times and found the same surrogate construction inputs and outputs.

The 128 pairs show that changing only private tokens leaves AliasGuard's exchanged Z' and Δ unchanged (Figure 5(a)). This matches Theorem 1; Theorem 3 extends the result through training under its stated conditions. An adaptive reconstruction procedure therefore sees the same C, Z', and

Table 1: (a) Five methods at seed 3407; task scores are H4 lexical F1, DialogSum (DS) ROUGE-L, and GSM8K (G8) answer accuracy. (b) Three GSM8K seeds: mean ± sample standard deviation, using observations from each trained model (Appendix D.2). (c) Highest measured recovery in four settings. Reductions compare AliasGuard with the lower NoDef/DualGuard rate, in percentage points (Appendix E.4). DialogSum DP-forward uses the higher of two BF rates (Appendix B.3).

	H4			DS			G8		
Method	GPT-2	Q1.5B	Q3B	GPT-2	Q1.5B	Q3B	GPT-2	Q1.5B	Q3B
BF private-token recovery under adaptive BiSR (%; lower is better)									
NoDef	99.92	96.96	98.43	100.00	94.16	95.89	99.47	83.30	86.76
DP-forward†	94.11	99.25	96.19	96.93	100.00	100.00	89.93	100.00	100.00
DP-SGD*†	99.54	100.00	100.00	99.59	100.00	100.00	99.65	100.00	100.00
DualGuard	100.00	97.28	98.59	100.00	93.94	95.02	99.65	84.32	86.76
AliasGuard	**30.68**	**22.28**	**19.30**	**3.28**	**4.98**	**3.68**	**0.35**	**2.04**	**3.05**
Task score (%; higher is better)									
NoDef	17.99	**49.79**	48.05	34.45	41.19	**42.51**	1.74	**36.62**	**51.33**
DP-forward†	1.17	19.52	23.80	5.51	0.03	23.67	0.00	1.29	3.18
DP-SGD*†	**19.30**	44.74	**48.80**	33.73	41.22	42.45	2.12	34.72	42.61
DualGuard	12.61	26.02	27.00	10.80	19.31	11.12	1.52	27.29	18.12
AliasGuard	18.73	46.69	45.55	**35.00**	**41.57**	41.82	**2.27**	31.31	40.56

(b) GSM8K across three training seeds

Model	Method	Task (%) ↑	BF (%) ↓
GPT-2	DualGuard	**1.44 ± 0.08**	100.00 ± 0.00
	AliasGuard	**1.44 ± 0.40**	**1.06 ± 0.18**
Q1.5B	DualGuard	26.79 ± 3.34	81.81 ± 0.12
	AliasGuard	**38.26 ± 2.81**	**0.81 ± 0.00**
Q3B	DualGuard	15.74 ± 1.82	90.22 ± 0.20
	AliasGuard	**36.69 ± 3.93**	**0.81 ± 0.00**

(c) Highest evaluated recovery

Setting	AliasGuard (%)	Lower baseline (%)	Reduction (points)
H4/Q1.5B	40.35	96.60	56.26
DS/Q1.5B	6.28	91.13	84.85
G8/Q1.5B	5.09	80.65	75.56
H4/GPT-2	48.89	100.00	51.11

†: DP reconstruction receives the unprotected tokens.

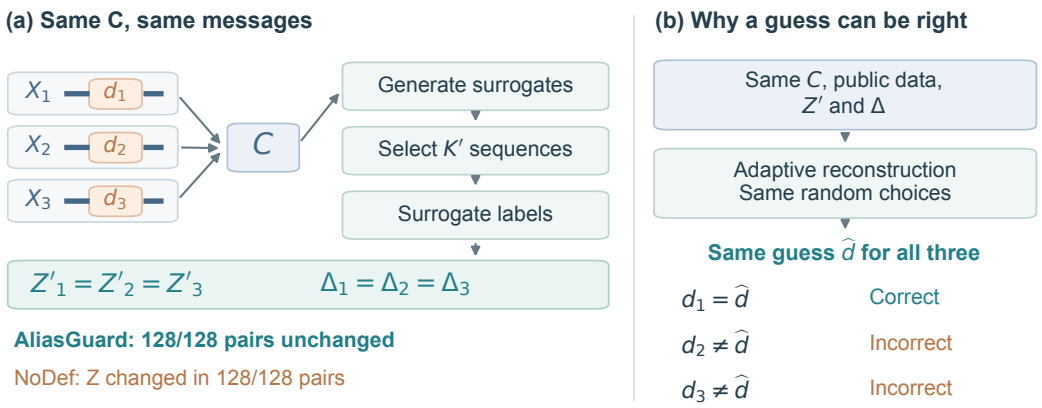

Figure 5: Fixed-context test. (a) Each pair shares C but has different private tokens. AliasGuard's Z' and Δ match exactly in all 128 pairs; NoDef's Z changes in all 128. (b) The same C, Z', and Δ lead to the same guess when the reconstruction procedure's random choices are fixed. C can still favor that guess.

Δ for each sequence in the same group. With fixed random choices, it makes the same guess. That guess may be right for one sequence and wrong for another because C and public data can favor it (Figure 5(b)). Appendix F reports the construction checks. As a result, the non-zero recovery is not because smashed data and gradients contain private information, but guessing based on C.

4.4 When Does Privacy Protection Affect Training Utility? (RQ3)

Theorem 2 bounds changes in smashed data at unprotected token positions; Theorem 4 keeps their next-token labels unchanged. We then test how these designs influence training utility. For the same sequence, we score it twice, first on all answer tokens, then only on unprotected answer tokens.

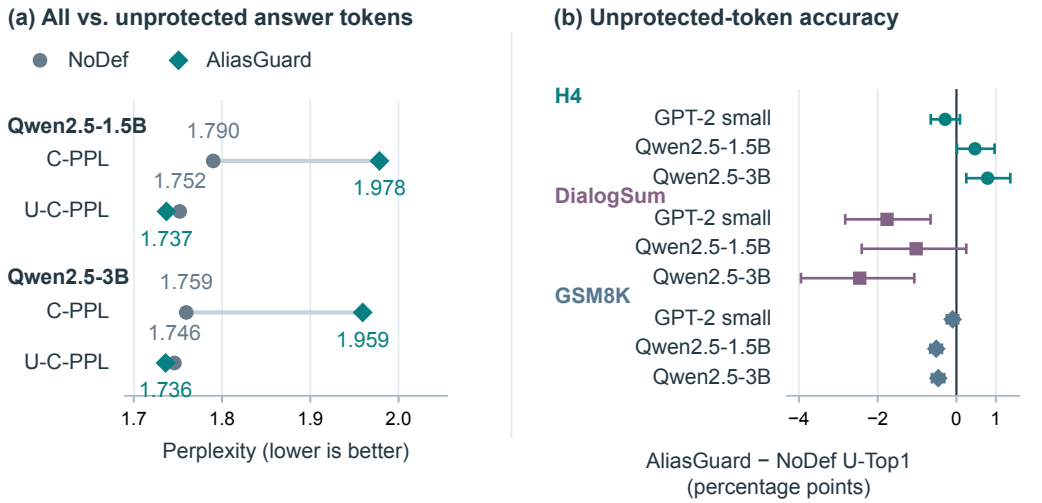

Figure 6: Token prediction. (a) C-PPL is the exponential of the mean negative log probability of correct answer tokens, including private tokens (Bengio et al., 2003); U-C-PPL excludes private-token positions. Lower is better. Both use the same 150 H4 sequences for each Qwen model. (b) U-Top1 is the percentage of unprotected tokens whose most likely next token is correct; higher is better. Dots show AliasGuard minus NoDef; lines show 95% intervals from resampling the same test sequences for both methods (Dror et al., 2018). Each prediction uses the earlier correct answer tokens (Bengio et al., 2015).

For H4/Qwen2.5-1.5B, NoDef/AliasGuard C-PPL is 1.790/1.978, whereas U-C-PPL is 1.752/1.737. For Qwen2.5-3B, 1.759/1.959 becomes 1.746/1.736 (Figure 6(a)). U-Top1 also increases by 0.47 and 0.79 percentage points (Figure 6(b)). Thus, AliasGuard's private-token predictions are worse than those of NoDef in these H4 settings, while prediction at unprotected positions improves.

Across all nine settings, the AliasGuard–NoDef U-Top1 difference stays between -2.46 and $+0.79$ points. For other datasets beyond H4, AliasGuard improves DialogSum ROUGE-L by 0.55/0.38 points with GPT-2/Qwen2.5-1.5B and GSM8K accuracy by 0.53 points with GPT-2 (Table 1). Across three training seeds, U-Top1 stays within 0.77 points of NoDef in all six Qwen/GSM8K pairs (Appendix D.3). These results attest that by keeping unprotected tokens intact, training utility can be retained. Appendix G shows more evaluation details with examples (Mirzadeh et al., 2025).

Insight. In the tested settings, AliasGuard predicts unprotected tokens nearly as accurately as NoDef; some task scores also improve.

4.5 Separate Effects of Surrogate-Sequences and Labels (RQ4)

Here we answer why surrogate-sequences and their labels are indispensable. We use either the original sequence or its surrogate-sequences for the forward pass, and either the original label or surrogate-sequences' labels for the loss. This gives four combinations per dataset (Table 2). Surrogate sequences lower F recovery from 78.78% to 20.85% on H4, 35.87% to 1.79% on GSM8K, and 49.58% to 3.39% on DialogSum. With Z' fixed, surrogate labels reduce B recovery by 14.02 percentage points on H4, 2.24 on GSM8K, and 7.20 on DialogSum. Surrogate-sequences therefore reduce recovery from the forward pass; surrogate labels reduce recovery from gradients. Using both gives the lowest rate when we take the highest of F, B, and BF on each tested dataset. Appendix H.7 reports all four combinations and the evaluation details.

Table 2: Highest F, B, or BF private-token recovery (%); lower is better. Z and Y come from the original sequence; Z' and Y' come from surrogate sequences and their labels.

Dataset	(Z, Y)	(Z', Y)	(Z, Y')	(Z', Y')
H4	88.56	36.90	78.78	**22.88**
GSM8K	41.70	4.93	35.87	**2.69**
DialogSum	57.20	13.14	49.58	**5.93**

4.6 Effect of the Number of Selected Surrogates (RQ5)

AliasGuard introduces K'. To test its influence on privacy, training utility and training efficiency, we compare $K' = 1, 3, 5$ with training sequences and other AliasGuard settings fixed. Figure 7(a) and Figure 7(b) show that increasing K' may improve privacy and training utility in some cases. For example, on GSM8K/Qwen2.5-1.5B, increasing K' from 1 to 3 to 5 raises accuracy from 32.68% to 35.56% to 36.32%, with BF recovery of 0.41%, 2.24%, and 4.68%. However, most cases do not have this phenomenon. Thus, increasing K' is not necessary for AliasGuard.

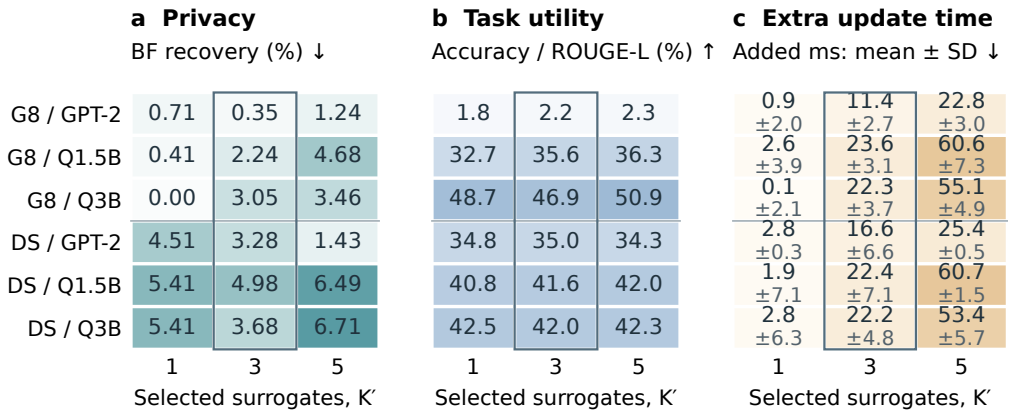

Figure 7: Effect of K'. (a) and (b) present the privacy and training utility results. (c) Extra time per update relative to NoDef, in milliseconds; lower is better. Values show the mean and standard deviation across five repeats of 50 updates. Outlines mark $K' = 3$; timing details are in Appendix H.4.

Slightly increasing K' does not add training time too much, adding around 0.1-60.7 milliseconds (ms) (Figure 7(c)). Most added training time comes from the head, which processes the selected surrogate sequences and averages their smashed data into one Z' for the trunk. Since increasing K' does not necessarily improve privacy and training utility, setting $K' = 1$ can keep AliasGuard's update time nearly the same as NoDef. Additionally, preparing 30 surrogates per training sequence takes an estimated 18–21 minutes per training set (Appendix H.4), but surrogate generation is fully offline and can be performed before online SL-FT. Appendix H shows more details about these tests.

5 Conclusion

AliasGuard is a surrogate-based defense for SL-FT. Its smashed data and gradients provide no additional information about private tokens beyond the training sequences' unprotected contexts. It preserves unprotected tokens and their next-token one-hot vectors for training utility.

Across nine settings, AliasGuard's BF private-token recovery is at most one-third of the best baseline's, with the highest task score in three. Across three training seeds, unprotected-token accuracy stays within 0.77 percentage points of NoDef in all six Qwen/GSM8K comparisons. Some task scores also exceed NoDef; how to explain these gains requires further study.

When a task requires private tokens, protecting them can conflict with learning to reproduce them. Future work should examine privacy and training utility in such tasks.

Ethics Considerations

This work evaluates a privacy method for split-based LM fine-tuning. The reconstruction procedures are applied only to public benchmarks and to training records produced for this study; no third-party service or private user data is targeted. The artifact separates public auxiliary fitting from sealed target scoring and reports aggregate reconstruction results.

The experiments use public text corpora and benchmark datasets rather than newly collected human-subject data. We do not interact with users, deploy the system against third-party services, scrape private platforms, or attempt to recover real user secrets. The private data in the experiments are controlled protected data items used for measurement; they are not presented as real-world personally identifiable information. Because the study does not involve live systems or undisclosed vulnerabilities, we do not identify a separate vulnerability-disclosure obligation. If AliasGuard is later applied to proprietary, medical, financial, or otherwise regulated data, the deployment should undergo the relevant institutional or legal review, document the data-handling basis, and verify that public surrogate sources do not introduce harmful bias or unauthorized disclosure. For an accepted version, we intend to release reproducibility artifacts subject to dataset and model licenses.

AI Use Statement

The authors developed the core method and experimental design. Generative AI tools assisted with language polishing, literature retrieval and reference checking, and research execution, including coding, debugging, testing, and analysis of experimental results. The authors critically reviewed the suggestions and outputs, corrected errors, and conducted their own analyses before making final research and writing decisions. The authors take responsibility for the final manuscript, code, and reported results.

Reproducibility Statement

Appendix B records the evaluation matrix and protocol differences, and Appendix C reports the broader results. The anonymous artifact contains code, parameter records, frozen predictions, source hashes, and plotting inputs. Reproduction also requires the surrogate generator and prompts, filtering and ordering rules, tokenizer, seeds, selected surrogates, initial model and optimizer states, shifted-label alignment, averaging precision, gradient scaling, and clipping rules.

References

Martín 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. doi: 10.1145/2976749.2978318. URL https://research.google.com/pubs/archive/45428.pdf.

Mislav Balunović, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. LAMP: Extracting Text from Gradients with Language Model Priors. In *Advances in Neural Information Processing Systems*, volume 35, pp. 7641–7654, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/32375260090404f907ceae19f3564a7e-Abstract-Conference.html.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/e995f98d56967d946471af29d7bf99f1-Abstract.html.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. URL https://jmlr.org/papers/v3/bengio03a.html.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting Training Data from Large Language Models. In *30th USENIX Security Symposium*

(USENIX Security 21), pp. 2633–2650, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

Sahil Chaudhary. Code Alpaca: An Instruction-following LLaMA model for code generation. GitHub repository, 2023. URL https://github.com/sahil280114/codealpaca.

Chen Chen, Xiang Gao, Xianshun Wang, Chengran Li, Shengyu Xia, Xueluan Gong, Linru Zhang, Qian Wang, and Kwok-Yan Lam. The Art of Mixology: Mixup-based Obfuscation for Privacy-Preserving Split Learning in Large Language Models, 2026. URL https://arxiv.org/abs/2606.16801. Preprint.

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 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2904–2918, 2024. doi: 10.1145/3658644.3690295. URL https://doi.org/10.1145/3658644.3690295.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. DialogSum: A Real-Life Scenario Dialogue Summarization Dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 5062–5074, 2021. doi: 10.18653/v1/2021.findings-acl.449. URL https://aclanthology.org/2021.findings-acl.449/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168. Preprint.

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. doi: 10.18653/v1/2021.findings-emnlp.305. URL https://aclanthology.org/2021.findings-emnlp.305/.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1383–1392, 2018. doi: 10.18653/v1/P18-1128. URL https://aclanthology.org/P18-1128/.

Minxin Du, Xiang Yue, Sherman S. M. 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. doi: 10.1145/3576915.3616592. URL https://doi.org/10.1145/3576915.3616592.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting Gradients - How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, volume 33, pp. 16937–16947, 2020. URL https://papers.neurips.cc/paper_files/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html.

Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, 2018. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label Leakage and Protection in Two-party Split Learning. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=cOtBRgsf2fO.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large Language Models Can Be Strong Differentially Private Learners. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=bVuP3ltATMz.

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004. URL https://aclanthology.org/W04-1013/.

Junlin Liu, Xinchen Lyu, Qimei Cui, and Xiaofeng Tao. Similarity-Based Label Inference Attack Against Training and Inference of Split Learning. *IEEE Transactions on Information Forensics and Security*, 19:2881–2895, 2024. doi: 10.1109/TIFS.2024.3356821. URL https://doi.org/10.1109/TIFS.2024.3356821.

Yi Liu, Weixiang Han, Chengjun Cai, Xingliang Yuan, and Cong Wang. PrivTune: Efficient and Privacy-Preserving Fine-Tuning of Large Language Models via Device-Cloud Collaboration. In *IEEE INFOCOM 2026 - IEEE Conference on Computer Communications*, pp. 1–10, 2026a. doi: 10.1109/INFOCOM59046.2026.11571688. URL https://doi.org/10.1109/INFOCOM59046.2026.11571688.

Zheng Liu, Chen Gong, Terry Yue Zhuo, Kecen Li, Weichen Yu, Matt Fredrikson, and Tianhao Wang. PrivCode: When Code Generation Meets Differential Privacy. In *Network and Distributed System Security Symposium*, 2026b. doi: 10.14722/ndss.2026.240936. URL https://www.ndss-symposium.org/ndss-paper/privcode-when-code-generation-meets-differential-privacy/.

Zihan Liu, Yizhen Wang, Rui Wang, and Sai Wu. DualGuard: A Parameter Space Transformation Approach for Bidirectional Defense in Split-Based LLM Fine-Tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17065–17080, 2025. doi: 10.18653/v1/2025.acl-long.835. URL https://aclanthology.org/2025.acl-long.835/.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Jinglong Luo, Zhuo Zhang, Yehong Zhang, Shiyu Liu, Ye Dong, Hui Wang, Yue Yu, Xun Zhou, and Zenglin Xu. SecP-Tuning: Efficient Privacy-Preserving Prompt Tuning for Large Language Models via MPC. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7f2e5d16c2c5ac161ce7aa44af35c0fe-Abstract-Conference.html.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 1273–1282, 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Shiyu Miao, Yunlong Mao, Zirui Huang, Liang Yao, Tianshuo Zheng, Yanhui Gu, Fan Liu, and Sheng Zhong. Gradient Mirage: Trainable yet Label-Unidentifiable Gradients in Large Language Model Split Learning, 2026. URL https://arxiv.org/abs/2608.18767. Preprint.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ec2e7a896f8250986b3907f57621ce94-Abstract-Conference.html.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M. Rush. Text Embeddings Reveal (Almost) As Much As Text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12448–12460, 2023. doi: 10.18653/v1/2023.emnlp-main.765. URL https://aclanthology.org/2023.emnlp-main.765/.

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. doi: 10.1145/3460120.3485259. URL https://doi.org/10.1145/3460120.3485259.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language-models.pdf.

Aki Rehn, Linzh Zhao, Mikko Heikkilä, and Antti Honkela. On Optimal Hyperparameters for Differentially Private Deep Transfer Learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/895834beb08f2806edf3e998851ec2ff-Abstract-Conference.html.

Amrita Roy Chowdhury, David Glukhov, Divyam Anshumaan, Prasad Chalasani, Nicholas Papernot, Somesh Jha, and Mihir Bellare. Pr$\epsilon\epsilon$mpt: Sanitizing Sensitive Prompts for LLMs. In *Network and Distributed System Security Symposium*, 2026. doi: 10.14722/ndss.2026.231277. URL https://www.ndss-symposium.org/ndss-paper/pr%CE%B5%CE%B5mpt-sanitizing-sensitive-prompts-for-llms/.

Xicong Shen, Yang Liu, Yi Liu, Peiran Wang, Huiqi Liu, Jue Hong, Bing Duan, Zirui Huang, Yunlong Mao, Ye Wu, and Sheng Zhong. 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. doi: 10.24963/ijcai.2025/57. URL https://www.ijcai.org/proceedings/2025/57.

Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. Beyond Memorization: Violating Privacy via Inference with Large Language Models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9028b8a3ca98f58e373f0c1497a17448-Abstract-Conference.html.

Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. Language Models are Advanced Anonymizers. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f478b2e8ad9ff0756bf5b79fb31c330f-Abstract-Conference.html.

Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. Soteria: Provable Defense Against Privacy Leakage in Federated Learning From Representation Perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9311–9319, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Sun_Soteria_Provable_Defense_Against_Privacy_Leakage_in_Federated_Learning_From_CVPR_2021_paper.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data, 2018. URL https://arxiv.org/abs/1812.00564. Preprint.

Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. NoPeek: Information leakage reduction to share activations in distributed deep learning. In *2020 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 933–942, 2020. doi: 10.1109/ICDMW51313.2020.00134. URL https://doi.org/10.1109/ICDMW51313.2020.00134.

Dong Yan, Jian Liang, Ran He, and Tieniu Tan. Stop Tracking Me! Proactive Defense Against Attribute Inference Attack in LLMs. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f1c131d41c3d5119a8de329140b3384e-Abstract-Conference.html.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang,

Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report, 2024. URL https://arxiv.org/abs/2412.15115. Technical report.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially Private Fine-tuning of Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Q42f0dfjECO.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/60a6c4002cc7b29142def8871531281a-Abstract.html.

Xiaochen Zhu, Xinjian Luo, Yuncheng Wu, Yangfan Jiang, Xiaokui Xiao, and Beng Chin Ooi. Passive Inference Attacks on Split Learning via Adversarial Regularization. In *Network and Distributed System Security Symposium*, 2025. URL https://www.ndss-symposium.org/ndss-paper/passive-inference-attacks-on-split-learning-via-adversarial-regularization/.

Tianyuan Zou, Zinan Lin, Sivakanth Gopi, Yang Liu, Ya-Qin Zhang, Robert Sim, Xin Deng, and Sergey Yekhanin. PE-SGD: Differentially Private Deep Learning via Evolution of Gradient Subspace for Text. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b2e20d7402c9985eae4ba924c65370a8-Abstract-Conference.html.

