Supplementary Material

A Discussion

A.1 Threat Model Discussion

In this section, we further discuss the threat model chosen by LAMP and compare it to the related work. To make our attack as generic as possible, we relax common assumptions that have been exploited to reconstruct client's data in the literature before. In particular, LAMP is applicable even if:

- The model's word embeddings are not fine-tuned. As the gradients of the word embedding vectors are non-zero for words that are contained in the client's training data and zero otherwise, revealing the gradients to the server will allow it to easily obtain the client sequence up to reorder. This constitutes a serious breach of clients' privacy and, thus, we assume the word embeddings are not trainable.
- The model's positional embeddings are not fine-tuned. Similarly to the word-embedding gradients, Lu et al. [22] have recently demonstrated that for batch size $B = 1$ positional-embedding gradients can leak client's full sequence. To this end, we assume models without trainable positional embeddings.
- The model's transformer blocks contain no bias terms. Lu et al. [22] have also shown that the popular attack by Phong et al. [28] can be applied on the bias terms of transformer blocks to leak the client's data. To this end, we assume models without transformer block biases.
- The transformer model is fine-tuned on a classification task. As language modeling tasks are usually self-supervised, they often feed the same data to the model both as inputs and outputs. Based on this observation, Fowl et al. [6] have recently shown that label reconstruction algorithms can be used to obtain the client's word counts. We thus assume the more challenging binary classification setting.
- The server is honest-but-curious, i.e., it aims to learn as much as possible about clients' data from gradients but does not tamper with the learning protocol. While prior work has shown that a malicious server can force a client to leak much more data [6], this is orthogonal to our work. We focus on the honest-server setting instead, which is the harder setting to attack.

Note that the assumptions we make for our transformer networks can result in a small amount of accuracy loss on the final fine-tuned model, but preserve the client's data privacy much better. We emphasize that while LAMP focuses on attacks in the harder setting, it is also applicable to the simpler settings without modification.

A.2 Improvements over Prior Work

In this section, we outline the differences between LAMP and TAG, and we discuss how these differences help LAMP to significantly improve its text data reconstruction from gradients compared to TAG.

A major difference between the two methods is the introduction of our discrete optimization that takes advantage of a GPT-2 language model to help reconstruct the token order better than TAG. Our discrete optimization step is novel in several ways:

- It is based on a set of discrete transformations that fix common token reordering problems arising from the continuous reconstruction.
- We take advantage of the perplexity computed by the existing language models such as GPT-2 to evaluate the quality of different discrete transformations.

Table 6: Visualization of intermediate steps of text reconstruction from gradients, on a sequence from the