A Proofs and Scope of the Guarantees

Notation and comparison. For a sequence $X = (x_1, \ldots, x_I)$, let T_p and T_u be its private-token and unprotected-token positions. For the guarantees below, the unprotected context C records the unprotected tokens at their original positions. It also records the positions and token length of each private datum item. Thus, C specifies where tokens can be inserted without using the removed private tokens. The tokenizer and the public pre-trained models and auxiliary data are fixed throughout a comparison. Probabilities below condition on these fixed public quantities, even when they are not written in the conditioning expression.

The symbol d denotes one private datum item, which can contain several tokens. In probability statements, d is random. The symbol $d^\star$ denotes one particular possible token sequence for that item.

For a batch of B sequences at round t, define

$$C^{(t)} = (C_1^{(t)}, \ldots, C_B^{(t)}), \qquad Z'^{(t)} = (Z_1'^{(t)}, \ldots, Z_B'^{(t)}).$$

Here $Z_b'^{(t)}$ is the complete smashed-data matrix sent for sequence b, including all private-token and unprotected-token rows. Among K generated surrogates, K' are selected. The k-th selected surrogate gives a sequence $X_{b,k}^{(t)}$ by replacing all private tokens and retaining every unprotected token at its original position, as in Eq. (4). At head parameters $\phi^{(t)}$, Eq. (5) computes

$$Z_{b,k}^{(t)} = f_{\phi^{(t)}}(X_{b,k}^{(t)}), \qquad Z_b'^{(t)} = \frac{1}{K'} \sum_{k=1}^{K'} Z_{b,k}^{(t)}, \qquad 1 \le K' \le K.$$

All random choices that can affect these computations are included in the stated conditions. Once the inputs, states, and these choices are fixed, an execution is determined. The comparison uses the same computation settings in both executions.

How the four guarantees fit together. Theorems 1 and 3 concern privacy. The first fixes the head and compares the smashed data produced by different private tokens with the same C. The third follows this comparison through every model update and every exchanged message. Theorems 2 and 4 concern what AliasGuard preserves for training. The second bounds the change in smashed data at unprotected-token positions. The fourth shows that the one-hot vectors for unprotected next tokens stay exactly the same.

The implementation checks follow this order. Appendix H.5 checks which inputs determine surrogate construction; Appendix F compares messages after private tokens change; Appendix E.10 checks complete training runs. Reconstruction results in Appendices C–E and utility results in Appendix G then measure how the trained methods perform.

A.1 Forward privacy given unprotected context

Theorem 1 (Forward privacy given the unprotected context)**.** *Fix a round t, batch size B, candidate counts K, K', head parameters $\phi^{(t)}$, and the other forward computation settings. The probability statement in this theorem concerns this fixed-head experiment.*

Assume that generating, checking, ordering, and selecting surrogates uses only $C^{(t)}$, the fixed public quantities, and random choices. The joint distribution of all random choices used in surrogate construction and head computation is independent of all private tokens conditional on $C^{(t)}$. All other inputs to the head, including positions and attention masks, are fixed or constructed from these same permitted inputs.

Then any two assignments of the private tokens that have the same $C^{(t)}$ give the same conditional distribution of $Z'^{(t)}$. Consequently, for any private datum item d in the batch and any possible $d^\star$,

$$\Pr(d = d^\star \mid C^{(t)}, Z'^{(t)}) = \Pr(d = d^\star \mid C^{(t)}) \quad \textit{almost surely}. \tag{9}$$

Here "almost surely" means that the equality may exclude observations having total probability zero. The distributional assertion covers the joint assignment of all private tokens.

Proof. Compare two batches with the same $C^{(t)}$ but possibly different private tokens. Use the same random choices in their two executions. This is possible because, given $C^{(t)}$, the choices have the same joint distribution in both executions.

The surrogate generator receives the same inputs. Checking, ordering, and selection also receive the same inputs. They therefore produce the same selected surrogates in the same order. Replacing every private-token position with these surrogates gives the same $X^{(t)}_{b,k}$ for every b, k: the unprotected tokens were already identical, and the removed private tokens are no longer inputs to the head.

The head parameters, other head inputs, and random choices are identical. Hence the complete matrices $Z^{(t)}_{b,k}$ are identical, including their unprotected-token rows. Averaging gives identical $Z'^{(t)}_b$ and therefore identical $Z'^{(t)}$.

This equality holds for each fixed realization of the common random choices. Averaging over their common distribution proves that the law of $Z'^{(t)}$, conditional on $C^{(t)}$, is unchanged by the private tokens. In particular, let A be any measurable set of possible smashed-data arrays, meaning a set of outputs for which a probability is defined. Then

$$\begin{aligned}&\Pr(d = d^\star,\ Z'^{(t)} \in A \mid C^{(t)})\\ &\quad = \Pr(d = d^\star \mid C^{(t)}) \Pr(Z'^{(t)} \in A \mid C^{(t)}).\end{aligned}$$

The product on the right says that, once $C^{(t)}$ is given, knowing d does not change the probability that $Z'^{(t)}$ lies in A. This property is called conditional independence. It gives Eq. (9): observing the smashed data leaves the probability of each possible private datum item unchanged. Expressing the argument with sets A also covers smashed data with continuously varying entries. □

Interpreting forward privacy. The result excludes additional information beyond the unprotected context and fixed public quantities; C may itself reveal a private datum item. For one sequence, Eq. (9) gives Eq. (3). A surrogate can coincide with the removed private tokens by chance. Testing a

candidate against the removed tokens to force inclusion or exclusion can introduce private-token dependence and violate the hypotheses. The head can be updated between rounds under the conditions of Theorem 3.

A.2 Changes in smashed data at unprotected-token positions

Theorem 2 (Averaging preserves a bound on smashed-data changes)**.** *Fix b, t, the head computation, and the selected sequences $X_{b,k}^{(t)}$. Let $T_{u,b}^{(t)}$ index the unprotected-token positions in $X_b^{(t)}$. Define*

$$Z_b^{(t)} = f_{\phi^{(t)}}(X_b^{(t)}), \qquad \varepsilon_k = \left\| (Z_{b,k}^{(t)} - Z_b^{(t)})_{T_{u,b}^{(t)}} \right\|_F .$$

The subscript selects the rows indexed by $T_{u,b}^{(t)}$. For a matrix A, its Frobenius norm is $\|A\|_F = (\sum_{i,j} A_{ij}^2)^{1/2}$. Then

$$\left\| (Z_b'^{(t)} - Z_b^{(t)})_{T_{u,b}^{(t)}} \right\|_F \le \frac{1}{K'} \sum_{k=1}^{K'} \varepsilon_k. \tag{10}$$

The values ε_k measure the differences between these fixed outputs. In particular, for $\varepsilon \ge 0$, if $\varepsilon_k \le \varepsilon$ for every selected sequence, the left-hand side is at most ε.

Proof. Subtracting $Z_b^{(t)}$ from the mean gives

$$Z_b'^{(t)} - Z_b^{(t)} = \frac{1}{K'} \sum_{k=1}^{K'} (Z_{b,k}^{(t)} - Z_b^{(t)}).$$

The same unprotected-token positions occur in every sequence. Selecting these rows before averaging gives the same entries as selecting them afterwards. The triangle inequality says that the size of a sum is at most the sum of the individual sizes. Applying it to these differences and multiplying by $1/K'$ gives

$$\begin{aligned} \left\| (Z_b'^{(t)} - Z_b^{(t)})_{T_{u,b}^{(t)}} \right\|_F &= \left\| \frac{1}{K'} \sum_{k=1}^{K'} (Z_{b,k}^{(t)} - Z_b^{(t)})_{T_{u,b}^{(t)}} \right\|_F \\ &\le \frac{1}{K'} \sum_{k=1}^{K'} \left\| (Z_{b,k}^{(t)} - Z_b^{(t)})_{T_{u,b}^{(t)}} \right\|_F = \frac{1}{K'} \sum_{k=1}^{K'} \varepsilon_k. \end{aligned} \tag{11}$$

If every summand is at most ε, their mean is at most ε. □

Relating the bound to utility. Equation (10) connects each surrogate's effect to the average sent to the server. If every selected surrogate changes the unprotected rows only slightly, their average satisfies the same bound. The quantities ε_k are measured differences; the theorem does not itself force them to be small. Appendix H.6 measures these changes for stored surrogates.

The trunk and tail use the complete smashed data, including private-token positions, to make predictions. Section 4.4 therefore measures prediction and final task utility separately. The original smashed data are a reference for this analysis. The client selects surrogates using only the inputs in Theorem 1; it does not use distance to the original private sequence as a selection rule.

The displayed bound uses an exact arithmetic mean. If the implementation returns $\widehat{Z}_b'^{(t)} = Z_b'^{(t)} + E_b^{(t)}$, where $E_b^{(t)}$ is its averaging and conversion error, the corresponding bound is

$$\left\| (\widehat{Z}_b'^{(t)} - Z_b^{(t)})_{T_{u,b}^{(t)}} \right\|_F \le \frac{1}{K'} \sum_{k=1}^{K'} \varepsilon_k + \left\| (E_b^{(t)})_{T_{u,b}^{(t)}} \right\|_F .$$

The additional term accounts for arithmetic error in the implemented average.

A.3 BACKWARD PRIVACY THROUGHOUT TRAINING

Let $\mathcal{C}$ be the ordered collection of the unprotected contexts of all sequences in the fixed training schedule. Fix the public data, schedule, loss-position masks, attention connections, surrogate-generation rules, and parameter-update rules. Any data-dependent choice affecting execution must use only $\mathcal{C}$, permitted current states, and the stated random choices.

Let $\Theta^{(t)}$ contain the head, trunk, and tail parameters before round t. It also contains the optimizer states, such as the moving averages used to compute updates, and all other stored quantities that can affect a later input, message, or update. Let $O^{(T)}$ denote all messages visible to the server through round T, together with the server parameters and optimizer states it observes. These messages include smashed data, trunk outputs, and gradients sent in both directions. The observation $O^{(T)}$ also includes any random choices known to the server. It excludes unmodeled side channels such as wall-clock timing.

Theorem 3 (Backward privacy throughout training)**.** *Assume the input restrictions on surrogate construction in Theorem 1 at every round. Conditional on $\mathcal{C}$ and the fixed public quantities, the joint distribution of $\Theta^{(1)}$ and all random choices is independent of all private tokens. Each computation and update uses only the current states, sequences constructed from $\mathcal{C}$ and surrogates, their averaged one-hot vectors, and these random choices.*

Then two executions with the same $\mathcal{C}$, initial states, and random choices have identical messages and states $\Theta^{(t)}$ at every round. Consequently, for every finite T, any private datum item d, and any possible $d^\star$,

$$\Pr(d = d^\star \mid \mathcal{C}, O^{(T)}) = \Pr(d = d^\star \mid \mathcal{C}) \quad \textit{almost surely}. \tag{12}$$

The message distribution is unchanged jointly for all assignments of the private tokens consistent with $\mathcal{C}$.

Proof. We first show which inputs determine the loss and gradients, and then show that updates preserve equality across rounds.

The loss and gradients. Fix one round and omit its superscript t in this calculation. Let $I_b \geq 2$ be the unpadded length of sequence b, and let V be the vocabulary size. For $1 \leq i < I_b$, the one-hot vector $Y_{b,i,k} \in \mathbb{R}^V$ has one in the vocabulary entry for token $(X_{b,k})_{i+1}$ and zero elsewhere. Its average and the sequence loss in Eqs. (7)–(8) are

$$Y'_{b,i} = \frac{1}{K'} \sum_{k=1}^{K'} Y_{b,i,k}, \qquad \mathcal{L}_{\mathrm{AG},b} = -\frac{1}{I_b - 1} \sum_{i=1}^{I_b - 1} (Y'_{b,i})^\top \log P_{b,i}.$$

Every entry of $Y'_{b,i}$ is nonnegative and its entries sum to one. If the selected next tokens differ, more than one entry can be positive. For example, selecting the same token twice among three sequences gives its entry weight $2/3$. With $U_b = g_\psi(Z'_b)$, define $a_{b,i} = h_\omega(U_b)_i$ to be the vector of scores before softmax, so that $P_{b,i} = \mathrm{softmax}(a_{b,i})$. The one-hot vectors and their averages are constants when differentiating the model.

Substituting the mean of the one-hot vectors gives

$$\mathcal{L}_{\mathrm{AG},b} = \frac{1}{K'} \sum_{k=1}^{K'} \left[-\frac{1}{I_b - 1} \sum_{i=1}^{I_b - 1} Y_{b,i,k}^\top \log P_{b,i} \right]. \tag{13}$$

Every term uses the same $P_{b,i}$ obtained by passing the averaged smashed data through the trunk and tail once.

For vocabulary indices $v, w \in \{1, \ldots, V\}$, let $P_{b,j,v}$ and $a_{b,j,w}$ denote the corresponding vector entries. Differentiating softmax gives

$$\frac{\partial \log P_{b,j,v}}{\partial a_{b,j,w}} = \mathbf{1}[v = w] - P_{b,j,w}.$$

Here $\mathbf{1}[v = w]$ is one if the indices agree and zero otherwise. Because $\sum_v Y'_{b,j,v} = 1$,

$$\frac{\partial \mathcal{L}_{\mathrm{AG},b}}{\partial a_{b,j,w}} = \frac{P_{b,j,w} - Y'_{b,j,w}}{I_b - 1}, \qquad \frac{\partial \mathcal{L}_{\mathrm{AG},b}}{\partial a_{b,j}} = \frac{P_{b,j} - Y'_{b,j}}{I_b - 1}. \tag{14}$$

The batch loss is the mean $\mathcal{L}_{\mathrm{AG}} = B^{-1}\sum_b \mathcal{L}_{\mathrm{AG},b}$. Assume separate sequences do not attend to each other. For row $U_{b,i} \in \mathbb{R}^{d_u}$ of the trunk output, $1 \le i \le I_b$, the gradient $\Delta_{b,i} = \partial\mathcal{L}_{\mathrm{AG}}/\partial U_{b,i}$ is therefore

$$\Delta_{b,i} = \frac{1}{B(I_b-1)} \sum_{j=1}^{I_b-1} \left(\frac{\partial a_{b,j}}{\partial U_{b,i}}\right)^{\top} (P_{b,j} - Y'_{b,j}). \tag{15}$$

The derivative matrix has V rows and d_u columns. The sum over j is needed because one trunk-output row can affect several tail predictions. The factor $1/B$ comes from the batch mean and $1/(I_b-1)$ from the sequence mean. All derivatives use the parameters before this round's updates.

The same loss determines the tail and trunk parameter gradients. Differentiating the mean smashed data gives

$$\frac{\partial\mathcal{L}_{\mathrm{AG}}}{\partial Z_{b,k}} = \frac{1}{K'}\frac{\partial\mathcal{L}_{\mathrm{AG}}}{\partial Z'_b}.$$

The head accumulates the parameter gradients from all its selected branches before its update. The batch factor $1/B$ is already included in $\mathcal{L}_{\mathrm{AG}}$.

Equality across rounds. Use identical initial states and random choices for two assignments of the private tokens with the same $\mathcal{C}$. At round 1 their $\Theta^{(1)}$ are equal by construction. Suppose their states are equal before some round t.

The proof of Theorem 1 gives identical selected sequences and smashed data at this round's common head. The selected sequences also give identical one-hot vectors and their averages. Identical trunk and tail parameters then give identical trunk outputs and predicted probabilities.

The two executions evaluate the same loss expression with the same model parameters, intermediate arrays, and averaged one-hot vectors. Differentiating this common computation gives identical gradients, including the communicated gradient in Eq. (15). Identical gradients, optimizer states, update rules, and random choices give identical states $\Theta^{(t+1)}$. Their sent and returned messages also agree.

The states agree initially, and agreement before one round implies agreement after it. Repeating this argument proves equality at every round; this is induction over the training rounds. Thus $O^{(T)}$ is identical for each fixed choice of the common initial states and random choices. Their joint distribution conditional on $\mathcal{C}$ is the same for every assignment of the private tokens. Averaging over it makes the conditional distribution of $O^{(T)}$ the same as well. The same conditional-probability factorization used in Theorem 1 proves Eq. (12). □

Conditions on training updates. The head, trunk, and tail can all be updated when the conditions apply to every update and every persistent state. In particular, an additional tail loss using the original private tokens' one-hot vectors, an initial state storing those tokens, or a stopping rule based on their original loss can invalidate the result. Random choices may be correlated across rounds; their joint distribution with the initial states must satisfy the stated independence condition.

For several sequences or rounds, Eq. (12) conditions on all their unprotected contexts. A guarantee conditioned on one sequence's C requires additional assumptions about the information in the other contexts. For a single sequence, both its forward message and its forward–backward message pair are contained in $O^{(T)}$, giving Eq. (6).

The derivative calculation above uses the stated unpadded, sequence-averaged loss. Padding or loss masking requires the implementation's actual fixed masks and normalization. Deterministic conversion, scaling, or clipping applied identically in both executions preserves message equality; if the transmitted array is such a processed gradient, its definition must distinguish that array from the mathematical derivative in Eq. (15).

A.4 One-hot vectors for unprotected next tokens

Theorem 4 (Preservation of one-hot vectors for unprotected next tokens)**.** *Fix b, t and write I for the length of $X_b^{(t)}$. Suppose every selected $X_{b,k}^{(t)}$ retains each unprotected token at its original position.*

Define

$$Y_{b,i}^{(t)} = \text{onehot}((X_b^{(t)})_{i+1}), \qquad Y_{b,i,k}^{(t)} = \text{onehot}((X_{b,k}^{(t)})_{i+1}),$$
$$Y_{b,i}^{\prime(t)} = \frac{1}{K'} \sum_{k=1}^{K'} Y_{b,i,k}^{(t)}.$$

Then, for every $1 \le i < I$ *such that* $i+1 \in T_{u,b}^{(t)}$,

$$Y_{b,i}^{\prime(t)} = Y_{b,i}^{(t)}. \tag{16}$$

Proof. The label used at prediction position i is the token at position $i+1$. If that next token is unprotected, every selected sequence retains it:

$$(X_{b,k}^{(t)})_{i+1} = (X_b^{(t)})_{i+1} \qquad \text{for every } k.$$

The one-hot vectors are therefore identical. Averaging K' copies of the same vector leaves it unchanged:

$$Y_{b,i}^{\prime(t)} = \frac{1}{K'} \sum_{k=1}^{K'} Y_{b,i}^{(t)} = Y_{b,i}^{(t)}. \qquad \square$$

Which labels are preserved. The condition $i+1 \in T_u$ means that the token predicted at position i is unprotected. Every selected sequence keeps that token and its position, so its one-hot vector is unchanged after averaging.

A.5 Implementation details and verification scope

The client can compute the AliasGuard loss by gathering the log probabilities at the selected next-token labels and averaging them. This evaluates Eq. (13) without storing full vocabulary-sized label vectors. All terms must use the prediction from the averaged smashed data.

After joining SL-FT, the client prepares the first batch's replacement sequences before the first forward pass; this preparation contributes to training time. The client can prepare the next batch while the server computes the current trunk outputs or the gradient returned to the head. Any unfinished preparation delays the next round. The next smashed data must be evaluated using the updated head. For B original sequences and K' replacements per sequence, the client processes BK' sequences through the head and sends B averaged outputs. Batching can improve hardware utilization; the head computation and its backward gradients process all BK' replacement sequences. Appendix H.4 reports measured component times.

Previously recorded checks used 30 joint surrogate inputs per sequence, selected four to six distinct candidates, and cached complete outputs from a deterministic fixed head. Their numerical equivalence results apply to that fixed-head implementation and its registered inputs. The real-valued bound in Theorem 2 includes a rounding-error allowance for finite-precision arithmetic.

B Experimental Settings and Parameters

The experiments connect three questions. First, does changing the private tokens change AliasGuard's messages when C is fixed? Second, how often can the evaluated reconstruction procedures recover those tokens? Third, how well does the trained model perform its task? Table 3 locates the evidence for each question. The message checks test the implemented construction; the reconstruction and utility scores measure its observed consequences.

B.1 Comparison, data and seeds

The main comparison measures performance across tasks and models. Table 1(a) compares 45 cases: five methods, three datasets and three models, at seed 3407. The methods are NoDef (M0), AliasGuard (M4), DualGuard (M5), DP-forward and DP-SGD*. One case is one trained method on

Table 3: Questions answered by the experiments. Each row identifies the measurements and the relevant theoretical result. Theorems establish properties under their stated conditions; experiments check particular executions and trained models.

Question	Measurement and location	Theoretical connection
How do methods compare?	Five methods in Table 1(a); training seeds in Appendix D; reconstruction settings in Appendix E	Privacy goals in Theorems 1, 3
Do messages depend on private tokens?	Surrogate inputs in Appendix H.5; message equality in Appendix F; training replay in Appendix E.10	Theorems 1, 3
What task information is retained?	Unprotected labels and predictions, generated answers, and code validity in Appendix G	Theorems 2, 4
Why replace sequences and labels?	Four combinations at fixed model parameters in Appendix H.7	Forward and backward constructions
What does increasing K' change?	Recovery, task scores, time, memory and communication in Appendix H	Averaging in Theorems 2, 4

one dataset and model. A checkpoint stores the model parameters at one training update. Utility uses the final checkpoint, selected before evaluation. Privacy evaluation uses observations captured at their corresponding updates. Test sequences within a case share one trained model.

Each formal run makes one ordered pass with batch size one: 169 original training sequences and updates for Qwen, or 154 for GPT-2, selected from the source training split. Within each dataset/model comparison, methods share the training sequence identities, tokenizer and evaluation splits. The H4–1.5B historical and DP runs use different orders; their reconstruction target identities and observation times are audited separately in Appendix E.5. AliasGuard constructs the corresponding surrogate sequences and surrogate labels. The baselines use the original sequences and their next-token labels. Thus the methods begin with the same original training sequences and apply their respective training algorithms. The private data are code identifiers in H4 and person names in DialogSum and GSM8K. Repeated occurrences are recorded at their respective token positions. GSM8K numerical quantities remain unchanged.

Table 4: Sequence counts and maximum generated tokens. Public columns describe the auxiliary decoder data. GPT-2 excludes the same one H4 and two DialogSum test sequences across methods because their native inputs exceed its supported length. Scores use the retained test sequences.

Dataset	Train Qwen/GPT-2	Test Qwen/GPT-2	Public fit/validation	Output cap
CodeAlpaca-H4	169/154	2,003/2,002	819/205	128
DialogSum	169/154	500/498	669/167	128
GSM8K	169/154	1,319/1,319	819/205	512

The public auxiliary sequences train and select the reconstruction decoders. The public-fit sequences train the decoder D_{dec} introduced in the Introduction, which predicts tokens from smashed data or trunk outputs. The separate public-validation sequences compare the fitted decoders and select their settings. When a reconstruction procedure combines evidence from forward and backward observations, it gives each possible reconstructed sequence a numerical score; public-validation sequences also select how these scores are combined. The registered IDs of these public sequences do not overlap with the formal training, validation, or test IDs; similar text and repeated templates can remain. A *reconstruction target* is an original training sequence whose messages were recorded during its update. We evaluate 128 such sequences because the privacy question concerns what those messages reveal. The reconstruction procedure does not receive their original private tokens. Their private-token position and private datum item counts appear in Table 8; token counts depend on the tokenizer.

The stored surrogate sequences for an original sequence form its surrogate bank, prepared before formal training. The five-method comparison uses seed 3407; Appendix D reports seed 3408 separately for AliasGuard. Appendix D.2 additionally reports a matched GSM8K comparison of AliasGuard and DualGuard at seeds 3407, 3409 and 3410, summarized in Table 1(b). Its seed-3407 models are newly trained. Appendix D.3 reports a separate matched GSM8K NoDef–AliasGuard comparison at seeds 3407, 3409 and 3410. It reuses the six Qwen AliasGuard models in Appendix D.2; checkpoint hashes and answer counts match. Its training sequence order and surrogate bank differ from those of Table 1(a). These studies answer different questions: the main comparison covers five methods, the DualGuard study checks repeated training, and the NoDef study separates unprotected-token prediction from generated-answer accuracy. Their models are identified by the study as well as the seed. A seed controls the prepared sequence order and model random choices; it does not identify a checkpoint by itself. The historical and fixed-K' evaluators use reconstruction seed 9107. Bootstrap resampling in Appendix G uses seed 20260921 for unprotected-token scores and seed 3407 for name changes. Private perturbation randomness is separate from the public experiment seed. Differences in K', software and evaluation targets across the two seeds are specified in Appendix D.

B.2 Model partitions and optimization

Each model places two Transformer blocks in the client head and two in the client tail. The remaining blocks form the server trunk (Table 5). This keeps most blocks on the server and fixes the split within each model's method comparison. The head also contains input embeddings; the tail contains the final normalization and vocabulary projection. These experiments compare methods at this split; they do not select a separate split for each method.

Table 5: Transformer-block partitions. The head includes input embeddings; the tail includes the final normalization and language-model projection. The two Qwen checkpoints use their shared tokenizer vocabulary, while GPT-2 requires separately tokenized inputs and surrogates.

Model	head/trunk/tail layers	Hidden size	Precision	Final update
GPT-2 small	2/8/2	768	FP32	154
Qwen2.5-1.5B	2/24/2	1,536	BF16	169
Qwen2.5-3B	2/32/2	2,048	BF16	169

NoDef, AliasGuard and the two DP implementations use AdamW (Loshchilov & Hutter, 2019) at learning rate 5×10^{-5} for each trainable partition and gradient-norm cap 1. Weight decay is zero for Qwen and 0.01 for GPT-2. In Qwen, the head uses LoRA with rank 8, scaling parameter 16 and dropout zero on the attention and feed-forward projections; trunk and tail have dense updates. GPT-2 uses dense updates with frozen token and position embeddings and disabled dropout. Attention uses PyTorch scaled dot-product attention (SDPA). FP32 stores numbers with 32 bits; BF16 uses the 16-bit bfloat16 format. TF32, a lower precision mode for some matrix operations, is disabled.

DualGuard retains its method-specific public preparation and optimization. Its public sequence-reconstruction model is trained for two epochs with batch size four, learning rate 10^{-3} and weight decay 10^{-5}. Local warm-up uses two epochs, batch size four, learning rate 10^{-5}, and weights 40 and 80 on its two auxiliary objectives. Formal training uses Adam, learning rate 10^{-5}, no weight decay and retention weight 10. Its LoRA rank/scaling/dropout are 2/32/0.1; the formal head is frozen, whereas NoDef and AliasGuard have trainable heads. The deployed targets are the Qwen query/value projections or GPT-2's fused attention projection. The warm-up has 86 updates for Qwen and 78 for GPT-2, in addition to the 169/154 formal updates. The warm-up and retention objectives are components of DualGuard, so we preserve its published training procedure. DualGuard reports experiments over its three objective weights and adopts $\lambda_1 = 40$, $\lambda_2 = 80$, and $\lambda_3 = 10$ as default values that consistently achieve near-optimal results across most tested scenarios (Liu et al., 2025). We retain these defaults together with DualGuard's published optimizer and LoRA settings.

AliasGuard surrogates. The prepared inputs store $K = 30$ surrogate sequences in each bank, with tokenizer-specific checks of the private spans and unchanged unprotected positions. The main comparison uses $K' = 3$, except historical H4–Qwen2.5-1.5B, which uses $K' = 4, 5, 6$ for 55, 67 and 47 updates. This exception also applies to the H4 example and measurements in Section 4.3.

Section 4.6 compares matched fixed-$K' = 1, 3, 5$ runs with nested prefixes of the same stored candidate ordering. The input packages and training checkpoints for these comparisons are kept separate. Appendix H.5 audits the implemented pipeline: 1,206 substitutions of private tokens leave its generator/checker inputs, ordering, selection and shared labels unchanged. It also reports complete structural checks and an independent blinded compatibility evaluation of selected surrogates.

DP-forward. The reported revision applies sequence-level normalization and Gaussian perturbation to the complete shallow head-output matrix (Du et al., 2023):

$$\widetilde{Z} = \frac{Z}{\|Z\|_F + 10^{-5}} + \xi, \qquad \xi_{ij} \sim \mathcal{N}(0, 0.68835485699^2).$$

The Frobenius norm $\|Z\|_F$ is the square root of the sum of squared entries. Dividing by it scales the complete smashed-data matrix to approximately unit length. Each entry then receives independent zero-mean Gaussian noise with standard deviation 0.68835485699. The radius is one and the single-release calibration uses $\epsilon = 16$, $\delta = 10^{-5}$ and sensitivity bound two. Perturbation covers the full head output, and the perturbed matrix passes directly to the trunk. Training retains the original next-token labels and loss masks. This adapts the forward mechanism to shallow causal-LM fine-tuning. The calibration applies to a single release; a guarantee for the evolving bidirectional trace would require additional accounting.

Appendix C.1 compares the observed F and B recovery under this fixed perturbation.

DP-SGD*. The asterisk identifies the gradient-perturbation implementation in the DualGuard code (Liu et al., 2025). Standard DP-SGD clips per-sequence parameter gradients, aggregates them and adds Gaussian noise (Abadi et al., 2016); the implementation here instead perturbs the exchanged gradient. For each returned gradient g at the trunk output, it computes

$$\bar{g} = \frac{g}{\max(1, \|g\|_\infty / 2000)}, \qquad \widetilde{g} = \bar{g} + \eta, \quad \eta_i \sim \mathrm{Laplace}(0, \mathrm{mean}\,|\bar{g}|).$$

Here $\|g\|_\infty$ is the largest absolute gradient entry. The division scales the whole gradient down if that entry exceeds 2000. The added Laplace noise has scale equal to the mean absolute entry after this scaling. The tail uses its original task gradient; trunk and head use $\widetilde{g}$. The forward path is unchanged. Its noise scale depends on the observed gradient through the formula above; the implementation provides no end-to-end privacy accountant. GPT-2 uses FP32; Qwen rounds the completed perturbation back to BF16. The continuous noise scores used in evaluation are approximations to the probabilities of finite-precision observations. Consequently, smashed data are still computed from the original sequence. Gradient perturbation can change B recovery, but it does not directly alter this forward computation. The task utility of DP-SGD* must therefore be read together with its F and BF recovery.

Interpreting privacy and utility for the perturbation baselines. Here, the privacy–utility trade-off means the difficulty of reducing private-token recovery from both forward smashed data and backward gradients while preserving task utility. We compare each baseline at its fixed deployed parameters.

The F/B/BF breakdown evaluates both observations. This matters because DP-forward perturbs smashed data while keeping the original next-token labels, whereas DP-SGD* perturbs the gradient while keeping the original forward input. Appendix C.1 connects each operation to its measured recovery and task score.

The published evidence follows the same distinction. DP-forward studies perturbation of smashed data (Du et al., 2023). DualGuard reports results for DP-forward and DP-SGD. DP-SGD names the baseline in that study; DP-SGD* names our implementation defined above. DualGuard reports that DP-forward reduces reconstruction from the forward observation but remains vulnerable to its backward evaluator. Its DP-SGD baseline reduces reconstruction from the backward observation but remains vulnerable to the forward evaluator (Liu et al., 2025). In the same paper, combining DP-forward with DP-SGD reduced downstream ROUGE-L F1 below 40% for GPT-2-Large on GSM8K and E2E in the tested setting (Liu et al., 2025). Our evaluation reports F, B, and BF recovery together with task utility for each deployed method.

B.3 RECONSTRUCTION AND TASK EVALUATION

Targets and recovery. A reconstruction target is an original training sequence whose messages were recorded during its update. Each case evaluates 128 such sequences. The program first saves its predictions. A separate scorer then compares them with the original private tokens at their recorded positions. Private-token recovery is the percentage of these positions predicted correctly. Every occurrence counts: if one identifier occurs twice, both positions enter the denominator. Appendix C.1 reports the denominators and complete-private-datum scores.

Within a task/model group, the five methods share seed 3407, original training sequence identities, one training pass, and the utility test sequences. Qwen has 169 updates and GPT-2 has 154. Utility evaluates the final checkpoint; reconstruction uses messages from the recorded updates. The historical H4–Qwen2.5-1.5B target and update differences are specified in Appendix E.5.

Forward, backward, and combined observations. F, B, and BF follow the observation choices of BiSR (Chen et al., 2024). F uses forward observations, B uses backward observations, and BF uses both. The reconstruction procedure generates possible private tokens and scores how well each resulting sequence explains the observations. Its simulation uses the evaluated method's algorithm. This is what makes the procedure adaptive to the method. Direct decoding is a separate evaluation: a fitted decoder predicts tokens from Z or U without iterative refinement. For AliasGuard the smashed data supplied in place of Z are Z'.

Public fitting and candidate selection. The historical direct decoders are linear maps with a bias, fitted separately for smashed data and trunk outputs. Their learning-rate and weight-decay pairs are $(5 \times 10^{-4}, 0)$, $(10^{-3}, 0)$, and $(10^{-3}, 10^{-4})$. Each is trained for five epochs with batch size one. The decoder with the lowest final public-validation cross-entropy is selected. Validation accuracy and then trial order break ties. Cross-entropy here averages the negative logarithm of the probability assigned to the correct public token.

The original tokens of public sequences supply the public fitting and validation labels. The original private tokens of the 128 reconstruction targets are available only to the final scorer. They are not used to fit a decoder, choose its settings, propose candidates, or select predictions. This separation lets the final score assess predictions made without reading the target's private tokens.

Computation budgets. An iteration is one update of the reconstruction program's proposed sequence. For the historical NoDef, AliasGuard, and DualGuard programs, F uses 200 iterations and B uses 32. BF first performs 32 B iterations and then 200 F iterations. HIGH increases this work to 96 B iterations, 400 F iterations, and a 400-iteration joint stage.

The current DP-forward and DP-SGD* programs allow B to propose tokens from the full vocabulary. They run F independently and also run F starting from B's result. BF retains candidates from all three runs and selects a complete sequence using scores fitted on public data. Its budget is therefore 32 B iterations and two 200-iteration F runs, followed by candidate scoring. The candidate list and the final selection rule are both part of a complete reconstruction program. B and BF each report their own selected sequence. Appendix B.3 explains why retaining B's candidates does not force BF to select B's prediction.

Permitted information and additional comparisons. The † on DP rows means that the evaluator receives the unprotected tokens. It copies those tokens into each proposed sequence, so their recovery is 100% by construction. Direct Z/U decoding still predicts the complete sequence. The historical AliasGuard evaluator also receives its selected surrogate sequences and surrogate labels. The separate K' study constructs surrogate sequences from public data and withholds the client's selected sequences and labels (Appendix H.1).

The five-method table compares reconstruction adapted to each method. Appendix E.7 additionally supplies the same unprotected context to every method and withholds private client parameters. Table 20 compares NoDef, AliasGuard, and DualGuard using the same H4 targets, 256 candidates, and scoring rule. These additional comparisons examine information and computation separately. Their results retain the target and observation conditions specified in Appendix E.5.

DP evaluation at matched training updates. For Qwen2.5-1.5B and 3B, the additional DP runs follow the 169-sequence order in the prepared AliasGuard inputs. They evaluate the 128 sequences selected by those inputs at the matching update numbers. The historical H4–1.5B row of Table 1 retains the different selection described above. Each model has 1,207 protected token positions in these 128 sequences. The reconstruction code read the unprotected tokens and its permitted forward or backward messages. It saved one prediction for each position before a separate scorer opened the original private tokens. F used 200 optimization steps, B used 32, and BF included B followed by two 200-step forward runs. The observed counts are:

Model	Method	F	B	BF
Qwen2.5-1.5B	DP-forward	1/1207	1207/1207	1201/1207
Qwen2.5-1.5B	DP-SGD*	1156/1207	1206/1207	1207/1207
Qwen2.5-3B	DP-forward	0/1207	1184/1207	1182/1207
Qwen2.5-3B	DP-SGD*	1182/1207	1188/1207	1207/1207

The predictions were sealed in 512 separate records, then checked for target identity, recorded update, communication hash, and duplicate or missing sequences. The public decoder inputs are identical across the old and new H4 packages: all 819 public-training and 205 public-validation records match after loading. These counts test the DP methods after their training order, target sequences, and observation updates have been aligned with the prepared AliasGuard inputs. They retain the DP reconstruction programs defined above. Table 1 reports the original comparison; the counts here provide the additional forward/backward breakdown.

For H4/GPT-2, the existing DP training records and the AliasGuard static input package agree on the selected 128 sequences and their update numbers. The two DP methods have 1,307 protected token positions in those sequences. DP-forward obtains F $0/1307$, B $1229/1307$, and BF $1230/1307$; DP-SGD* obtains F $1307/1307$, B $1288/1307$, and BF $1301/1307$. The 256 predictions passed separate identity and hash checks.

The DialogSum check uses the same 128 selected training sequences and update numbers as AliasGuard for each model. All six DP cases have independently checked prediction records; the two Qwen models each contain 462 protected token positions, and GPT-2 contains 488. The counts are:

Model	Method	F	B	BF
GPT-2 small	DP-forward	0/488	488/488	1/488
GPT-2 small	DP-SGD*	487/488	486/488	487/488
Qwen2.5-1.5B	DP-forward	0/462	462/462	35/462
Qwen2.5-1.5B	DP-SGD*	425/462	462/462	462/462
Qwen2.5-3B	DP-forward	0/462	462/462	11/462
Qwen2.5-3B	DP-SGD*	426/462	462/462	462/462

Table 7 compares these DP-forward results with the BF settings used in Table 1. The DialogSum/GPT-2 DP-SGD* case used a new 154-update training run with the original fixed mechanism; its sequence order, communication files and final checkpoint passed independent checks.

The GSM8K checks likewise cover the same 128 selected training sequences and update numbers for each model. GPT-2 has 566 protected positions, and each Qwen model has 491. All six cases passed independent prediction checks:

Model	Method	F	B	BF
GPT-2 small	DP-forward	0/566	509/566	509/566
GPT-2 small	DP-SGD*	566/566	560/566	564/566
Qwen2.5-1.5B	DP-forward	0/491	491/491	491/491
Qwen2.5-1.5B	DP-SGD*	407/491	491/491	491/491
Qwen2.5-3B	DP-forward	0/491	491/491	491/491
Qwen2.5-3B	DP-SGD*	426/491	491/491	491/491

The 18 DP cases above assess each DP method's forward and backward observations with its adapted reconstruction program. Their targets and observation updates were checked against the correspond-

ing AliasGuard inputs. The additional evaluation uses the candidate generation and BF scoring described above. Table 7 directly compares the two DialogSum DP-forward BF settings on the same recorded messages. Table 6 pairs the five newly trained DP checkpoints with their own final utility results. The four H4 runs use 169 updates and all 2,003 test sequences. The DialogSum/GPT-2 run uses 154 updates and the same 498 supported test sequences as the original comparison; two sequences exceed its supported length. Each checkpoint hash agrees between training and utility, and the prediction records have no missing or repeated sequence IDs.

Table 6: Final utility of the new DP checkpoints used for the per-view counts above. Task is lexical F1 for H4 and ROUGE-L for DialogSum. These checkpoint-specific results are distinct from the original runs in Table 1.

Dataset	Model	Method	Task (%)	Full PPL	C-PPL
H4	Qwen2.5-1.5B	DP-forward	18.465	4.277	2.796
H4	Qwen2.5-1.5B	DP-SGD*	51.059	2.644	1.935
H4	Qwen2.5-3B	DP-forward	20.025	3.765	2.672
H4	Qwen2.5-3B	DP-SGD*	52.855	2.658	1.948
DialogSum	GPT-2 small	DP-SGD*	33.208	8.133	5.957

Two BF evaluations of DP-forward. We evaluated two sets of adaptive BiSR settings on DialogSum. Both use the same trained DP-forward models and recorded messages, the same 128 target sequences, and B32 with two F200 runs. Their fitted public decoders and candidate-scoring parameters differ. We call them the original settings and the refitted settings in Table 7.

Table 7: Correctly reconstructed protected token positions for DP-forward on DialogSum. Each fraction uses the same targets and denominator within its row. The original BF settings give higher recovery on all three models; these are the DP-forward values in Table 1. Higher recovery means weaker privacy protection.

	Original settings		Refitted settings	
Model	B	BF	B	BF
GPT-2 small	473/488	473/488	488/488	1/488
Qwen2.5-1.5B	462/462	462/462	462/462	35/462
Qwen2.5-3B	462/462	462/462	462/462	11/462

Both BF procedures keep the candidates produced by B, add candidates from F and from F initialized with B, and score each complete candidate sequence. The score combines agreement with forward observations, agreement with backward observations, and probability under a public language model. Its relative weights are selected using public sequences whose private tokens are known for that public evaluation. The original settings use the archived decoder and scoring parameters. The refitted settings fit the decoder on 651 public sequences, select it on 159 other public sequences, and reserve 18 and eight further sequences for choosing and checking the scoring parameters. These sequences belong to the same public auxiliary pool described in Table 4.

Keeping a B candidate in the list does not force BF to select it. A candidate with a worse backward score can rank first after the forward score is added. With the original settings, B and BF have the same recovery counts; with the refitted settings, B recovers every protected position while BF selects substantially less accurate candidates. The changed decoder and scoring parameters can change both the candidate list and its ordering. Their separate contributions have not been isolated. The observed B result establishes that the backward observations remain useful for reconstruction under both evaluations.

The higher-BF statement applies to DialogSum. The later H4/Qwen tests also change training order and checkpoints: DP-forward BF becomes 99.50% and 97.93%, versus 99.25% and 96.19% in Table 1. Their utility is paired with those new checkpoints in Table 6. We retain each result with its measured training and reconstruction settings.

Table 26 now reports the verified archived NoDef/DualGuard/AliasGuard BF and HIGH results. Its H4–1.5B row retains the historical target population and is marked accordingly. DP HIGH was not run. Recovery rates and budgets are reported for each specified reconstruction procedure.

Utility uses unperturbed downstream inference of each trained final checkpoint, including for DP-forward. Generation is greedy, with one beam, no sampling, KV caching, at least one generated token and the caps in Table 4. Independent sequences are distributed among workers without changing the single-sequence rule. H4 reports lexical F1. DialogSum reports the highest ROUGE-L against its three references. GSM8K reports the percentage whose extracted numerical answer equals the reference answer. The task metrics use a 0–100 scale and are reported separately by dataset. For PPL, the model receives the correct preceding tokens and assigns a probability to the next correct token. We take the negative natural logarithm of each probability, average over all scored tokens, and exponentiate that average. A lower PPL means higher probability assigned to the correct tokens. Full PPL includes the prompt and reference answer. C-PPL applies this calculation only to the reference answer of each sequence, including its private tokens. DialogSum PPL uses its fixed first reference.

B.4 Hardware and reproducibility

Sequence coverage and identification of runs. Unprotected-token scores require known private-token positions in each reference answer. Appendix G.1 reports which sequences have those positions, which explicitly have none, and which have unknown positions. Unknown positions remain unknown. The scoring uses existing model outputs and introduces no new forward computation.

A seed alone does not identify an experiment. The three GSM8K–Qwen2.5-1.5B runs at seed 3407 have answer accuracies of 31.312% (main comparison), 41.092% (three-seed study), and 35.557% (fixed-$K' = 3$ study). The three-seed study uses a different surrogate bank, sequence order, and selected candidates. The main and fixed-K' studies share these inputs but already differ at their first recorded forward output. Their initialization, software, and hardware contributions to that difference have not been separated. We therefore compare methods within each specified study and keep its checkpoint and input hashes with the results.

A file hash is a value computed from its bytes; comparing hashes checks that two records refer to the same saved file. Input and checkpoint hashes identify the evaluated sequences and weights.

The main comparison uses NVIDIA A100, A40 and H100 GPUs. Model, tokenizer, input, checkpoint and code versions are recorded for each run. The timing comparison in Section 4.6 uses a common A100 configuration, detailed in Appendix H. Fixed seeds can still yield different floating-point results across devices and software versions. For the GSM8K–Qwen3B name-change diagnostic, both original and modified prompts were reevaluated on A100 using the same fixed checkpoints. Each reported pair therefore shares its evaluation hardware and software.

C Additional Results for the Main Comparison

These tables expand the seed-3407 comparison in Table 1(a), using the same checkpoints and method configurations. The recovery tables separate forward and backward observations. The utility tables evaluate the final trained models. DS/G8 denote DialogSum/GSM8K; Q1.5B/Q3B denote Qwen2.5-1.5B/Qwen2.5-3B. AliasGuard is shaded.

C.1 Reconstruction metrics and denominators

Private-token recovery counts correct predictions at private-token positions and divides by all evaluated private-token positions. Unprivate-token recovery uses the unprotected positions in both counts. A private datum item can contain several tokens. Exact-datum Top-1 counts an item as correct only when its entire predicted token sequence matches. Top-5 checks whether one of the five highest-scoring candidates matches that entire item. Both use the evaluator's saved candidate list.

These definitions explain why the scores differ. For H4–GPT-2, AliasGuard BF recovers 401/1,307 private-token positions (30.6809%), while only 16.9903% of complete private datum items match. Correctly predicting part of an item helps token recovery but does not count as exact-datum re-

covery. Direct Z/U decoding covers the complete recorded training run. F/B/BF/HIGH use 128 selected sequences within each case. The N columns state the direct-decoding and BF token counts separately.

Table 8: Additional privacy results for the 45 seed-3407 cases. P/O give recovery percentages at private-token/unprotected-token positions. Each N cell reports direct/BF denominators; Top-1/5 measure exact private-datum recovery. DP information conditions and evaluation protocols are specified in Appendix B.

Data	Model	Method	Z P/O	U P/O	BF Top-1	BF Top-5	N(P)	N(O)
H4	GPT-2	NoDef	96.26/98.01	88.33/94.31	99.88	99.88	1551/1307	21072/17808
		DP-Fwd†	0.32/3.51	0.00/2.32	91.99	93.33	1551/1307	21072/17808
		DP-SGD*†	96.26/97.99	87.81/94.86	99.27	99.88	1551/1307	21072/17808
		DualGuard	96.26/98.01	89.68/95.04	100.00	100.00	1551/1307	21072/17808
		AliasGuard	30.82/97.76	32.75/94.30	16.99	40.29	1551/1307	21072/17808
H4	Q1.5B	NoDef	93.53/94.99	72.39/80.72	96.55	96.66	1561/1248	15596/12078
		DP-Fwd†	0.13/1.40	0.00/0.43	98.95	100.00	1561/1207	15596/12251
		DP-SGD*†	93.08/95.00	73.35/80.95	100.00	100.00	1561/1207	15596/12251
		DualGuard	93.53/95.00	74.70/84.30	96.66	97.11	1561/1248	15596/12078
		AliasGuard	22.81/94.76	27.03/81.16	23.25	43.05	1561/1248	15596/12078
H4	Q3B	NoDef	94.43/95.51	61.11/76.97	98.03	98.03	1561/1207	15596/12251
		DP-Fwd†	0.13/1.24	0.77/0.56	95.35	96.40	1561/1207	15596/12251
		DP-SGD*†	94.43/95.47	57.98/74.35	100.00	100.00	1561/1207	15596/12251
		DualGuard	94.43/95.51	63.23/82.07	98.14	98.26	1561/1207	15596/12251
		AliasGuard	18.58/95.11	21.33/74.97	21.02	38.91	1561/1207	15596/12251
DS	GPT-2	NoDef	81.85/97.78	66.27/94.11	100.00	100.00	584/488	33938/28616
		DP-Fwd†	0.00/1.95	0.00/0.31	96.64	96.87	584/488	33938/28616
		DP-SGD*†	82.02/97.77	67.12/93.88	99.55	100.00	584/488	33938/28616
		DualGuard	81.85/97.77	72.43/94.35	100.00	100.00	584/488	33938/28616
		AliasGuard	2.74/97.76	3.25/93.92	2.91	14.09	584/488	33938/28616
DS	Q1.5B	NoDef	81.77/97.21	44.53/86.80	95.02	95.02	631/462	34127/26359
		DP-Fwd†	0.00/1.35	0.00/0.30	100.00	100.00	631/462	34127/26359
		DP-SGD*†	81.77/97.22	45.17/85.66	100.00	100.00	631/462	34127/26359
		DualGuard	81.77/97.21	45.48/89.19	94.31	94.31	631/462	34127/26359
		AliasGuard	4.44/97.22	3.96/85.87	4.50	16.35	631/462	34127/26359
DS	Q3B	NoDef	82.09/97.45	26.94/83.77	95.73	95.73	631/462	34127/26359
		DP-Fwd†	0.00/1.34	0.16/0.88	100.00	100.00	631/462	34127/26359
		DP-SGD*†	82.09/97.45	26.78/83.65	100.00	100.00	631/462	34127/26359
		DualGuard	82.09/97.44	33.28/85.04	95.02	95.02	631/462	34127/26359
		AliasGuard	3.96/97.44	3.96/82.94	3.32	16.82	631/462	34127/26359
G8	GPT-2	NoDef	70.88/96.36	38.64/88.31	99.28	99.28	704/566	22495/18448
		DP-Fwd†	0.00/1.06	0.00/0.10	90.17	90.41	704/566	22495/18448
		DP-SGD*†	70.88/96.35	37.36/87.45	99.52	100.00	704/566	22495/18448
		DualGuard	71.02/96.38	45.88/89.90	99.52	99.52	704/566	22495/18448
		AliasGuard	0.57/96.38	1.42/87.92	0.00	5.76	704/566	22495/18448
G8	Q1.5B	NoDef	65.35/97.15	27.18/86.49	81.19	81.96	710/491	29296/21905
		DP-Fwd†	0.00/1.80	0.00/1.22	100.00	100.00	710/491	29296/21905
		DP-SGD*†	65.49/97.15	31.27/86.68	100.00	100.00	710/491	29296/21905
		DualGuard	65.49/97.15	30.00/89.23	82.73	83.51	710/491	29296/21905
		AliasGuard	1.41/97.15	0.70/86.45	2.06	6.70	710/491	29296/21905
G8	Q3B	NoDef	67.61/97.44	21.27/82.89	84.28	84.79	710/491	29296/21905
		DP-Fwd†	0.00/1.68	0.00/0.48	100.00	100.00	710/491	29296/21905
		DP-SGD*†	67.89/97.44	24.51/82.67	100.00	100.00	710/491	29296/21905
		DualGuard	67.61/97.44	25.21/85.83	85.57	85.82	710/491	29296/21905
		AliasGuard	2.25/97.41	0.42/81.55	3.09	8.25	710/491	29296/21905

Table 9: Historical F, B, and BF private-token recovery (%) for all 45 seed-3407 cases. Protocol differences follow Table 1; reconstruction from complete candidate sequences is reported separately.

		H4			DialogSum			GSM8K		
View	Method	GPT-2	Q1.5B	Q3B	GPT-2	Q1.5B	Q3B	GPT-2	Q1.5B	Q3B
F	NoDef	100.00	97.12	98.18	100.00	95.24	95.45	99.47	79.63	84.11
	DP-forward†	**0.00**	**0.08**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	DP-SGD*†	100.00	95.61	98.01	100.00	92.64	92.21	100.00	82.89	86.97
	DualGuard	100.00	97.76	98.26	100.00	94.81	95.45	99.29	80.45	83.91
	AliasGuard	30.68	22.28	19.30	3.28	4.98	3.68	0.35	2.04	3.05
B	NoDef	92.04	80.05	66.69	76.84	57.14	34.20	48.41	35.03	25.46
	DP-forward†	94.03	100.00	96.60	96.93	100.00	100.00	89.93	100.00	100.00
	DP-SGD*†	98.55	99.83	97.93	99.18	100.00	100.00	98.94	100.00	100.00
	DualGuard	92.27	81.41	67.85	79.51	55.41	38.53	49.82	37.07	27.09
	AliasGuard	**32.82**	**26.36**	**21.87**	**3.07**	**4.55**	**4.98**	**1.59**	**1.02**	**0.41**
BF	NoDef	99.92	96.96	98.43	100.00	94.16	95.89	99.47	83.30	86.76
	DP-forward†	94.11	99.25	96.19	96.93	100.00	100.00	89.93	100.00	100.00
	DP-SGD*†	99.54	100.00	100.00	99.59	100.00	100.00	99.65	100.00	100.00
	DualGuard	100.00	97.28	98.59	100.00	93.94	95.02	99.65	84.32	86.76
	AliasGuard	**30.68**	**22.28**	**19.30**	**3.28**	**4.98**	**3.68**	**0.35**	**2.04**	**3.05**

The historical H4–1.5B record additionally evaluates a U decoder supplied with the selected surrogate sequences and labels. Private-token recovery is 32.4151% over 1,561 full-trace positions and 31.8910% over the 1,248-position evaluated subset. These results include the selected surrogate sequences and labels among the reconstruction inputs, alongside the original BF/HIGH results reported above.

Why both observation directions matter. The additional DP evaluations in Appendix B.3 separate the effects of the two perturbations. Across nine settings, DP-forward gives F recovery of 0–0.08% and B recovery of 89.93–100%. It changes smashed data but keeps the original next-token labels in the backward computation. DP-SGD* changes the exchanged gradient and keeps the original forward input; its F recovery is 82.89–100%. Thus each method still permits high recovery through one observation.

AliasGuard constructs both the input sequences and the labels from surrogates. The comparison above measures the resulting recovery; Appendix H.7 separately changes sequences and labels to test their contributions. For DP-forward, the two BF selection rules in Appendix B.3 can choose different candidates from the same recorded messages. The main table retains the higher DialogSum BF recovery, and the separate B results show the information available in the backward observations.

Recovery ratios in Table 1. For each task/model pair, we divide a baseline's BF recovery rate by AliasGuard's BF recovery rate. A ratio of three means that the baseline recovers three times the fraction of protected token positions recovered under AliasGuard. We calculate the ratios from the unrounded rates; Table 10 gives their ranges across the nine pairs. For the abstract and contribution summary, we first take the lowest BF recovery rate among the four baselines within each dataset–model pair. The relative reduction is $100 \times (1 - r_{\mathrm{AG}}/r_{\min})$, where r_{AG} is AliasGuard's BF recovery rate and $r_{\min}$ is that lowest baseline rate. Across the nine pairs, this gives 67.4–99.6%. The training and target conditions marked in Table 1 apply to these descriptive ratios, including its H4–Q1.5B target exception.

C.2 Prediction and generation quality

Task scores evaluate generated answers; recovery scores evaluate what can be inferred from the recorded training messages. These are different uses of the trained model. The Task column reproduces the main table at greater precision. C-PPL predicts reference-answer tokens using the correct preceding tokens, and Full PPL also scores prompt tokens (Appendix B.3).

Table 10: Baseline BF recovery divided by AliasGuard BF recovery within each task/model comparison. The final row first takes the lowest of the four baseline rates within each pair. The smallest ratios occur on H4/GPT-2 and the largest on GSM8K/GPT-2. On the latter, AliasGuard recovers 2 of 566 protected positions, compared with NoDef's 563, DP-forward's 509, DP-SGD*'s 564 and DualGuard's 564.

Baseline	Smallest ratio	Largest ratio
NoDef	3.26	281.50
DP-forward	3.07	254.50
DP-SGD*	3.24	282.00
DualGuard	3.26	282.00
Lowest of the four baselines	3.07	254.50

EOS is the percentage of generated answers that end with the model's end-of-sequence token. Cap is the percentage that reach the generation limit without that token. The Secondary column uses a further score for each task. For H4, the complete generated and reference answers must match after whitespace is standardized. For DialogSum, ROUGE-L uses the fixed first reference. For GSM8K, exact match additionally requires the recorded answer marker. H4 lexical F1 measures matching words; Appendix G.3 separately checks Python grammar and compilation.

Table 11: Additional final-checkpoint utility results for the 45 seed-3407 cases. N is evaluated task rows; Full PPL scores tokens throughout each sequence, including its input and reference answer. C-PPL scores only tokens in the reference answer of each sequence, including its private tokens. EOS and Cap are percentages ending normally and reaching the generation limit without EOS. Secondary is whitespace-normalized exact match (H4), single-reference ROUGE-L (DS), or strict ####-marker answer match (G8), in percent.

Data	Model	Method	N	Task	Full PPL	C-PPL	EOS	Cap	Secondary
H4	GPT-2	NoDef	2002	17.99	**5.153**	**4.472**	6.99	93.01	**0.00**
		DP-Fwd†	2002	1.17	1228.562	1047.366	0.00	100.00	**0.00**
		DP-SGD*†	2002	**19.30**	5.180	4.501	**10.94**	**89.06**	**0.00**
		DualGuard	2002	12.61	15.680	10.954	0.00	100.00	**0.00**
		AliasGuard	2002	18.73	5.350	4.676	2.65	97.35	**0.00**
H4	Q1.5B	NoDef	2003	**49.79**	**2.669**	**1.963**	**90.81**	**9.19**	**3.39**
		DP-Fwd†	2003	19.52	4.001	2.620	31.05	68.95	0.00
		DP-SGD*†	2003	44.74	2.704	1.990	90.51	9.49	2.45
		DualGuard	2003	26.02	18.308	4.800	10.13	89.87	0.10
		AliasGuard	2003	46.69	2.749	2.037	86.72	13.28	3.15
H4	Q3B	NoDef	2003	48.05	**2.702**	**1.967**	**91.91**	**8.09**	3.15
		DP-Fwd†	2003	23.80	3.442	2.436	19.32	80.68	0.15
		DP-SGD*†	2003	**48.80**	2.717	1.975	91.11	8.89	**3.79**
		DualGuard	2003	27.00	8.352	3.477	3.25	96.75	0.00
		AliasGuard	2003	45.55	2.765	2.043	88.92	11.08	2.55
DS	GPT-2	NoDef	498	34.45	8.123	5.821	**99.20**	**0.80**	28.04
		DP-Fwd†	498	5.51	57.159	74.345	0.00	100.00	4.51
		DP-SGD*†	498	33.73	8.127	5.870	97.99	2.01	27.81
		DualGuard	498	10.80	12.195	13.947	0.00	100.00	9.10
		AliasGuard	498	**35.00**	**8.118**	**5.785**	98.59	1.41	**28.58**
DS	Q1.5B	NoDef	500	41.19	6.423	**3.566**	**100.00**	**0.00**	**34.53**
		DP-Fwd†	500	0.03	139.191	145.472	0.00	100.00	0.02
		DP-SGD*†	500	41.22	**6.389**	3.618	**100.00**	**0.00**	33.82
		DualGuard	500	19.31	13.691	8.641	52.20	47.80	16.06
		AliasGuard	500	**41.57**	6.442	3.667	99.60	0.40	34.23

Continued on next page

Table 11 (continued)

Data	Model	Method	N	Task	Full PPL	C-PPL	EOS	Cap	Secondary
DS	Q3B	NoDef	500	**42.51**	**6.087**	**3.300**	**100.00**	**0.00**	35.08
		DP-Fwd†	500	23.67	9.646	7.827	49.20	50.80	19.10
		DP-SGD*†	500	42.45	6.171	3.362	**100.00**	**0.00**	**35.31**
		DualGuard	500	11.12	13.919	11.833	44.20	55.80	8.96
		AliasGuard	500	41.82	6.148	3.407	99.80	0.20	34.35
G8	GPT-2	NoDef	1319	1.74	**6.434**	**4.401**	**32.52**	**67.48**	0.76
		DP-Fwd†	1319	0.00	526.439	436.406	0.00	100.00	0.00
		DP-SGD*†	1319	2.12	6.500	4.466	27.60	72.40	**0.99**
		DualGuard	1319	1.52	22.874	22.971	0.00	100.00	0.00
		AliasGuard	1319	**2.27**	6.495	4.418	30.63	69.37	**0.99**
G8	Q1.5B	NoDef	1319	**36.62**	**2.909**	1.634	95.00	5.00	**36.39**
		DP-Fwd†	1319	1.29	36.329	18.889	0.00	100.00	0.00
		DP-SGD*†	1319	34.72	2.916	**1.632**	**97.42**	**2.58**	34.65
		DualGuard	1319	27.29	3.403	2.244	96.89	3.11	0.00
		AliasGuard	1319	31.31	2.935	1.667	93.71	6.29	31.08
G8	Q3B	NoDef	1319	**51.33**	**2.761**	**1.596**	98.94	1.06	**51.33**
		DP-Fwd†	1319	3.18	5.363	3.644	1.21	98.79	0.68
		DP-SGD*†	1319	42.61	2.768	1.618	96.21	3.79	42.61
		DualGuard	1319	18.12	3.325	2.262	94.84	5.16	0.00
		AliasGuard	1319	40.56	2.773	1.614	**99.32**	**0.68**	40.49

DualGuard uses the published warm-up, retention objectives, and default weights specified in Appendix B. We compare the complete methods using their F, B, and BF recovery and task scores.

On H4–1.5B, AliasGuard generates normal endings in 86.72% of sequences, compared with 10.13% for DualGuard, and has lower C-PPL (2.037 versus 4.800). These scores compare the complete deployed methods; individual training components are held at their respective configurations.

D Results Across Training Seeds and Additional Runs

This appendix separates three uses of repeated training. Appendix D.1 reports additional AliasGuard runs across nine task/model settings. Appendix D.2 compares AliasGuard and DualGuard under three matched seeds. Appendix D.3 compares AliasGuard and NoDef under those seeds, separating token prediction from final answers.

D.1 Additional AliasGuard runs at seed 3408

The nine seed-3408 AliasGuard cases use $K = 30$, $K' = 3$, trainable heads, and final updates 169 (Qwen) or 154 (GPT-2); privacy evaluation uses 128 training sequences. F has 200 iterations, B has 32, BF has 32 B iterations followed by 200 F iterations, and HIGH has the 96/400/400 stages defined in Appendix B.3. Tables 12 and 13 report the same metric definitions as the primary seed. These tables report AliasGuard alone, separately from the seed-3407 five-method comparison.

Private-token recovery remains below unprotected recovery in all four reconstruction methods for all nine cases. BF/HIGH protected recovery ranges from 0.8230% to 26.8745%, versus 94.9235–97.7478% at unprotected positions. This repeats the qualitative separation, while task scores do change. In GPT-2/1.5B/3B order, seed-3408 minus seed-3407 task differences are $-0.03/-3.91/-0.002$ F1 points on H4, $-3.03/+0.95/+0.71$ ROUGE-L points on DialogSum, and $-0.15/+1.52/+2.81$ numerical-answer accuracy points on GSM8K. Unrounded per-case differences remain in the source archive.

Several settings differ between these runs. Historical H4–1.5B used K' between four and six, whereas seed 3408 fixes it to three. Software versions and the training sequences selected for reconstruction also differ; each table retains its own denominators. Consequently, the observed differences

combine seed variation with these configuration changes. We report each seed and its configuration separately.

Table 12: AliasGuard reconstruction at seed 3408. P/O are private/unprotected token recovery percentages; each row uses 128 targets. Configuration differences from seed 3407 are specified in the text.

Data	Model	Method	F P/O	B P/O	BF P/O	HIGH P/O	N(P)	N(O)
H4	GPT-2	**AliasGuard**	26.87/97.61	18.78/92.74	26.87/97.61	26.87/97.61	1347	17765
H4	Q1.5B	**AliasGuard**	17.90/94.92	19.81/80.62	17.90/94.92	17.90/94.92	1257	12154
H4	Q3B	**AliasGuard**	16.39/95.11	17.82/74.71	16.39/95.11	16.39/95.11	1257	12154
DS	GPT-2	**AliasGuard**	4.87/97.75	5.30/93.97	4.87/97.75	4.87/97.75	472	28505
DS	Q1.5B	**AliasGuard**	2.08/97.30	3.54/88.20	2.08/97.30	2.08/97.30	480	26051
DS	Q3B	**AliasGuard**	1.25/97.50	1.67/83.89	1.25/97.50	1.25/97.50	480	26051
G8	GPT-2	**AliasGuard**	0.86/96.15	1.21/88.46	0.86/96.15	0.86/96.15	579	18484
G8	Q1.5B	**AliasGuard**	2.26/97.12	1.23/86.11	2.26/97.12	2.26/97.12	486	21793
G8	Q3B	**AliasGuard**	0.82/97.38	0.00/83.70	0.82/97.38	0.82/97.38	486	21793

Table 13: AliasGuard final-checkpoint utility at seed 3408. Metric definitions match Table 11. Higher task and EOS scores and lower PPL and Cap are better.

Data	Model	Method	n	Task	Full PPL	C-PPL	EOS	Cap	Secondary
H4	GPT-2	**AliasGuard**	2002	18.70	5.458	4.786	3.75	96.25	0.00
H4	Q1.5B	**AliasGuard**	2003	42.78	2.789	2.079	87.97	12.03	0.80
H4	Q3B	**AliasGuard**	2003	45.54	2.778	2.066	89.22	10.78	1.90
DS	GPT-2	**AliasGuard**	498	31.97	8.190	6.046	97.99	2.01	26.80
DS	Q1.5B	**AliasGuard**	500	42.52	6.343	3.603	99.60	0.40	35.65
DS	Q3B	**AliasGuard**	500	42.53	6.006	3.388	99.60	0.40	35.58
G8	GPT-2	**AliasGuard**	1319	2.12	6.452	4.395	42.38	57.62	0.99
G8	Q1.5B	**AliasGuard**	1319	32.83	2.969	1.650	83.62	16.38	32.37
G8	Q3B	**AliasGuard**	1319	43.37	2.868	1.636	93.10	6.90	43.06

D.2 Matched three-seed GSM8K comparison

This controlled study compares AliasGuard and DualGuard at training seeds 3407, 3409 and 3410, with three models and 18 newly trained checkpoints. Within each model, all six runs use the same training identities (169 for Qwen, 154 for GPT-2), input banks, software and hardware. Each seed changes the training order and method randomness; both methods share that seed's order. AliasGuard fixes $K = 30$ and $K' = 3$. DualGuard repeats its public preparation, warm-up and formal optimization at every seed using its unchanged method-specific settings. Qwen uses A100 and BF16; GPT-2 uses A40 and FP32. All runs use PyTorch 2.10.0, CUDA 12.8, SDPA, disabled TF32 and deterministic algorithms.

Common evaluation conditions. Every final checkpoint is evaluated on the same 1,319 official test questions, with greedy generation capped at 512 tokens and the same numerical-answer extraction. For reconstruction, fresh observations are computed on 128 fixed target sequences per tokenizer at the final checkpoint, without optimizer updates; parameter hashes are checked before and after capture. The protected denominators are 491 tokens for each Qwen model and 566 for GPT-2. Capture randomness and the client candidate schedule are fixed across training seeds. BF uses B32 followed by F200 and reconstruction seed 9107. Public-context proposals and public decoders are available; the actual selected target surrogates, target labels and private client weights are withheld. Public decoders use the same 819/205 fit/validation split and are fitted once per model and method, then reused across the three training seeds. These final-checkpoint observations are reported separately from the historical training-time observations in Table 1(a) and the seed-3408 results above.

Variation across training seeds. Table 1(b) reports the arithmetic mean and sample standard deviation (SD) across the three independently trained models. For scores x_1, x_2, x_3, the mean is

$\bar{x} = (x_1 + x_2 + x_3)/3$ and SD is $\sqrt{\sum_{i=1}^{3}(x_i - \bar{x})^2/2}$. These quantities describe changes across training seeds. They are calculated from unrounded scores. Per-run values and paired differences appear in Tables 14 and 15. AliasGuard lowers BF private-token recovery in all nine matched comparisons. Its recovery is 0.815% for both Qwen models at every seed and 0.883–1.237% for GPT-2; DualGuard recovers 81.670–81.874%, 90.020–90.428% and 100%, respectively. AliasGuard has higher numerical-answer accuracy in all six Qwen comparisons: by 8.112–17.665 percentage points for Q1.5B and 15.997–25.246 points for Q3B. GPT-2 has the same mean accuracy for both methods (1.440%); its seed-level differences range from -0.531 to $+0.303$ points. AliasGuard has lower C-PPL in all nine comparisons.

Table 14: All 18 final-checkpoint results in the matched study. Answer counts use 1,319 questions; BF uses 128 sequences. The last column gives recovered private tokens over the full protected-token denominator.

Model	Method	Seed	Correct answers ↑	C-PPL ↓	BF (%) ↓	Correct / total
GPT-2	AliasGuard	3407	13/1,319	**4.526**	**0.883**	5/566
GPT-2	DualGuard	3407	**20/1,319**	22.969	100.000	566/566
GPT-2	AliasGuard	3409	**23/1,319**	**4.477**	**1.060**	6/566
GPT-2	DualGuard	3409	19/1,319	22.581	100.000	566/566
GPT-2	AliasGuard	3410	**21/1,319**	**4.446**	**1.237**	7/566
GPT-2	DualGuard	3410	18/1,319	22.475	100.000	566/566
Q1.5B	AliasGuard	3407	**542/1,319**	**1.645**	**0.815**	4/491
Q1.5B	DualGuard	3407	309/1,319	2.278	81.670	401/491
Q1.5B	AliasGuard	3409	**468/1,319**	**1.652**	**0.815**	4/491
Q1.5B	DualGuard	3409	354/1,319	2.252	81.874	402/491
Q1.5B	AliasGuard	3410	**504/1,319**	**1.664**	**0.815**	4/491
Q1.5B	DualGuard	3410	397/1,319	2.194	81.874	402/491
Q3B	AliasGuard	3407	**542/1,319**	**1.624**	**0.815**	4/491
Q3B	DualGuard	3407	209/1,319	2.308	90.224	443/491
Q3B	AliasGuard	3409	**442/1,319**	**1.632**	**0.815**	4/491
Q3B	DualGuard	3409	231/1,319	2.254	90.020	442/491
Q3B	AliasGuard	3410	**468/1,319**	**1.635**	**0.815**	4/491
Q3B	DualGuard	3410	183/1,319	2.329	90.428	444/491

Paired evaluation uncertainty. For each trained pair, we draw 1,319 questions with replacement from the 1,319 test questions, or 128 complete targets with replacement from the 128 reconstruction targets. Both methods use the same drawn IDs. We repeat this 2,000 times at resampling seed 9107; this procedure is called paired bootstrap resampling. C-PPL is recomputed from the resampled summed negative log probabilities and the count of scored tokens in the reference answers of those sequences; recovery is recomputed from summed correct and protected-token counts. We sort the resulting differences and report their 2.5th and 97.5th percentiles as a 95% interval. This describes variation across evaluation sequences for each fixed trained pair; the SD in Table 1(b) describes variation across training seeds. The Qwen accuracy differences and every recovery reduction have intervals above zero. GPT-2 accuracy intervals include zero, while all C-PPL differences favor AliasGuard.

Table 15: Paired differences [95% bootstrap interval]. Δ answer accuracy and ΔC-PPL are AliasGuard minus DualGuard; recovery reduction is DualGuard minus AliasGuard, in percentage points (pp).

Model	Seed	Answer accuracy change (pp)	ΔC-PPL	BF reduction (pp)
GPT-2	3407	-0.531 [-1.289, +0.227]	-18.443 [-19.209, -17.747]	+99.117 [+97.454, +100.000]
GPT-2	3409	+0.303 [-0.455, +1.061]	-18.104 [-18.863, -17.421]	+98.940 [+97.207, +100.000]
GPT-2	3410	+0.227 [-0.531, +0.910]	-18.028 [-18.783, -17.349]	+98.763 [+97.043, +99.842]
Q1.5B	3407	+17.665 [+14.481, +20.925]	-0.633 [-0.652, -0.615]	+80.855 [+75.236, +86.025]
Q1.5B	3409	+8.643 [+5.534, +11.751]	-0.600 [-0.618, -0.583]	+81.059 [+75.365, +86.299]
Q1.5B	3410	+8.112 [+5.231, +11.298]	-0.530 [-0.547, -0.514]	+81.059 [+75.585, +86.221]
Q3B	3407	+25.246 [+22.136, +28.506]	-0.684 [-0.707, -0.662]	+89.409 [+85.144, +93.241]

Model	Seed	Answer accuracy change (pp)	ΔC-PPL	BF reduction (pp)
Q3B	3409	+15.997 [+12.813, +19.181]	-0.622 [-0.643, -0.603]	+89.206 [+84.699, +93.192]
Q3B	3410	+21.607 [+18.499, +24.792]	-0.694 [-0.717, -0.673]	+89.613 [+85.408, +93.392]

D.3 GSM8K RESULTS ACROSS THREE TRAINING SEEDS

For each Qwen model and seed, NoDef and AliasGuard use the same ordered 169 training sequences and are scored at the final checkpoint. The seed-3407 models in this study use a different sequence order and surrogate bank from the seed-3407 models in Table 1(a); their scores are reported separately. The six AliasGuard checkpoint hashes match those in Appendix D.2. Each pair is evaluated on all 1,319 original GSM8K test questions. Exact private-token positions are recorded for 591 of those questions. U-Top1 counts correct most-likely predictions at unprotected positions; private-position Top-1 counts them at private positions. Both use the correct preceding tokens and only this 591-question subset. For the remaining 728 questions, private-token positions are unknown. All of these questions still count when numerical-answer accuracy is computed on the complete test set. All six paired official evaluations and twelve positional scoring records passed row-ID, checkpoint, denominator, and hash checks.

Table 16: Matched NoDef and AliasGuard results in the separate GSM8K three-seed study. U-Top1 uses 72,755 recorded unprotected-token positions in the reference answers of the 591 sequences per model–seed pair; private-position Top-1 uses 1,146 registered private positions. These are reference-token prediction metrics, not private-token reconstruction from communicated data. Correct answers use the complete 1,319-question official test set.

Model	Seed	U-Top1 (%) NoDef / AliasGuard	Private-position Top-1 (%) NoDef / AliasGuard	Correct / 1,319 NoDef / AliasGuard
Qwen2.5-1.5B	3407	86.750 / 86.523	65.445 / 77.749	507 / 542
	3409	86.673 / 86.380	61.169 / 77.487	499 / 468
	3410	86.944 / 86.408	66.230 / 78.534	552 / 504
Qwen2.5-3B	3407	86.679 / 86.978	65.620 / 75.567	455 / 542
	3409	87.194 / 86.679	59.860 / 69.895	676 / 442
	3410	87.411 / 86.644	65.445 / 78.098	630 / 468

Across all six pairs, U-Top1 stays within 0.77 percentage points of NoDef: the differences range from -0.767 to $+0.300$. At seed 3407, AliasGuard answers 35 more questions correctly with Qwen2.5-1.5B and 87 more with Qwen2.5-3B, gains of 2.65 and 6.60 percentage points. Private-position Top-1 is also higher for AliasGuard in every pair. The complete correct-answer differences range from -17.741 to $+6.596$ points across seeds. Across all seeds, the unprotected-token prediction differences are small under the same correct preceding tokens. Final answer accuracy varies more: each generated answer uses the model's own previous predictions. These are two distinct measurements of the same trained models. The results support close unprotected-token prediction and show task-score gains for some model–seed pairs; the table reports all six pairs.

For an additional fixed-checkpoint check, each model answers the same 111 original questions and three predetermined name replacements per question. One replacement leaves the name unchanged, giving 332 changed-name prompts per pair. AliasGuard minus NoDef changed-name accuracy, in percentage points, is:

Model	Seed 3407	Seed 3409	Seed 3410
Qwen2.5-1.5B	-1.506	-7.530	-4.819
Qwen2.5-3B	$+0.602$	-23.795	-21.386

These replacements occur only at evaluation time. They measure how changing names affects answers from the already trained models; the training sequences remain those specified above.

E RECONSTRUCTION FROM COMPLETE CANDIDATE SEQUENCES

This appendix asks how the recovery comparison changes when the reconstruction procedure changes. We examine three choices: how many candidate sequences it evaluates, which information it receives, and how many refinement iterations it performs. These comparisons complement the five-method results in Table 1(a). Each score below comes from one complete procedure with specified inputs and a fixed rule for selecting its output.

E.1 TARGETS, PERMITTED INFORMATION, AND SCORING

The study evaluates H4–Qwen2.5-1.5B, DialogSum–Qwen2.5-1.5B, GSM8K–Qwen2.5-1.5B, and H4–GPT-2 small. Each configuration has 128 fixed reconstruction targets. Their private-token denominators, in this order, are 1,207, 462, 491, and 1,307.

A candidate sequence fills all private-token positions and retains the unprotected tokens. The program saves its selected candidate before a separate scorer reads the original private tokens. The evaluator receives C and the public auxiliary data. F also receives smashed data (Z' for AliasGuard) and trunk outputs U; BF additionally receives the gradient Δ. The recorded field `G_U` contains that gradient after the implementation's scaling, clipping, and numerical conversion. Private client weights and the target's selected surrogates and labels are withheld in this comparison.

Fitted decoder. We evaluate H4–Qwen2.5-1.5B, DialogSum–Qwen2.5-1.5B, GSM8K–Qwen2.5-1.5B, and H4–GPT-2 small on 128 existing targets each. The decoder first transforms each observed input into 512 numbers with a GELU nonlinear layer. It then projects them to the public model's token embedding size. A fixed embedding matrix and a learned bias produce scores for all eligible vocabulary tokens. Public validation selects the decoder from the states saved after epochs 2, 5, and 10. The original public protected tokens supply the training labels. Public AliasGuard observations use shared surrogate labels; original private target labels are not inserted into that simulation. The actual public fit/validation counts are 819/205 for H4–Qwen and GSM8K, 669/167 for DialogSum, and 747/187 for the historical H4–GPT-2 manifest.

Budget and scoring. Each complete candidate sequence receives two scores. The first averages the logarithms of the decoder probabilities assigned to its proposed private tokens. The second averages the logarithms of the public LM's next-token probabilities for the complete sequence. The combined score adds the first score to a coefficient times the second. The coefficient is selected from $\{0, 0.1, 0.3, 1\}$ using public validation.

At coefficient zero, the decoder alone determines the score, and private tokens can be selected independently. With a nonzero coefficient, the public LM also evaluates how the tokens fit together. When validation prefers zero, that complete decoder-only result is retained; the nonzero-coefficient procedure is separately used to examine the candidate budget. Neither procedure selects settings using the private target labels. The snapshots contain 16, 64, and 256 distinct complete assignments from nested sets of candidate sequences. The initial NoDef and DualGuard comparison uses 16 and 64; the additional H4 comparison also evaluates 256. F reads Z and U; BF additionally reads gradients. Public checks verify alignment between a predicted token and its next-token label, whether candidate changes affect the score, the effect of shuffling observations with the same array dimensions, and the calculation with U held fixed. Recomputing the saved scores with the fixed decoder reproduces all 458,752 candidate log-probability values exactly.

Inference-budget result. For BF, the high-minus-medium changes are $+0.414$, 0, -0.407, and 0 percentage points in the configuration order above. The measured changes are within 0.414 points across the four settings. Appendix E.4 reports their intervals and the prespecified one-point criterion. In the zero-change groups, the set of correctly reconstructed protected tokens stays unchanged. Their zero-width bootstrap intervals describe these fixed evaluated targets. The bootstrap resamples original sequences jointly across budgets 10,000 times with seed 20260922. The four primary upper bounds use $\alpha = 0.05/4$; other input comparisons are reported separately.

Highest recovery among the evaluated attacks. The highest AliasGuard recovery is 487/1,207 protected tokens on H4–Qwen and 639/1,307 on H4–GPT-2. Both values come from F and are in-

Table 17: BF recovery as the number of candidate sequences increases. Recovery is a percentage of correctly reconstructed private-token positions. Changes and one-sided 98.75% upper bounds use percentage points. Each budget counts distinct complete candidate sequences.

Configuration	16	64	256	256-64	Upper bound
H4–Q1.5B	30.489 (368/1207)	32.477 (392/1207)	32.891 (397/1207)	+0.414	1.635
DialogSum–Q1.5B	6.277 (29/462)	6.277 (29/462)	6.277 (29/462)	+0.000	0.000
GSM8K–Q1.5B	4.888 (24/491)	5.092 (25/491)	4.684 (23/491)	-0.407	0.000
H4–GPT2	47.131 (616/1307)	47.131 (616/1307)	47.131 (616/1307)	+0.000	0.000

cluded together with the BF results. The lower corresponding NoDef/DualGuard rates are 96.603% and 100%, giving ratios of 2.394 and 2.045. DialogSum and GSM8K yield AliasGuard recovery of 29/462 and 25/491, versus lower corresponding rates of 91.126% and 80.652%. Each value is the highest aggregate recovery of one complete evaluated attack under matching information, target, and observation conditions; values are not combined token by token across attacks. Historical H4–Qwen decoding with the selected surrogate sequences and labels supplied remains separate: 398/1,248 on its selected subset and 506/1,561 on the full trace.

Which head and surrogate sequences are used. Public simulations use three surrogate sequences and a public pretrained head. The historical H4–Qwen observations use four to six selected surrogate sequences and the head state at the recorded update. These computations are distinguished in the reported comparisons. For DualGuard on DialogSum, the summary retains the stronger existing BF result of 426/462 under the same information conditions, together with the new 29/462 result in the detailed table.

E.2 Nested candidate budgets

Here the budget counts distinct candidate sequences, not optimization iterations. The 64-candidate set contains the 16-candidate set, and the 256-candidate set contains the 64-candidate set. Repeated candidate sequences are removed before counting.

The decoder and score coefficient remain fixed as the budget increases. When the coefficient is zero, each private-token prediction can be chosen independently from the decoder probabilities. A nonzero coefficient also scores how well the proposed tokens fit together in the complete sequence. The budget curve evaluates this latter procedure. We report the two procedures separately and use each procedure's complete output.

E.3 Public surrogate computation and recorded messages

This comparison keeps the fitted F decoder and the budget of 256 candidate sequences fixed. One input is computed without the target's recorded messages: three surrogate sequences are independently constructed from public information and passed through the public pretrained head. The other input is the actual forward computation recorded during training. Both are scored by the same decoder. The comparison tests whether the way forward inputs are produced affects this finite decoder.

Table 18: F private-token recovery with the same fitted decoder and 256 candidate sequences. Public surrogates use three independently prepared sequences and the public pretrained head. Recorded messages use the client's selected sequences and head state. Each row has 128 targets; fractions give correct private-token positions over all evaluated private-token positions.

Dataset (Qwen2.5-1.5B)	Public surrogates	Recorded messages
H4	37.03 (447/1207)	40.35 (487/1207)
DialogSum	4.76 (22/462)	5.41 (25/462)
GSM8K	2.44 (12/491)	2.85 (14/491)

On H4, the public-surrogate input recovers 447/1,207 positions and the recorded-message input recovers 487/1,207. The difference is 40 correct positions, or 3.31 percentage points. An exploratory paired 95% interval is $[-0.35, 6.88]$, obtained by drawing complete sequences with replacement 10,000 times and recomputing the difference.

The two inputs are produced differently. None of the 384 public surrogate sequences matches the corresponding selected sequence in the formal inputs; all 128 ordered triples differ. The public computation also uses the pretrained head, whereas the recorded computation uses the head state at the training update. Thus this comparison changes surrogate sequences and head parameters together. It has not separated their contributions to the measured difference. Appendix F tests the theorems' specific comparison: change private tokens while keeping C and the other computation inputs fixed.

E.4 Equal-budget and highest-observed comparisons

At the same 64-candidate budget, AliasGuard has lower BF recovery than both NoDef and DualGuard in three of four settings. On DialogSum, it recovers fewer tokens than NoDef and ties DualGuard at 29/462. Table 20 evaluates both F and BF for all three H4 methods at 256 candidates using their frozen public reconstruction models and the same scoring rule. The DP baselines use the separately specified iteration-based reconstruction in Appendix C. Table 1(c) selects the highest recovery from complete evaluated programs within each compatible target and information condition. It therefore preserves earlier attacks that recover more from some baselines. Budgets are specific to each method. Each reported rate comes from one complete reconstruction procedure, with its target and information conditions fixed.

Table 19: BF private-token recovery using the same candidate construction and scoring rule, with 64 candidate sequences for each of the three methods. The last column separately gives AliasGuard recovery at 256 candidates.

Configuration	NoDef 64	AliasGuard 64	DualGuard 64	AliasGuard 256
H4–Q1.5B	85.501 (1032/1207)	32.477 (392/1207)	76.305 (921/1207)	32.891 (397/1207)
DialogSum–Q1.5B	63.420 (293/462)	6.277 (29/462)	6.277 (29/462)	6.277 (29/462)
GSM8K–Q1.5B	43.788 (215/491)	5.092 (25/491)	20.367 (100/491)	4.684 (23/491)
H4–GPT2	90.972 (1189/1307)	47.131 (616/1307)	79.725 (1042/1307)	47.131 (616/1307)

Table 20: H4 recovery (%) with 256 complete candidates per method and view on the same 128 sequences. Counts show correctly recovered protected tokens over all protected positions. The BF 64-candidate predictions were reproduced exactly before extension.

Model	View	NoDef	AliasGuard	DualGuard
Qwen2.5-1.5B	F	85.92 (1037/1207)	**40.35 (487/1207)**	86.25 (1041/1207)
	BF	85.50 (1032/1207)	**32.89 (397/1207)**	75.48 (911/1207)
GPT-2 small	F	91.28 (1193/1307)	**48.89 (639/1307)**	91.05 (1190/1307)
	BF	90.44 (1182/1307)	**47.13 (616/1307)**	79.72 (1042/1307)

For H4–Qwen2.5-1.5B, increasing the candidate count from 64 to 256 changes BF recovery by 0.414 percentage points. Its paired 95% interval is $[-0.584, 1.490]$ points. We obtain these intervals by drawing complete original sequences with replacement 10,000 times at seed 20260922. For each draw, all private-token occurrences in a sequence stay together, and recovery is recomputed from the summed correct and total counts.

The prespecified check asks whether the increase is below one percentage point. It uses a one-sided 98.75% upper bound for each of four settings, allocating $\alpha = 0.05/4$ to each. H4–Qwen2.5-1.5B has an upper bound of 1.635 points, so this check does not establish the one-point criterion there. DialogSum and H4–GPT-2 have identical correctness outcomes at both budgets, giving zero-width intervals on these evaluated targets. GSM8K's observed BF recovery decreases by 0.407 points.

Table 21: Programs and exact counts underlying Table 1(c). Each row uses 128 sequences. Counts are correctly reconstructed private tokens divided by their total number. The baseline is the lower of NoDef's and DualGuard's highest evaluated recovery rates. F256 uses 256 complete candidates; BF16 and BF64 use 16 and 64. Historical BF budgets appear in Appendix E.8.

Setting	AliasGuard	Lower baseline	Baseline count
H4/Q1.5B	F256: 487/1,207	DualGuard, BF	1,166/1,207
DS/Q1.5B	BF16/64/256: 29/462	NoDef, BF	421/462
G8/Q1.5B	BF64: 25/491	DualGuard, BF	396/491
H4/GPT-2	F256: 639/1,307	NoDef, F	1,307/1,307

E.5 Target sequences, observation updates, and permitted information

The supplementary audit uses Qwen2.5-1.5B, training seed 3407 and reconstruction seed 9107 on 128 targets per dataset. Paired intervals are unadjusted for multiple comparisons and describe one trained model. It reuses frozen communications and public pretrained weights, without retraining a client model. Predictions are committed before a separate CPU process opens the original private tokens. Recovery is the number of correctly reconstructed private token occurrences divided by their total count; exact-datum recovery requires every token in the recorded span to match. All repeated occurrences within a sequence stay together during resampling.

Which H4 sequences and updates are compared. The fixed DP target list contains 128 sequences, 1,207 private token occurrences and 861 spans. Historical NoDef/AliasGuard/DualGuard reconstruction contains 128 sequences, 1,248 occurrences and 899 spans: 102 sequence IDs are shared and 26 sequences in the DP list were not evaluated historically. All sequences in the DP list exist in the historical 169-sequence training set, but their recorded updates differ in 128/128 cases because the training orders differ. The new controls evaluate exactly these DP sequence IDs and positions at their *actual historical recorded updates*; the DP rows retain their own original updates. The new diagnostic thus aligns target identities and positions while retaining the different recorded updates. Table 1(a) retains the historical populations and values. DialogSum and GSM8K match all 128 IDs, protected spans and recorded updates across the five methods.

Permitted inputs. Condition A withholds the client's selected surrogate sequences and surrogate labels. Its AliasGuard simulator uses public-context proposals, the public head and shared candidate labels. Condition B additionally receives the actual selected sequences and their next-token labels, from which the surrogate labels are computed. Condition A uses an independently constructed public proxy for the client's surrogate distribution. Neither condition reads private head/tail weights or the client's random state. NoDef and DualGuard receive unprotected context at both the proposal and objective stages; DualGuard uses its public architecture and reciprocal-reference loss. DP rows reuse their previously completed context-conditioned evaluators, with their own calibrated candidate selection. Every method receives the unprotected context and is denied private client parameters. Each method retains its adaptive reconstruction algorithm and its specified budget. Table 22 shows these inputs explicitly.

E.6 What determines the decoder predictions

Giving the evaluator the selected surrogates. Condition A withholds the target's selected surrogate sequences and labels; condition B supplies them. All 384 complete predicted sequences are identical between A and B, so recovery differences and their paired intervals are zero.

In this evaluator, changing a proposed private token does not change the surrogate sequences used by its simulator or the surrogate-label loss. The objective used to refine that proposed token therefore has no gradient with which to change it. The final tokens remain the decoder's initial predictions. Supplying the actual selected surrogates changes the simulated arrays, but does not change these predictions. The following test checks that explanation directly.

Table 22: Information used by the finite evaluators. A and B use public head/tail proxies; the server state is permitted but unnecessary for their F/B/BF routines once the recorded observations are fixed. Historical evaluators remain separate from the comparison with the same unprotected context.

Information	New A	Historical DP	Historical AG	New B
Public pretrained model, algorithm, auxiliary data	Yes	Yes	Yes	Yes
Forward/backward observations used by F, B, or BF	Yes	Yes	Yes	Yes
Unprotected tokens supplied	Yes	Yes	No	Yes
Original private tokens or client head/tail weights	No	No	No	No
Selected AliasGuard surrogates and surrogate labels	No	N/A	Yes	Yes
Client random state	No	No	No	No

Removing surrogate inputs and gradients. We further isolate inference inputs on all 384 targets, using Qwen2.5-1.5B at seed 3407 in each dataset. A trusted CPU export retains only the frozen Z', U tensors, unprotected tokens, protected-position layout and public decoder references. A separate GPU process reads this restricted package and the original frozen public decoders; it cannot read the original communication containers, gradients, client prompts, actual or public surrogate sequences, shared labels, or private client checkpoints. Tensor equality and file hashes are checked before inference, and predictions are sealed before a separate CPU scorer opens the original private tokens. Direct Z' decoding reproduces every F and BF sequence; direct U decoding reproduces every B sequence (Table 23). The equality identifies which computation produced these predictions: the fitted decoders applied to Z' and U. The observed gradients affect the recorded matching loss but do not change the selected tokens in this implementation. Setting the decoder input to zero instead gives zero private-token recovery, showing that its learned bias alone does not produce the recorded correct predictions.

Table 23: Surrogate-free and gradient-free inference with the same frozen public decoders. Each match is an entire predicted sequence, including all protected occurrences, not merely an aggregate score. Across datasets and views, 1,152/1,152 sequences agree. Setting the decoder's smashed-data or trunk-output input to zero instead gives 0% private-token recovery for both decoders in every dataset. This zero-input ablation isolates decoder bias.

Dataset	Z' only (%)	U only (%)	Exact matches to F / B / BF
H4	23.20	26.18	128 / 128 / 128
DialogSum	4.98	4.55	128 / 128 / 128
GSM8K	2.04	1.02	128 / 128 / 128

The decoder has learned from public sequences. During fitting, it receives smashed data or trunk outputs simulated from those sequences and predicts their original tokens. Public validation selects its settings. H4 and GSM8K use 819 fitting and 205 validation sequences; DialogSum uses 669 and 167. AliasGuard's public simulation constructs surrogate sequences from the public context and uses their surrogate labels.

The resulting decoder can favor tokens that fit the unprotected context and patterns in public data. For example, code that copies an array may make a particular identifier likely. This explains how the decoder can make correct guesses even when changing the true private tokens leaves the messages unchanged. Appendix F states that message-equality argument explicitly. The decoder measurements here identify its actual computation; they do not by themselves measure a probability distribution over all possible private tokens.

E.7 SUPPLYING THE SAME UNPROTECTED CONTEXT TO FIVE METHODS

With the unprotected context supplied to every method and private client parameters withheld, AliasGuard has the lowest BF recovery in all three Qwen2.5-1.5B comparisons: 23.20% on H4, 4.98% on DialogSum, and 2.04% on GSM8K. Thus its lower recovery remains visible under this common information condition. H4 aligns the 128 sequence identities and positions but retains each run's recorded updates; DialogSum and GSM8K also align those updates. The table captions specify these conditions, and Appendix E.5 gives the target correspondence.

Table 24: BF private-token recovery (%) with unprotected context supplied and private client parameters withheld. DP values reuse sealed records; NoDef/DualGuard/AliasGuard are newly evaluated. Algorithms and budgets remain defense-specific. ‡: H4 has the same 128 IDs, 1,207 protected positions and spans, but DP and historical methods observe them at different updates. Thus only DialogSum and GSM8K meet the exact chronology condition. AliasGuard is highlighted.

Dataset	NoDef	DualGuard	DP-forward	DP-SGD*	AliasGuard A
H4‡	96.77	96.60	99.25	100.00	**23.20**
DialogSum	91.13	92.21	100.00	100.00	**4.98**
GSM8K	81.87	80.65	100.00	100.00	**2.04**

Table 25: Paired sequence-bootstrap intervals for the comparisons with the same unprotected context, using 2,000 draws. H4–DP pairs are excluded because recorded updates differ. These are single-model, unadjusted intervals; reconstruction algorithms and compute budgets remain different.

Dataset	Baseline	AliasGuard A minus baseline (pp), 95% CI
H4	NoDef	-73.57 [-78.75, -68.17]
H4	DualGuard	-73.41 [-78.54, -68.05]
DialogSum	NoDef	-86.15 [-90.67, -81.26]
DialogSum	DualGuard	-87.23 [-91.32, -82.65]
DialogSum	DP-forward	-95.02 [-97.51, -92.18]
DialogSum	DP-SGD*	-95.02 [-97.51, -92.18]
GSM8K	NoDef	-79.84 [-84.95, -74.58]
GSM8K	DualGuard	-78.62 [-84.11, -72.73]
GSM8K	DP-forward	-97.96 [-99.57, -96.14]
GSM8K	DP-SGD*	-97.96 [-99.57, -96.14]

E.8 INCREASING THE RECONSTRUCTION ITERATIONS

Increasing the registered BF budget to HIGH leaves AliasGuard's complete predictions unchanged in all nine task/model settings: all 1,152 sequence pairs match. In the same table, HIGH increases NoDef and DualGuard recovery in several settings. The following implementation check explains why the additional iterations leave this AliasGuard procedure unchanged.

We verified 13,824 archived prediction records in 108 method/view groups, each containing 128 targets, against the recorded scores. Another 222 records were identical resume duplicates and were counted once. NoDef, DualGuard and AliasGuard use BF stages B32+F200; HIGH registers B96, F400 and a 400-step joint stage. DP BF instead includes B32, independent F200, B-initialized F200 and discrete calibrated selection. The registered budgets above specify each method's computation. No new HIGH reconstruction or DP HIGH measurement was run. DP BF already recovers 100% on both natural-language audit cases; on the fixed H4 targets the lowest DP BF recovery is 99.25%, so none of these three cases meets the registered BF-below-95% trigger for an optional DP HIGH run.

The historical AliasGuard simulator uses the fixed recorded forward array and a surrogate-label loss. Changing the proposed private tokens does not change either input, so repeating its refinement steps leaves the proposal unchanged. Conditions A and B instead recompute surrogate head outputs with

Table 26: Archived private-token recovery (%). ‡: the historical H4–1.5B population has 1,248 positions, not the 1,207-position population used for the new controls. AliasGuard BF and HIGH have identical full predictions on all 1,152 sequence pairs; both the maximum and mean HIGH-minus-BF difference are exactly zero.

		NoDef		DualGuard		AliasGuard	
Dataset	Model	BF	HIGH	BF	HIGH	BF	HIGH
H4	GPT-2	99.92	100.00	100.00	100.00	30.68	30.68
H4 ‡	Qwen-1.5B	96.96	99.76	97.28	99.60	22.28	22.28
H4	Qwen-3B	98.43	100.00	98.59	99.83	19.30	19.30
DialogSum	GPT-2	100.00	100.00	100.00	100.00	3.28	3.28
DialogSum	Qwen-1.5B	94.16	98.92	93.94	98.48	4.98	4.98
DialogSum	Qwen-3B	95.89	100.00	95.02	99.57	3.68	3.68
GSM8K	GPT-2	99.47	100.00	99.65	100.00	0.35	0.35
GSM8K	Qwen-1.5B	83.30	93.28	84.32	95.11	2.04	2.04
GSM8K	Qwen-3B	86.76	98.17	86.76	97.56	3.05	3.05

public model weights. Their surrogate inputs and labels also remain independent of the proposed private tokens.

For each dataset and condition, we selected five targets spanning sequence lengths. We ran both the full registered loops and the implementation that skips these unchanged updates. Their predictions and every recorded objective value agree exactly. This verifies the acceleration for these specified programs and explains their identical BF/HIGH predictions. The complete-candidate comparisons earlier in this appendix test a different change: they expand the candidate set itself.

E.9 Paired uncertainty and scope of the baseline ratio

For archived breadth results, we resample original sequences 2,000 times (seed 9107), carrying all their token occurrences together. Thirty-four method comparisons have matching 128 target identities, positions and recorded updates. The two H4–1.5B comparisons with DP do not, and receive no paired interval. Each comparison retains its historical information conditions. Among the eight fully time-aligned dataset/model settings, the minimum of the lowest-baseline/AliasGuard BF ratios is 3.06733167: H4–GPT-2, DP-forward 94.10864575% divided by AliasGuard 30.68094874%. This descriptive point estimate summarizes the eight aligned settings.

Table 27: Archived paired BF recovery difference: AliasGuard minus baseline, in percentage points, with 95% sequence-bootstrap interval. Negative values favor AliasGuard. This table quantifies sampling variation under the information conditions of each historical evaluation.

Dataset	Model	Baseline	Difference [95% CI]
H4	GPT-2	NoDef	-69.24 [-72.90, -65.57]
H4	GPT-2	DualGuard	-69.32 [-73.08, -65.63]
H4	GPT-2	DP-forward	-63.43 [-67.55, -59.40]
H4	GPT-2	DP-SGD*	-68.86 [-72.60, -65.20]
H4	Qwen-1.5B	NoDef	-74.68 [-79.34, -69.51]
H4	Qwen-1.5B	DualGuard	-75.00 [-79.53, -70.07]
H4	Qwen-3B	NoDef	-79.12 [-83.27, -74.61]
H4	Qwen-3B	DualGuard	-79.29 [-83.49, -74.72]
H4	Qwen-3B	DP-forward	-76.88 [-81.60, -71.77]
H4	Qwen-3B	DP-SGD*	-80.70 [-84.88, -76.05]
DialogSum	GPT-2	NoDef	-96.72 [-98.92, -93.90]
DialogSum	GPT-2	DualGuard	-96.72 [-98.92, -93.90]
DialogSum	GPT-2	DP-forward	-93.65 [-97.76, -88.05]
DialogSum	GPT-2	DP-SGD*	-96.31 [-98.86, -93.07]
DialogSum	Qwen-1.5B	NoDef	-89.18 [-93.13, -84.42]

Dataset	Model	Baseline	Difference [95% CI]
DialogSum	Qwen-1.5B	DualGuard	-88.96 [-92.77, -84.58]
DialogSum	Qwen-1.5B	DP-forward	-95.02 [-97.51, -92.18]
DialogSum	Qwen-1.5B	DP-SGD*	-95.02 [-97.51, -92.18]
DialogSum	Qwen-3B	NoDef	-92.21 [-95.13, -88.91]
DialogSum	Qwen-3B	DualGuard	-91.34 [-94.76, -87.34]
DialogSum	Qwen-3B	DP-forward	-96.32 [-98.35, -94.07]
DialogSum	Qwen-3B	DP-SGD*	-96.32 [-98.35, -94.07]
GSM8K	GPT-2	NoDef	-99.12 [-99.82, -98.26]
GSM8K	GPT-2	DualGuard	-99.29 [-100.00, -98.31]
GSM8K	GPT-2	DP-forward	-89.58 [-97.18, -79.08]
GSM8K	GPT-2	DP-SGD*	-99.29 [-99.83, -98.58]
GSM8K	Qwen-1.5B	NoDef	-81.26 [-85.99, -76.32]
GSM8K	Qwen-1.5B	DualGuard	-82.28 [-87.28, -76.91]
GSM8K	Qwen-1.5B	DP-forward	-97.96 [-99.57, -96.14]
GSM8K	Qwen-1.5B	DP-SGD*	-97.96 [-99.57, -96.14]
GSM8K	Qwen-3B	NoDef	-83.71 [-87.84, -79.35]
GSM8K	Qwen-3B	DualGuard	-83.71 [-88.26, -78.72]
GSM8K	Qwen-3B	DP-forward	-96.95 [-98.76, -94.64]
GSM8K	Qwen-3B	DP-SGD*	-96.95 [-98.76, -94.64]

E.10 Additional implementation checks

Input preservation and training reproducibility. We checked the actual static inputs before training. For H4, DialogSum, and GSM8K, under both GPT-2 and Qwen tokenization, we reconstructed the permitted context from each formal original sequence and from each of its three selected surrogate sequences. All 2,907 original–surrogate pairs gave the same complete context, selected bank key, and unprotected token IDs; the M4 loader read no original sequence. Two H4/GPT-2 selected sequences happened to reproduce the original text, which the protocol permits. In a separate full-trajectory check, we loaded only the sealed context-derived M4 input package and reran the 154-update DialogSum/GPT-2 and 169-update DialogSum/Qwen3B training procedures. The 616 and 676 archived communication tensors, respectively, matched bit for bit at every update, as did the final parameters, gradients, optimizer states, and random states. These checks verify the implemented input path and the recorded sequence of updates. The general conditional-independence claim follows from the theorem under its stated assumptions. A separate 169-update H4/Qwen2.5-1.5B replay at $K' = 3$ also reads no original private-token rows. Its 507 selected surrogate sequences reproduce all 676 archived communication tensors and the final model, optimizer, and random states exactly. This check covers the $K' = 3$ ablation branch; the original H4 main-table branch used variable $K' = 4$–6. The full-training replays verify reproduction of the recorded updates. The paired substitutions above verify message equality after changing private tokens.

F Fixed-Context Message Equality

Theorems 1 and 3 compare different private tokens under the same unprotected context C. This appendix performs that comparison on the implemented message construction. It fixes C, the model state, surrogate choices, and other computation inputs, then changes the original private tokens. Appendix H.5 checks the preceding surrogate construction, and Appendix E.10 checks complete training executions. Together, these checks connect the inputs used by the implementation to the equality argument in the proofs.

F.1 Experiment design

We test H4 with Qwen2.5-1.5B and GPT-2 small, and DialogSum and GSM8K with Qwen2.5-1.5B. Each setting contains 16 public-validation groups. A group consists of one original sequence and two alternatives obtained from its stored surrogate sequences. Each alternative changes at least one private token relative to the original. All unprotected tokens, private-token positions, and token

lengths remain fixed, so all three sequences have the same unprotected context C. Each alternative is compared separately with the original, giving 32 pairs per setting and 128 pairs overall. The selected surrogate sequences, their labels, model state, and all other inputs and random choices used by the tested update are fixed within each group. We compare the constructed smashed data and sent gradient directly, rather than comparing reconstruction accuracies. The two alternatives need not differ from each other.

F.2 Why the surrogate sequences and labels remain unchanged

The implemented construction uses C to determine the generator request, candidate checks, ordering, and selection. The removed private tokens are absent from these inputs. The separate pipeline audit in Appendix H.5 covers 603 contexts with two private-token substitutions each. All 1,206 substitutions preserve the generator request, checker inputs and outputs, ordering and selection inputs, selected indices, constructed token sequences, and surrogate labels. The audit checks the LM request and stored candidates; it makes no new LM calls. Holding the generator and its random choices fixed is the condition under which identical requests give the same surrogate construction. The 128 message comparisons start from the stored selected sequences and check the subsequent computation. These two checks cover successive parts of the implemented construction.

F.3 Exact equality of the constructed messages

Table 28: Private-token substitution with all other computation inputs fixed. Selected surrogate sequences and labels are retained unchanged by construction in every pair. The message columns count exact array equality; no numerical tolerance or rounded aggregate is used. Each group has one original sequence and two alternatives.

Configuration	Groups	Pairs	Same Z'	Same Δ	NoDef Z changed
H4 / Qwen2.5-1.5B	16	32	32/32	32/32	32/32
DialogSum / Qwen2.5-1.5B	16	32	32/32	32/32	32/32
GSM8K / Qwen2.5-1.5B	16	32	32/32	32/32	32/32
H4 / GPT-2 small	16	32	32/32	32/32	32/32
Total	64	128	128/128	128/128	128/128

Each comparison evaluates one message-construction step. Table 28 reports all 128 pairs. AliasGuard's smashed data Z' and sent gradient Δ are exactly equal in every pair; NoDef's smashed data change in every pair. The archived gradient field `G_U` records Δ after the implementation's scaling, clipping, and dtype conversion. Thus the check compares the gradient actually passed between model segments. These measurements validate the message-construction computation used in Theorem 3. Its statement across training additionally requires the theorem's conditions on initial states, randomness, and every update. Separate checks of training reproducibility are recorded in Appendix E.10.

F.4 Why recovery can remain nonzero

Fix C, the public data, model state, and random choices. For two sequences that differ only in their private tokens, the tested construction gives

$$(Z'_1, \Delta_1) = (Z'_2, \Delta_2).$$

An adaptive reconstruction procedure receives C, its permitted public inputs, and these messages. It therefore receives identical inputs for the two sequences. With its own random choices fixed, it returns the same reconstruction; if these choices are drawn independently from the same law, its output distribution is the same. The messages consequently provide no way to distinguish these two private-token sequences under the fixed conditions. Theorems 1 and 3 state the corresponding conditional-independence results under their specified premises.

Correctness still compares the reconstruction with the original private tokens. For a simple schematic example, consider two valid identifiers with the same token length in a code context that copies an

array and then returns it. The unprotected code may make a name such as `copy` more likely than a less descriptive identifier. Public code can also teach the reconstruction procedure that frequency. If it always returns `copy` for this C, it is correct when the private identifier is `copy` and incorrect when the identifier differs. Both outcomes are compatible with identical AliasGuard messages. Figure 5(b) illustrates this logic; its three alternatives do not specify a measured recovery rate or assume equally likely private tokens.

In probabilistic terms, $\Pr(d \mid C)$ describes how likely each private datum item is after the unprotected context is known. These probabilities can be unequal. Eqs. (3) and (6) say that observing AliasGuard's messages leaves them unchanged. A likely identifier can therefore remain a successful guess after the messages are observed. Text-anonymization studies likewise address inference of personal attributes, such as age or location, from published text (Staab et al., 2024; 2025; Yan et al., 2026).

Private-token recovery measures how often a particular reconstruction procedure guesses correctly. Message equality asks a different question: can changing the true private tokens change what the server receives? The fixed-input experiment directly tests the second question. Different datasets can still have different recovery rates because their contexts and public token frequencies differ. Different finite decoders can also use the same context with different success. The results in Appendix E examine those procedures while retaining the distinction between correct guesses and dependence on the true private tokens.

F.5 Which inputs each method uses

Table 29 summarizes the implemented methods specified in Appendix B. NoDef and DualGuard process the original sequence and next-token labels. DP-forward perturbs the smashed data while retaining original labels; DP-SGD* perturbs the backward computation while retaining the original forward input. AliasGuard constructs both paths from the selected surrogate sequences and labels. The table shows why both the sequences and labels enter AliasGuard's construction. Section 4.5 measures their separate contributions. The direct changed-token message comparisons above cover NoDef and AliasGuard.

Table 29: Inputs to the five message computations. Only AliasGuard replaces the original private tokens in both input sequences and supervision.

Method	Forward computation	Backward computation
NoDef	Original sequence	Original next-token labels
DualGuard	Original sequence; learned transformations	Original labels; retention objectives
DP-forward	Original smashed data, normalized and perturbed	Original next-token labels
DP-SGD*	Original sequence	Original labels; clipped and perturbed gradients
AliasGuard	Selected surrogate sequences	Surrogate labels

G Utility Measurements

The main task scores evaluate complete generated answers. This appendix asks which parts of those answers account for the utility difference. Appendix G.1 scores reference-answer tokens with and without private positions, using the same model outputs. Appendix G.2 changes person names at evaluation time and checks the answers of fixed trained models. Appendix G.3 complements H4 word overlap with Python grammar and compilation checks. These measurements distinguish token prediction, answer generation, and code validity.

G.1 Prediction of unprotected tokens

Section 4.4 separates prediction of reference-answer tokens from scores of generated answers. We use the same final seed-3407 NoDef and AliasGuard models as Table 1. The saved evaluation

records identify private-token positions in some test answers and explicitly confirm their absence in others. We call their union the registered subset. U-C-PPL and U-Top1 use this subset so that every scored position has a known status. The remaining answers have unknown private-token positions and retain this designation in Table 30. All positions used in the scores come directly from the original records.

Table 30 accounts for every official test sequence. The Qwen models share the recorded positions. GPT-2 has different token positions and length exclusions. NoDef and AliasGuard use identical sequences and positions within every reported pair.

Table 30: Coverage of known private-token positions in test answers. †One sequence exceeds the supported length; ‡two do. There are no alignment failures. The registered subset is the sum of the private-token and confirmed-absent columns.

Dataset	Tokenizer	Official	Private tokens	Confirmed absent	Unknown
H4	Qwen	2003	150	0	1853
H4	GPT-2	2003	142	0	1860†
DialogSum	Qwen	500	55	5	440
DialogSum	GPT-2	500	55	5	438‡
GSM8K	Qwen	1319	355	236	728
GSM8K	GPT-2	1319	355	236	728

For a test sequence s, let $\mathcal{I}_s$ contain the answer-token positions scored by C-PPL. Let $\mathcal{I}_{p,s}$ contain the recorded private-token positions, and let $\mathcal{I}_{u,s} = \mathcal{I}_s \setminus \mathcal{I}_{p,s}$ contain the remaining positions. The correct token at position i is $y_{s,i}$. After receiving the correct preceding tokens, the model assigns probability $P_{s,i}(v)$ to each possible next token v.

C-PPL takes the negative natural logarithm of the probability assigned to each correct answer token, averages these numbers over all scored tokens, and exponentiates the average. U-C-PPL performs the same calculation using only unprotected positions. A lower value means that the model assigns higher probabilities to the correct tokens. U-Top1 instead counts how often the correct token has the highest probability and divides by the number of scored unprotected tokens:

$$L_u = \frac{\sum_s \sum_{i \in \mathcal{I}_{u,s}} -\log P_{s,i}(y_{s,i})}{\sum_s |\mathcal{I}_{u,s}|}, \qquad \text{U-C-PPL} = \exp(L_u), \tag{17}$$

$$\text{U-Top1} = 100\, \frac{\sum_s \sum_{i \in \mathcal{I}_{u,s}} \mathbf{1}[\arg\max_v P_{s,i}(v) = y_{s,i}]}{\sum_s |\mathcal{I}_{u,s}|}. \tag{18}$$

Here $|\mathcal{I}_{u,s}|$ counts positions, $\arg\max$ chooses the token with the highest probability, and $\mathbf{1}$ is one for a correct prediction and zero otherwise. Replacing $\mathcal{I}_{u,s}$ by $\mathcal{I}_s$ in the first line gives C-PPL. We add token totals before dividing; each token has equal weight. A token at input position j is predicted from the output at $j-1$. The answer boundaries, ignored padding, and treatment of the end-of-sequence token follow the original C-PPL code.

NoDef and AliasGuard use the same test sequences, correct preceding tokens, and scored positions. Private tokens earlier in the reference answer remain in the input. Only their positions in the score are omitted. The trained weights stay fixed throughout evaluation. Thus these scores measure prediction given the correct preceding tokens; generated-answer accuracy is measured separately.

For each difference in Table 32, we draw 2,000 sets of test sequences with replacement, using random seed 20260921. Both methods receive the same drawn sequence indices. We recompute the score from the token totals in each draw and report the middle 95% of these differences. These intervals describe variation across test sequences for the fixed training seed 3407; they do not describe variation across training seeds.

Why including private tokens changes the H4 comparison. Let N_u and N_p count the scored unprotected and private positions in these same sequences. Let L_p be the average negative log probability at private positions, and L_{all} the average at all scored positions. The average over all

Table 31: Prediction of unprotected tokens on the registered test subsets. n/N gives sequences with known private-token positions over all supported official test sequences. AliasGuard columns are shaded; bold selects the better point estimate within each pair. U-Top1 is a percentage.

			U-C-PPL ↓		U-Top1 ↑	
Dataset	Model	n/N	NoDef	AliasGuard	NoDef	AliasGuard
H4	GPT-2 small	142/2002	**2.455**	2.500	**78.152**	77.865
H4	Qwen2.5-1.5B	150/2003	1.752	**1.737**	84.181	**84.652**
H4	Qwen2.5-3B	150/2003	1.746	**1.736**	84.280	**85.072**
DialogSum	GPT-2 small	60/498	**5.884**	6.094	**58.712**	56.949
DialogSum	Qwen2.5-1.5B	60/500	**3.797**	4.063	**64.734**	63.711
DialogSum	Qwen2.5-3B	60/500	**3.443**	3.620	**67.667**	65.211
GSM8K	GPT-2 small	591/1319	**4.346**	4.379	**65.674**	65.575
GSM8K	Qwen2.5-1.5B	591/1319	**1.607**	1.647	**86.813**	86.301
GSM8K	Qwen2.5-3B	591/1319	**1.573**	1.600	**87.473**	87.011

Table 32: Paired changes on the registered subsets. Brackets give 95% intervals. Negative changes favor AliasGuard for U-C-PPL; positive changes favor AliasGuard for U-Top1 (percentage points).

Dataset	Model	Δ U-C-PPL	Δ U-Top1
H4	GPT-2 small	+0.045 [+0.019, +0.075]	-0.287 [-0.652, +0.090]
H4	Qwen2.5-1.5B	-0.015 [-0.039, +0.007]	+0.470 [+0.013, +0.968]
H4	Qwen2.5-3B	-0.010 [-0.040, +0.019]	+0.792 [+0.250, +1.367]
DialogSum	GPT-2 small	+0.210 [+0.110, +0.327]	-1.763 [-2.825, -0.657]
DialogSum	Qwen2.5-1.5B	+0.267 [+0.137, +0.419]	-1.023 [-2.406, +0.249]
DialogSum	Qwen2.5-3B	+0.177 [+0.107, +0.261]	-2.456 [-3.948, -1.071]
GSM8K	GPT-2 small	+0.034 [+0.023, +0.044]	-0.099 [-0.245, +0.042]
GSM8K	Qwen2.5-1.5B	+0.040 [+0.034, +0.045]	-0.513 [-0.665, -0.369]
GSM8K	Qwen2.5-3B	+0.027 [+0.021, +0.032]	-0.462 [-0.622, -0.310]

Table 33: The same 150 H4 test sequences per Qwen model, scored with all answer tokens (C-PPL) or only unprotected tokens (U-C-PPL). Lower is better. The models and preceding input tokens are unchanged.

Model	Method	C-PPL ↓	U-C-PPL ↓
Qwen2.5-1.5B	NoDef	**1.7900**	1.752
	AliasGuard	1.9784	**1.737**
Qwen2.5-3B	NoDef	**1.7594**	1.746
	AliasGuard	1.9594	**1.736**

tokens is the count-weighted average of the two parts:

$$L_{\text{all}} = \frac{N_u L_u + N_p L_p}{N_u + N_p}.$$

For each H4 Qwen model, AliasGuard has a higher L_{all} and a lower L_u than NoDef. Because the two methods score exactly the same positions, the increase in the all-token average must come from the private positions. After these positions are omitted from the score, AliasGuard has lower U-C-PPL and higher U-Top1 for both H4 Qwen models. U-Top1 improves by 0.470 and 0.792 percentage points, with paired intervals above zero. The U-C-PPL improvements are smaller and their intervals contain zero.

This calculation explains the reversal without changing the model or its input: the all-token score includes a part whose prediction becomes harder, while prediction of the unprotected part is preserved or improved. Theorem 4 explains why unprotected next-token labels are preserved during training; Theorem 2 bounds the corresponding change in smashed data.

The nine U-Top1 comparisons stay within 2.46 percentage points of NoDef (Table 32), with improvements in both H4 Qwen models. Task-score gains also occur in other datasets: Table 1 shows higher DialogSum ROUGE-L for GPT-2 and Qwen2.5-1.5B, and higher GSM8K answer accuracy for GPT-2. These observations support retaining useful prediction at unprotected positions while protecting private tokens. Theorems 2 and 4 establish the corresponding properties of smashed data and labels under their stated assumptions.

The scores describe two stages of model use. U-Top1 and U-C-PPL predict a token after receiving the correct preceding tokens. Task scores evaluate answers that the model generates using its own preceding predictions. Accordingly, their changes can differ: the main-table Qwen GSM8K task scores change by -5.31 and -10.77 points, while U-Top1 changes by -0.513 and -0.462 points. The complete three-seed study in Table 16 further shows U-Top1 within 0.77 points of NoDef and task-score gains in some settings. During generation, each predicted token becomes part of the input for later predictions. The reference-token scores instead supply the correct preceding tokens at every position. Small differences in the latter measurement therefore need not imply equally small changes in complete generated answers. The H4 decomposition locates the source of its C-PPL difference in private-token predictions; explaining generated answers additionally requires examining the task, model, and training seed.

Numerical checks. All 18 fixed models retain each supported sequence's per-token probabilities and correctness counts. On 20 sequences per model, the calculation exactly reproduces the original C-PPL from the same model outputs. The largest relative difference from the archived full-test C-PPL is 0.048861%; this comparison is separate from the 20-sequence checks.

Checking unprotected next-token labels in the training inputs. For Qwen2.5-1.5B at seed 3407, we inspect the saved selected surrogate sequences for 128 training sequences per dataset. These are training inputs, separate from the test sequences above. At every unprotected position, we compare the next-token label in each selected sequence with the corresponding original token. All 206,770 comparisons over 60,515 positions agree (Table 34). Consequently, averaging the selected labels leaves each unprotected next-token label unchanged, as stated by Theorem 4. This check uses the saved labels from the completed training runs.

Table 34: Saved-label check for Theorem 4. Each unprotected position is counted once per original training sequence; its label is compared once per selected surrogate sequence. H4 uses four to six selected sequences per training sequence, and the other datasets use three. All cases use Qwen2.5-1.5B and seed 3407.

Dataset	Training sequences	Unprotected positions	Label comparisons	Differences
H4	128	12,251	61,978	0
DialogSum	128	26,359	79,077	0
GSM8K	128	21,905	65,715	0

G.2 CHANGING PERSON NAMES IN TEST SEQUENCES

These evaluations use the original seed-3407 Table 1 checkpoints. Each model answers the original test sequence and three copies separately. In each copy, all occurrences of a person name are changed consistently; the other words and numbers remain fixed. No training is repeated. The retained GSM8K questions have the same correct numerical answer under all four names. DialogSum references receive the same name changes as their inputs. The role of private tokens in the answer also matters for prompt sanitization. Pr$\epsilon\epsilon$mpt chooses its protection according to whether the answer needs their content or only their format (Roy Chowdhury et al., 2026).

The input checks require the replacement tokens to occupy the recorded private positions and leave all remaining text unchanged. For each model, 111 of 128 selected GSM8K sequences pass these checks. DialogSum retains 34 of 53 sequences for each Qwen model and 36 of 50 for GPT-2. Selection uses these input checks, not model answers. Meaning has not been independently checked by human reviewers. Across tasks and models, 437 sequence–model pairs remain. A sequence evaluated with two models counts as two pairs, not two distinct questions. For GSM8K with Qwen2.5-3B, H100 and A100 numerical checks did not agree; all original and modified prompts were therefore evaluated on A100 using the same fixed weights. The other five groups use their original evaluation environments.

GSM8K numerical-answer accuracy is the percentage of generated answers whose extracted number equals the correct number. DialogSum ROUGE-L compares the longest sequence of matching words, in the same order, in a generated summary and a reference summary. Let a and b be the length of that matching sequence divided by the generated and reference summary lengths, respectively. The score is $100 \times 2ab/(a + b)$, or zero when no words match. We use the highest score among the three references. To exclude spelling differences in person names, each corresponding name in the generated and reference summaries is replaced by the same marker before computing ROUGE-L. All scores are on a 0–100 scale.

For each original sequence, we average the scores of its three changed-name copies. The change is that average minus the original score. To form each 95% interval, we draw original sequences with replacement 2,000 times, using random seed 3407. The same draw is used for both methods, with each sequence's three copies kept together. We report the middle 95% of the resulting changes. A zero-width interval occurs when all retained changes are identical.

Table 35: Task scores before and after changing person names. Change is the score after changing names minus the original score; brackets give paired 95% intervals. GSM8K uses numerical-answer accuracy and DialogSum uses ROUGE-L after corresponding names are mapped to the same marker. Each GSM8K group has 111 sequences; DialogSum has 34/34/36 for the two Qwen models/GPT-2.

Dataset	Model	Method	Original	After	Change [95% interval]
GSM8K	Qwen2.5-1.5B	NoDef	32.432	35.435	+3.003 [-3.303, +9.017]
GSM8K	Qwen2.5-1.5B	AliasGuard	31.532	32.432	+0.901 [-4.805, +6.306]
GSM8K	Qwen2.5-3B	NoDef	45.946	48.048	+2.102 [-3.303, +7.508]
GSM8K	Qwen2.5-3B	AliasGuard	31.532	36.036	+4.505 [-0.901, +10.811]
GSM8K	GPT-2 small	NoDef	0.901	1.802	+0.901 [-0.601, +3.003]
GSM8K	GPT-2 small	AliasGuard	0.901	0.901	+0.000 [+0.000, +0.000]
DialogSum	Qwen2.5-1.5B	NoDef	45.067	46.483	+1.416 [-1.023, +4.090]
DialogSum	Qwen2.5-1.5B	AliasGuard	48.719	45.368	-3.351 [-7.336, -0.001]
DialogSum	Qwen2.5-3B	NoDef	49.098	46.990	-2.108 [-5.071, +0.806]
DialogSum	Qwen2.5-3B	AliasGuard	46.872	44.604	-2.268 [-5.719, +0.972]
DialogSum	GPT-2 small	NoDef	40.334	39.443	-0.890 [-3.490, +1.925]
DialogSum	GPT-2 small	AliasGuard	38.776	39.224	+0.448 [-2.348, +3.208]

After names change, AliasGuard's GSM8K accuracy changes by $+0.901$, $+4.505$, and 0.000 points for Qwen2.5-1.5B, Qwen2.5-3B, and GPT-2, respectively. All six GSM8K intervals include zero. In the A100 evaluation with Qwen2.5-3B, NoDef answers 51/111 original questions and 160/333 changed-name questions correctly; AliasGuard answers 35/111 and 120/333. The AliasGuard–NoDef difference between these changes is $+2.402$ points $[-5.105, +9.910]$; their accuracy dif-

ference after changing names is -12.012 points. For DialogSum/Qwen2.5-1.5B, the difference between the methods' changes is -4.768 points $[-9.455, -0.895]$. These comparisons measure how a fixed trained model responds to a name change. The arithmetic relation stays the same, so a changed answer can be traced to the changed wording of the input. Each arithmetic problem is repeated under several names only during evaluation. This tests answer generation; Appendix F separately tests whether private-token changes affect AliasGuard's training messages.

G.3 H4 WORD OVERLAP AND PYTHON CODE VALIDITY

F1 counts matching word occurrences in the generated code and its reference. Let a be this count divided by the number of generated words, and b the same count divided by the number of reference words. The score is $100 \times 2ab/(a+b)$, or zero when no words match. C-PPL uses the definition in Appendix G.1. These two scores retain the complete official test sets.

For the sequences whose original language label specifies Python, we also check whether each generated program follows Python's grammar (Parse) and whether Python can compile it into instructions (Compile). The latter check does not run the program. Both rates divide successful outputs by eligible Python outputs: 663 per Qwen model and 662 for GPT-2. We use the original Python 3.11 implementation and language labels. Removing one complete Python code fence does not change the Parse results. All 12,016 saved outputs are retained, including non-Python outputs outside these two denominators. The recorded Parse decisions are reproduced exactly.

Table 36: H4 word-overlap and code-validity scores at seed 3407. F1 and C-PPL retain the original full-test scores. Parse and compile columns show successful Python outputs over eligible outputs, followed by percentages. AliasGuard rows are shaded; bold marks the better rate or task score within each model. U-C-PPL and U-Top1 on the registered subsets appear in Table 31.

Model	Method	F1 ↑	C-PPL ↓	Parse ↑	Compile ↑
GPT-2 small	NoDef	17.995	**4.472**	55/662 (8.31)	55/662 (8.31)
GPT-2 small	AliasGuard	**18.732**	4.676	204/662 (**30.82**)	204/662 (**30.82**)
Qwen2.5-1.5B	NoDef	**49.795**	**1.963**	614/663 (**92.61**)	613/663 (**92.46**)
Qwen2.5-1.5B	AliasGuard	46.688	2.037	599/663 (90.35)	599/663 (90.35)
Qwen2.5-3B	NoDef	**48.053**	**1.967**	603/663 (90.95)	603/663 (90.95)
Qwen2.5-3B	AliasGuard	45.547	2.043	617/663 (**93.06**)	616/663 (**92.91**)

The GPT-2 parse and compile gain is $+22.508$ points $[+18.035, +26.698]$, reproducing the earlier 8.31% versus 30.82% comparison as 55/662 versus 204/662. The Qwen1.5B parse/compile changes are -2.262 $[-4.970, +0.315]$ and -2.112 $[-4.793, +0.462]$; the Qwen3B changes are $+2.112$ $[-0.304, +4.566]$ and $+1.961$ $[-0.453, +4.362]$. Intervals use 2,000 paired draws of complete official sequences at bootstrap seed 20260921, with Python eligibility retained in each draw.

AliasGuard improves GPT-2 word overlap and Python validity: F1 rises from 17.995 to 18.732, and 149 additional outputs pass both Python checks. Qwen2.5-3B also produces more valid Python outputs, with 14 additional Parse passes and 13 additional Compile passes. Both Qwen models have higher U-Top1. The complete table separates these gains from changes in other scores: GPT-2 has higher C-PPL and U-C-PPL, and the Qwen models have lower F1. F1 measures word overlap; Parse and Compile check grammar and compilation. Execution-based checks of whether programs return the correct answer are outside the recorded evaluation.

H SELECTED SURROGATES: EVALUATION AND TIMING DETAILS

This appendix separates the number, construction, and use of surrogates. Appendices H.1–H.4 vary K' and measure recovery, task scores, and computation cost. Appendix H.5 checks that surrogate construction uses only permitted inputs; Appendix H.6 measures how well the selected surrogates preserve unprotected-token computation. Appendix H.7 then separates the effects of surrogate sequences and surrogate labels. These are different roles: independence from original private tokens supports privacy, while compatibility with C helps retain useful task information.

H.1 Scope and comparison conditions

The controlled study compares $K' = 1, 3, 5$ at seed 3407 on GSM8K and DialogSum with GPT-2 small and Qwen2.5-1.5B/3B. Within each dataset–model pair, all three runs share the initial model, optimizer and random state, original sequence order, stored $K = 30$ bank, tokenizer, precision, model partition, training budget, target manifest and evaluators. Each original sequence has one fixed ordering of its 30 candidates. The three runs select the first one, first three, or first five candidates, respectively: indices [0], [0:3], and [0:5]. Thus every candidate used at a smaller K' is also used at the larger K'. Qwen uses 169 updates and GPT-2 uses 154, each on one original sequence. All six new $K' = 3$ runs use the same PyTorch 2.10.0/CUDA 12.8 implementation as the twelve retained $K' = 1, 5$ runs. Source, input, initial-state and checkpoint hashes are verified for this matched matrix.

F, B, and BF use the same 128 targets at their recorded updates. F performs 200 refinement iterations, B performs 32, and BF performs 32 B iterations followed by 200 F iterations. The evaluator constructs its surrogate approximation from public training vocabulary, simulates the observations, and reconstructs under the same information condition at every K'. Client-selected surrogates and private head/tail weights remain withheld; true private tokens are joined only after predictions are sealed. Every new run passes complete server-state replay and the eight-target reference/accelerated reconstruction check before formal evaluation. Final utility uses all 1,319 GSM8K questions and 500 DialogSum sequences (498 for GPT-2), with unchanged decoding and output caps. GSM8K reports numerical-answer accuracy; DialogSum reports the best ROUGE-L over three references, and its PPL uses the fixed first reference.

H.2 Privacy and task scores

Table 37 reports all eighteen matched runs. BF is the main recovery measure used to compare K'; F and B show the separate observation directions from the same reconstruction program. Recovery denominators follow the model's tokenizer. Bold entries identify the best value within a dataset–model pair; the shaded row is the operational default $K' = 3$.

Table 37: Matched sensitivity results. F/B/BF and task scores are percentages; lower recovery/PPL and higher task scores are better. BF count retains the exact numerator and denominator.

Dataset	Model	K'	F	B	BF	BF count	Task	Full PPL	C-PPL
GSM8K	Q1.5B	1	**0.407**	1.018	**0.407**	2/491	32.676	2.894	1.652
		3	2.240	1.629	2.240	11/491	35.557	2.992	1.658
		5	4.684	**0.407**	4.684	23/491	**36.315**	**2.845**	**1.650**
GSM8K	Q3B	1	**0.000**	1.426	**0.000**	0/491	48.673	2.862	1.610
		3	3.055	**0.815**	3.055	15/491	46.929	2.741	1.616
		5	3.462	**0.815**	3.462	17/491	**50.872**	**2.729**	**1.604**
GSM8K	GPT-2	1	0.707	**1.060**	0.707	4/566	1.820	6.495	4.429
		3	**0.353**	1.590	**0.353**	2/566	2.199	6.495	4.418
		5	1.237	**1.060**	1.237	7/566	**2.274**	**6.493**	**4.404**
DialogSum	Q1.5B	1	5.411	6.061	5.411	25/462	40.802	6.438	**3.584**
		3	**4.978**	**3.680**	**4.978**	23/462	41.649	**6.433**	3.655
		5	6.494	4.545	6.494	30/462	**42.014**	6.445	3.673
DialogSum	Q3B	1	5.411	**3.030**	5.411	25/462	**42.544**	6.147	**3.387**
		3	**3.680**	4.762	**3.680**	17/462	42.012	6.154	3.419
		5	6.710	6.710	6.710	31/462	42.333	**6.117**	3.408
DialogSum	GPT-2	1	4.508	**2.049**	4.508	22/488	34.783	8.124	5.831
		3	3.279	3.074	3.279	16/488	**34.978**	8.118	**5.785**
		5	**1.434**	3.689	**1.434**	7/488	34.285	**8.115**	5.787

All eighteen runs retain BF recovery at or below 6.710%. The default $K' = 3$ gives 0.353–4.978% across the six dataset–model pairs. Higher K' improves task scores in some settings: GSM8K/Qwen2.5-1.5B rises from 32.676% to 36.315%, and DialogSum/Qwen2.5-1.5B rises from 40.802 to 42.014 ROUGE-L. Increasing K' does not produce a single direction of change across all settings. In three of the six pairs, BF recovery changes direction between $K' = 1, 3, 5$; task scores

also change direction in three pairs. The choice $K' = 1$ is also covered by the construction: the average then contains one selected sequence. The results allow K' to be chosen according to the measured task score, recovery, and computation cost for each setting.

H.3 Matched overhead relative to NoDef

The timing reference is one NoDef training update with trainable head, trunk, and tail. AliasGuard evaluates K' selected sequences in the head, averages their smashed data, and passes that one average through the trunk and tail. This adds head computation while retaining one exchanged smashed-data matrix per original sequence. AliasGuard uses its original selected-surrogate update. For every configuration, one instrumented update matches the uninstrumented update exactly in loss, parameters, gradients, optimizer state and random state. NoDef and all three AliasGuard configurations use one identical A100 GPU model. Within each dataset–model pair they also use the same physical GPU, 50 original training sequences stratified over sequence length, and input order.

Each of five repeats performs ten warmup updates, restores the same initial model and optimizer state, and measures fifty updates. The orders are $(0, 1, 3, 5)$, $(1, 3, 5, 0)$, $(3, 5, 0, 1)$, $(5, 0, 1, 3)$ and $(0, 3, 1, 5)$, where 0 denotes NoDef. We wait for pending GPU work to finish immediately before and after each update, so the elapsed time includes its completed computation. GPU event timestamps measure the individual components. Tables report 250 measured updates per configuration. SD is the standard deviation of update times. P50 is the median, and P95 is the time at or below which 95% of updates finish. The archive also contains the mean of each repeat and the difference between methods within that repeat.

The online interval includes stored-bank lookup, candidate selection, sequence and label assembly, head/trunk/tail forward and backward work, three optimizer updates, tensor copies and protocol checks. We report one-time bank preparation separately. Model loading and checkpoint persistence are common costs outside both methods' update timers. Communication volume is the total byte count of the arrays sent between model segments. The single-GPU timing measures computation and local transfers; an actual client–server connection adds its network delay.

Table 38: Matched A100 time per update (ms). The last two columns compare each configuration with NoDef on the same 50 sequences.

Dataset	Model	Config.	Mean	SD	P50	P95	Δ ms	Δ %
GSM8K	Q1.5B	NoDef	304.63	2.95	304.66	309.25	0.00	0.00
		$K' = 1$	307.26	14.75	305.66	311.24	2.63	0.86
		$K' = 3$	328.23	9.32	327.62	332.76	23.60	7.75
		$K' = 5$	365.27	27.56	363.53	369.39	60.63	19.90
GSM8K	Q3B	NoDef	419.83	5.84	418.40	430.21	0.00	0.00
		$K' = 1$	419.90	12.86	417.52	429.66	0.07	0.02
		$K' = 3$	442.14	6.61	440.48	454.49	22.30	5.31
		$K' = 5$	474.91	12.13	472.18	491.97	55.08	13.12
GSM8K	GPT-2	NoDef	45.88	13.81	44.25	48.70	0.00	0.00
		$K' = 1$	46.80	3.43	46.06	51.23	0.92	2.02
		$K' = 3$	57.30	3.57	56.59	61.05	11.42	24.90
		$K' = 5$	68.68	4.08	67.80	72.85	22.80	49.70
DialogSum	Q1.5B	NoDef	312.19	24.82	309.27	315.21	0.00	0.00
		$K' = 1$	314.07	22.28	312.40	317.07	1.88	0.60
		$K' = 3$	334.62	21.90	333.25	338.97	22.43	7.18
		$K' = 5$	372.84	31.79	369.45	376.38	60.66	19.43
DialogSum	Q3B	NoDef	426.09	17.68	423.94	432.27	0.00	0.00
		$K' = 1$	428.91	14.36	426.80	437.50	2.82	0.66
		$K' = 3$	448.31	4.35	448.33	455.20	22.22	5.21
		$K' = 5$	479.52	6.98	479.26	486.32	53.43	12.54
DialogSum	GPT-2	NoDef	48.91	5.73	47.75	57.23	0.00	0.00
		$K' = 1$	51.69	5.68	50.61	60.07	2.78	5.68
		$K' = 3$	65.53	30.60	60.75	72.59	16.62	33.99
		$K' = 5$	74.28	7.66	72.25	85.01	25.37	51.88

Allocated memory counts memory held by live GPU arrays; reserved memory also includes memory retained by the framework for later use. One MiB is 2^{20} bytes and one KiB is 2^{10} bytes.

Table 39: GPU memory and actual communication volume. Memory columns are mean/max peak allocated and mean peak reserved over five repeats (MiB). Payload is mean raw tensor volume per update (MiB), summed over Z, U, G_U, G_Z; the two directions each contribute half.

Dataset	Model	Config.	Alloc. mean	Alloc. max	Reserved	Payload
GSM8K	Q1.5B	NoDef	14217.3	14217.3	15474.0	2.114
		$K' = 1$	14218.4	14218.4	15478.0	2.114
		$K' = 3$	14481.2	14481.2	15542.0	2.114
		$K' = 5$	14747.5	14747.5	15744.0	2.114
GSM8K	Q3B	NoDef	28890.3	28890.3	30056.0	2.819
		$K' = 1$	28890.7	28890.7	30062.0	2.819
		$K' = 3$	29240.1	29240.1	30186.0	2.819
		$K' = 5$	29582.6	29582.6	30592.0	2.819
GSM8K	GPT-2	NoDef	2505.4	2505.4	2918.0	1.796
		$K' = 1$	2507.4	2507.4	2920.0	1.796
		$K' = 3$	2662.5	2662.5	2834.0	1.796
		$K' = 5$	2802.5	2802.5	3126.0	1.796
DialogSum	Q1.5B	NoDef	14264.1	14264.1	15818.0	2.437
		$K' = 1$	14266.0	14266.0	15828.0	2.437
		$K' = 3$	14572.4	14572.4	15926.0	2.437
		$K' = 5$	14886.9	14886.9	16248.0	2.437
DialogSum	Q3B	NoDef	28970.6	28970.6	30186.0	3.249
		$K' = 1$	28974.2	28974.2	30328.0	3.249
		$K' = 3$	29356.8	29356.8	31006.0	3.249
		$K' = 5$	29747.0	29747.0	31332.0	3.249
DialogSum	GPT-2	NoDef	2698.1	2698.1	3012.0	2.656
		$K' = 1$	2699.6	2699.6	3014.0	2.656
		$K' = 3$	2910.4	2910.4	3408.0	2.656
		$K' = 5$	3109.5	3109.5	3570.0	2.656

At $K' = 3$, added time per update is 11.42–23.60 ms (5.21–33.99%) relative to NoDef. All 24,000 recorded transmissions have the same array dimensions, numerical type, and byte count across configurations for the same sequence. Increasing K' therefore changes the computation but not the measured volume of the exchanged arrays.

H.4 Component timing and one-time preparation

Table 40 decomposes the same measured updates. head, trunk, and tail times each include forward and backward computation. Opt. adds the three parameter-update times. Prep measures candidate selection and construction of the sequences and labels during the update. Other is the elapsed time left after these components are subtracted; it includes averaging, surrogate-label loss, copies, and checks. The component timers can include a pause while the CPU schedules the next operation on the same GPU execution stream.

Table 40: Mean component times (ms), using the same 250 updates as the latency table.

Dataset	Model	Config.	Prep	head	trunk	tail	Opt.	Other	Total
GSM8K	Q1.5B	NoDef	0.63	10.40	199.59	19.64	39.95	34.42	304.63
		$K' = 1$	1.17	10.37	199.00	21.69	39.95	35.09	307.26
		$K' = 3$	1.23	30.52	200.21	20.65	39.82	35.80	328.23
		$K' = 5$	1.63	64.03	200.16	22.12	40.64	36.69	365.27
GSM8K	Q3B	NoDef	0.67	10.44	259.90	19.46	79.30	50.05	419.83
		$K' = 1$	1.24	10.39	258.84	20.11	79.26	50.06	419.90
		$K' = 3$	1.31	30.61	259.41	20.40	79.29	51.12	442.14
		$K' = 5$	2.68	60.31	259.86	20.39	79.32	52.36	474.91

Dataset	Model	Config.	Prep	head	trunk	tail	Opt.	Other	Total
GSM8K	GPT-2	NoDef	0.49	4.04	17.80	7.32	8.42	7.81	45.88
		$K' = 1$	1.03	4.06	17.62	7.99	7.57	8.52	46.80
		$K' = 3$	1.09	14.07	17.55	8.07	7.56	8.96	57.30
		$K' = 5$	1.44	24.61	17.60	8.08	7.58	9.37	68.68
DialogSum	Q1.5B	NoDef	0.68	10.63	203.35	22.67	39.89	34.97	312.19
		$K' = 1$	1.35	10.62	204.07	21.94	39.85	36.24	314.07
		$K' = 3$	1.41	31.13	204.74	21.07	39.82	36.45	334.62
		$K' = 5$	1.95	65.49	205.36	22.70	39.78	37.57	372.84
DialogSum	Q3B	NoDef	0.72	10.72	265.27	20.86	79.30	49.22	426.09
		$K' = 1$	3.08	10.72	265.21	20.68	79.30	49.93	428.91
		$K' = 3$	1.40	31.56	264.53	20.84	79.30	50.67	448.31
		$K' = 5$	1.86	61.67	264.44	20.81	79.32	51.42	479.52
DialogSum	GPT-2	NoDef	0.60	4.76	18.92	8.56	7.60	8.47	48.91
		$K' = 1$	1.23	4.77	19.07	9.52	7.64	9.46	51.69
		$K' = 3$	1.32	15.28	20.60	10.89	7.62	9.83	65.53
		$K' = 5$	1.83	26.23	19.02	9.58	7.61	10.00	74.28

At $K' = 3$, the additional head computation contributes 10.03–20.84 ms per update. It is the main source of the added work: the head processes three surrogate sequences, whereas the trunk and tail each process one averaged result. The component table also reports selection, label construction, parameter updates, and the remaining measured time.

The external LM is invoked during one-time preparation of the $K = 30$ bank, before the online training loop. We reuse 618 recorded generator invocations attached to 8,991 accepted natural-language banks. A batch's observed wall latency is divided by its request count and attached repeated invocations are summed per context. This gives a per-request share of the observed service time, including waiting at the service. It measures preparation latency, not GPU computation on the training device. For each dataset and tokenizer, we select five fixed training contexts covering different sequence lengths. Replaying their original proposals through the original checker, including rejected candidates, reproduces all twenty final candidate banks exactly. This CPU measurement separates checking/filtering/tokenization from gzip/atomic storage and the original SQLite writer, including its integrity check. The twenty SQLite replays reproduce every stored row. Tokenizer loading is startup. Qwen1.5B and Qwen3B share a bank; GPT-2 has its own tokenizer-specific bank.

Table 41: One-time preparation per context. Generator mean is batch-amortized wall time over the complete formal training set; check and store are mean CPU milliseconds from five exact proposal replays per stratum. Total P50/P95 combine each replay with its recorded generator cost (seconds).

Dataset	Tokenizer	Gen. s	Check ms	Store ms	Total P50 s	Total P95 s
GSM8K	Qwen	7.433	26.21	17.43	5.515	14.147
GSM8K	GPT-2	7.571	20.97	13.20	5.380	6.898
DialogSum	Qwen	7.448	25.56	15.81	5.452	6.891
DialogSum	GPT-2	6.894	29.95	16.19	5.567	23.493

Table 42: Actual stored input cost. SQLite size is the complete stored bank file, including indices and metadata. Per-context JSON bytes give mean/median/P95; sequence mean counts its replacement text, token IDs and labels. The archive includes sequence median/P95 and schedule-file bytes. Tokenizer/model weights are shared assets.

Dataset	Tokenizer	Banks	SQLite MiB	Mean KiB	P50 KiB	P95 KiB	Seq. KiB
GSM8K	Qwen	3171	446.3	74.3	70.2	118.4	2.0
GSM8K	GPT-2	3379	473.6	70.5	66.5	112.1	1.9
DialogSum	Qwen	1221	179.0	93.1	87.8	163.5	2.6
DialogSum	GPT-2	1220	178.4	93.0	88.0	161.6	2.6

For a schedule of N updates we estimate local training as $N\overline{T}_{\text{online}}$. Adding the recorded batch-amortized preparation for its unique contexts and the measured mean CPU checking/storage cost gives the estimated total when no surrogate bank has yet been prepared. These are cost estimates from component measurements; bank preparation can be batched or parallelized. The same bank is reused across K', training seeds and the two Qwen sizes, so its one-time cost is charged once per input group. With a prepared bank, the incremental cost is the online column alone. The final column retains the actual new training-loop wall time, which also writes the recorded communication stream; startup and final checkpoint persistence are separately archived.

Table 43: Schedule cost accounting (seconds). NoDef and $K' = 3$ online columns extrapolate the matched mean times per update; prep and total estimate preparing the bank once and then training, with the associated total overhead relative to NoDef. Actual is the completed $K' = 3$ training loop.

Dataset	Model	N	NoDef	K3	Prep	Total	Δ %	Actual
GSM8K	Q1.5B	169	51.5	55.5	1263.5	1319.0	2462.0	72.4
GSM8K	Q3B	169	71.0	74.7	1263.5	1338.3	1786.2	93.1
GSM8K	GPT-2	154	7.1	8.8	1171.2	1180.0	16601.4	21.8
DialogSum	Q1.5B	169	52.8	56.5	1265.6	1322.2	2406.1	74.4
DialogSum	Q3B	169	72.0	75.8	1265.6	1341.4	1762.8	94.5
DialogSum	GPT-2	154	7.5	10.1	1068.8	1078.9	14224.7	20.9

H.5 Checking surrogate construction

PrivCode trains a code generator with DP-SGD and uses its generated code to train another LM (Liu et al., 2026b). For AliasGuard, the construction requirement is that surrogate generation uses only C and the permitted public inputs. The checks below test this input restriction.

The audit of seed-3407 stored inputs covers all 169 Qwen training contexts per dataset and a 32-context GPT-2 subset per dataset: 603 contexts, each paired with two alternative sets of private tokens. Holding the public context and random choices fixed, all 1,206 substitutions preserve the serialized generator request, checker/filter inputs and outputs, bank key, candidate ordering, selected indices, constructed token sequences and surrogate labels. Each comparison asks whether replacing the original private tokens can change an input or decision in surrogate construction. Identical results through these steps check the implemented input restriction in Theorems 1 and 3.

The full stored banks contain 360,120 candidates across the three datasets and two tokenizer groups. For every candidate, the checks verify the number of private datum items, the token length of each replacement, its recorded positions, valid token IDs, and correct next-token labels. They also verify that every unprotected token stays unchanged. H4 additionally checks that replacements are valid identifiers and that the resulting code parses as Python. Its syntax comparison consistently renames corresponding identifiers before comparing the parsed code, so an identifier change alone does not count as a change in code structure. The training selections use 3,237 distinct bank entries. The historical natural-language builder uses a context-conditioned Codex prompt. Historical H4 instead compiles a neutral Codex identifier pool against each public code context, retaining its recorded $K' = 4$–6 selection. The dependency audit replays these actual construction paths.

For compatibility, 100 contexts per dataset span sequence lengths and datum counts. For each context, its first three selected surrogates are compared with replacements taken from other public surrogate banks. We call these replacements donor surrogates. Each donor has the same private-datum type, number of items, and token lengths as the selected surrogate. Donors are considered in a fixed random order without ranking their meaning. Both choices must pass the same position and syntax checks. Qwen2.5-7B-Instruct, from a different family than the Codex generator, sees only the masked context and two candidate substitutions in the sequence. The original private tokens and method labels are withheld. The judging model chooses A, B, or Tie by greedy decoding: at each output position it chooses the highest-probability token. A and B identify the two displayed substitutions; Tie indicates no preference. Per dataset, the selected surrogate appears on A in 150 pairs and on B in 150.

Across all 900 pairs, selected surrogates win 15.67%, donor controls win 14.33% and 70.00% are tied. The selected surrogates receive a positive net preference on DialogSum ($+7.00$ pp, 95% CI

Table 44: Complete structural audit (100% pass) and 900 blinded pairwise judgments. Net preference is selected wins minus donor wins, in percentage points. Intervals use 2,000 paired bootstrap draws over the 100 original contexts per dataset, keeping each context's three judgments together.

	Audited candidates		Blind preference (%)			Net preference
Dataset	Qwen	GPT-2	**Selected**	Tie	Donor	pp [95% CI]
H4	47,160	43,230	22.67	51.67	25.67	-3.00 [-11.00, +4.33]
GSM8K	95,130	101,370	6.33	87.33	6.33	+0.00 [-4.33, +4.67]
DialogSum	36,630	36,600	18.00	71.00	11.00	+7.00 [+1.00, +13.00]

$[+1.00, +13.00]$). GSM8K is tied overall, with 87.33% ties; H4's -3.00 pp difference has an interval spanning zero. Thus the contextual preference is dataset-dependent, with the clearest separation on DialogSum.

H.6 How context-compatible surrogates affect unprotected positions

We compare selected and matched donor substitutions at the same final seed-3407 Qwen2.5-1.5B AliasGuard checkpoint, without parameter updates. Of 128 prespecified training contexts per dataset, valid matched controls are available for 123 H4, 124 GSM8K and all 128 DialogSum contexts. Each contributes three paired substitutions. The remaining five H4 and four GSM8K contexts have no donor satisfying all structural constraints. For each substitution s, we compute

$$D_Z(s) = \frac{\|(Z_s - Z)_{T_u}\|_F}{\sqrt{|T_u| d_h}},$$

where Z is the original sequence's smashed data, Z_s is the smashed data after substitution, T_u contains its unprotected input positions, and d_h is the head output width. This score measures the size of the change per unprotected position and output coordinate; lower is better. We also compute L_u from Appendix G.1 separately for each sequence: the mean negative log probability of the original unprotected tokens in its reference answer. The same fixed trunk and tail produce these probabilities after each substitution. All 375 masks match the original reference-answer boundaries and recorded private positions; a token at position j is scored using the prediction at $j - 1$. We first average the three substitutions within each original sequence, then average sequences equally to obtain the means and paired intervals below. Here each sequence has equal weight; the main U-C-PPL calculation instead gives each scored token equal weight.

Table 45: Local changes on matched original sequences. Difference is selected minus donor; lower values are better. Bold marks the lower mean within a pair. Intervals use 2,000 original-sequence paired bootstrap draws.

Dataset (n)	Metric ↓	Selected	Donor	Difference	95% paired CI
H4 (123)	D_Z	0.053984	**0.053906**	+0.000078	[-0.000404, +0.000509]
H4 (123)	L_u	**0.307717**	0.309855	-0.002138	[-0.004403, +0.000074]
GSM8K (124)	D_Z	**0.035870**	0.036098	-0.000228	[-0.001022, +0.000551]
GSM8K (124)	L_u	0.263089	**0.262652**	+0.000436	[-0.002705, +0.003468]
DialogSum (128)	D_Z	**0.023311**	0.024420	-0.001109	[-0.001588, -0.000615]
DialogSum (128)	L_u	**0.650173**	0.695047	-0.044874	[-0.068516, -0.022866]

On DialogSum, selected surrogates reduce D_Z by 4.54% and mean L_u by 6.46% relative to matched donor substitutions; both paired intervals lie below zero. Thus the context-compatible choices keep smashed data closer to those of the original sequence and assign higher probabilities to its correct unprotected tokens in this comparison. H4 and GSM8K show smaller differences whose intervals include zero. Median D_Z (selected/donor) is 0.055202/0.054436 on H4, 0.033721/0.034474 on GSM8K, and 0.022950/0.024270 on DialogSum. These measurements distinguish two properties of surrogate construction. Using only C and permitted public inputs removes dependence on the original private tokens. Selecting surrogates that fit C can also reduce changes in unprotected-token

computation, as the DialogSum results show. The first property supports privacy; the second helps explain utility. The comparison holds model parameters fixed. Final task scores after training appear in Section 4.2.

H.7 Surrogate sequences and surrogate labels

Section 4.5 compares four combinations of sequences and labels. For each dataset, we use its final seed-3407 Qwen2.5-1.5B AliasGuard model and 64 registered training sequences. Model parameters remain fixed throughout evaluation. The original sequence produces Z; the recorded surrogate sequences produce Z'. Independently, the loss uses original labels Y or surrogate labels Y'. We retain the recorded surrogate selections, including H4's $K' = 4$–6, and the same unprotected labels in all four combinations. The four combinations are (Z, Y), (Z', Y), (Z, Y'), and (Z', Y'). For each one, the backward gradient is computed from that forward input and that set of labels. This holds the trained model fixed while separating the two construction choices.

Reconstruction uses the registered public pretrained head and tail, decoder, and surrogate sequences constructed from public context. The client's selected surrogates and trained client weights remain private. F uses 200 refinement steps; B uses 32; BF uses 32 backward steps followed by 200 forward steps. All combinations use reconstruction seed 9107 and the same token vocabulary. Unprotected positions remain fixed as the candidate sequences are updated. On five sequences per dataset, the accelerated implementation exactly reproduces the reference predictions and intermediate computations. F calculations are shared between (Z, Y) and (Z, Y'), and between (Z', Y) and (Z', Y'), because each pair supplies F with identical inputs.

Table 46: Complete results for Section 4.5. Recovery is the percentage of private-token positions reconstructed correctly, counting every occurrence. The same 64 sequences supply 542 positions on H4, 223 on GSM8K, and 236 on DialogSum. All four combinations use the same trained model for that dataset. Lower is better; bold marks the lowest value within each dataset and column. Shading marks AliasGuard's (Z', Y').

Dataset	Forward	Labels	F (%)	B (%)	BF (%)
H4	Z	Y	78.782	59.410	88.561
H4	Z'	Y	**20.849**	36.900	**20.849**
H4	Z	Y'	78.782	43.358	78.782
H4	Z'	Y'	**20.849**	**22.878**	**20.849**
GSM8K	Z	Y	35.874	5.381	41.704
GSM8K	Z'	Y	**1.794**	4.933	**1.794**
GSM8K	Z	Y'	35.874	**1.794**	35.874
GSM8K	Z'	Y'	**1.794**	2.691	**1.794**
DialogSum	Z	Y	49.576	10.593	57.203
DialogSum	Z'	Y	**3.390**	13.136	**3.390**
DialogSum	Z	Y'	49.576	**3.814**	49.576
DialogSum	Z'	Y'	**3.390**	5.932	**3.390**

Table 2 takes the largest of the three complete F, B, and BF rates in each row of Table 46. Table 47 gives the exact B counts for the comparison that holds Z' fixed. To obtain paired 95% intervals, we resample the 64 sequences with replacement 2,000 times using seed 20260922. Each draw uses the same sequence indices for both combinations; we divide their respective correct-token totals by the total number of private-token positions and subtract the two rates. The 2.5th and 97.5th percentiles give the reported interval.

With Z' fixed, replacing Y by Y' reduces B recovery by 14.022, 2.242, and 7.203 percentage points on H4, GSM8K, and DialogSum, respectively. All three paired intervals lie above zero. F is unchanged between these two combinations because it receives the same forward input. Together, the four combinations show the separate roles of surrogate sequences and surrogate labels: the former change what F observes, and the latter further reduce recovery from the backward computation in these comparisons.

Table 47: Effect of surrogate labels with Z' fixed. Fractions give correctly reconstructed private-token positions divided by all private-token positions. Reduction is the B rate with Y minus the B rate with Y', in percentage points (pp). Larger reductions indicate a greater decrease in recovery.

Dataset	B: (Z', Y)	B: (Z', Y')	Reduction (pp)	95% interval (pp)
H4	200/542	124/542	14.022	[10.020, 18.313]
GSM8K	11/223	6/223	2.242	[0.412, 4.899]
DialogSum	31/236	14/236	7.203	[3.652, 11.225]

I Joint Observations and Optimal Reconstruction Loss

This appendix explains why the best possible reconstruction rule can use both observations at least as well as either one alone. A rule receiving both can always ignore one. We then distinguish this mathematical fact from the output of a particular F, B, or BF program.

The server reconstructs a sequence X from smashed data Z and a gradient observation Δ. Here Δ can contain one position's gradient or several gradients. The unprotected context C is available in every comparison. All other permitted public information is fixed; the probability expressions below leave this common conditioning implicit.

Let a denote a reconstruction and $\mathcal{A}$ the set of allowed reconstructions. A loss $\ell(a, X) \geq 0$ assigns a cost to predicting a when the true sequence is X. For example, a cost of one for an incorrect sequence and zero for a correct sequence measures exact-reconstruction error. We compare the smallest expected loss over all rules that use the specified observations. The rules and loss are measurable, meaning that the probabilities and expected losses used below are defined.

Define

$$\begin{aligned} R^\star_C &= \inf_f \mathbb{E}[\ell(f(C), X)], & R^\star_{C,Z} &= \inf_f \mathbb{E}[\ell(f(C, Z), X)], \\ R^\star_{C,\Delta} &= \inf_f \mathbb{E}[\ell(f(C, \Delta), X)], & R^\star_{C,Z,\Delta} &= \inf_f \mathbb{E}[\ell(f(C, Z, \Delta), X)]. \end{aligned} \tag{19}$$

The symbol $\inf$ denotes the greatest lower bound of the expected losses over all permitted reconstruction rules. A rule attaining that bound need not exist. Every expectation uses the same joint distribution of X, C, Z, Δ and the same loss.

For the logarithmic-loss statement, suppose X takes values in a finite set $\mathcal{S}$ of possible sequences, and the output is any probability distribution q on $\mathcal{S}$. Its loss is $-\log q(X)$, with infinite loss if it assigns probability zero to a realized sequence having positive probability. A fixed vocabulary and a bounded sequence length give such a finite set. This loss rewards assigning more probability to the true sequence. We use natural logarithms; their unit is called a nat. For any observation O, write $p_O(x) = \Pr(X = x \mid O)$ and define conditional entropy by

$$H(X \mid O) = \mathbb{E}[-\log p_O(X)].$$

The expectation averages over sequences and observations. Thus $H(X \mid O)$ is the average logarithmic loss when predictions use the true conditional probabilities given O. Conditional mutual information measures how much this average loss falls when another observation is available. In the cases below,

$$\begin{aligned} I(X; \Delta \mid C, Z) &= H(X \mid C, Z) - H(X \mid C, Z, \Delta), \\ I(X; Z \mid C, \Delta) &= H(X \mid C, \Delta) - H(X \mid C, Z, \Delta). \end{aligned}$$

The conditional probabilities are well defined for these finite sequences and finite-dimensional array observations.

Theorem 5 (Joint observations and optimal reconstruction loss)**.** *Use the same joint distribution, possible reconstructions, and loss in all four problems. Then*

$$R^\star_{C,Z,\Delta} \leq \min\{R^\star_{C,Z}, R^\star_{C,\Delta}\} \leq R^\star_C. \tag{20}$$

For logarithmic loss with finite $\mathcal{S}$ and unrestricted probability predictions,

$$\begin{aligned} R^\star_{C,Z} - R^\star_{C,Z,\Delta} &= I(X; \Delta \mid C, Z), \\ R^\star_{C,\Delta} - R^\star_{C,Z,\Delta} &= I(X; Z \mid C, \Delta). \end{aligned} \tag{21}$$

Thus, joint observations strictly improve optimal logarithmic loss over one observation exactly when the additional conditional mutual information is positive.

Proof. Take any rule f that observes (C, Z). The rule $h(C, Z, \Delta) = f(C, Z)$ is allowed in the joint-observation problem and has exactly the same expected loss. The infimum in the joint problem cannot exceed the expected loss of this h. Taking the infimum over f gives $R^\star_{C,Z,\Delta} \le R^\star_{C,Z}$. Ignoring Z instead proves the corresponding inequality with $R^\star_{C,\Delta}$. A rule observing C alone can similarly ignore the additional arrays in either single-observation problem. This proves Eq. (20) without assuming an optimal rule is attained.

For the logarithmic-loss identities, let O be one of (C, Z), (C, Δ), or (C, Z, Δ). At a fixed observation o, write $p_o(x) = \Pr(X = x \mid O = o)$. For any prediction q_o on $\mathcal{S}$, with zero-probability terms interpreted as zero,

$$\sum_x p_o(x)[-\log q_o(x)] = -\sum_x p_o(x) \log p_o(x) + \sum_x p_o(x) \log \frac{p_o(x)}{q_o(x)}.$$

The last sum is nonnegative. Indeed, if $q_o(x) = 0$ where $p_o(x) > 0$, it is infinite. Otherwise $\log u \le u - 1$ gives

$$\sum_{x:p_o(x)>0} p_o(x) \log \frac{q_o(x)}{p_o(x)} \le \sum_{x:p_o(x)>0} [q_o(x) - p_o(x)] \le 0.$$

Choosing $q_o = p_o$ makes that sum zero. Hence the conditional expected logarithmic loss is minimized by the true conditional distribution. Averaging over O shows that the smallest expected logarithmic loss using O equals $H(X \mid O)$. Subtracting the corresponding conditional entropies yields Eq. (21). The finite set $\mathcal{S}$ makes these optimal risks finite, so their differences are well defined. □

Relation to AliasGuard. Use Z' in place of Z for AliasGuard. Under Theorem 3, adding its messages does not change the conditional distribution of the private tokens given all relevant unprotected contexts. For a finite set of possible private-token reconstructions, the best possible exact reconstruction success is therefore unchanged: both choose a private-token sequence with the largest conditional probability given the available observations. The theorem makes those conditional probabilities the same before and after the messages are added. In the single-sequence experiment, observing C also fixes the unprotected tokens, so the same conclusion applies to reconstructing X. Across sequences and rounds, the corresponding comparison uses $\mathcal{C}$ and $O^{(T)}$. The success probability of inference from context depends on its largest conditional probability.

Relation to implemented reconstruction. The mathematical comparison includes every rule that can ignore an additional observation. A particular BiSR program instead generates a finite list of candidates and selects one using its implemented score. An added gradient-matching term can change the ordering and select a less accurate candidate. Appendix B.3 shows this effect using two DP-forward BF evaluations of the same recorded messages.

We therefore report each complete program's result and specify its candidates, information, and computation budget. These measurements test concrete reconstruction procedures. Theorem 5 explains the best possible rule, while Theorem 3 establishes when AliasGuard's messages leave the conditional distribution unchanged.

Logarithmic loss can improve while the best exact reconstruction success stays constant. For example, let C, Z be constant and let $X \in \{0, 1\}$. Suppose two equally likely gradients give $\Pr(X = 1 \mid \Delta) = 0.6$ and 0.8. Without the gradient, this probability is 0.7. The best exact guess is 1 both before and after observing the gradient, with average success 0.7 in both cases. The posterior probabilities do change, so logarithmic loss improves. A positive logarithmic-loss improvement and an increase in exact reconstruction are different conclusions.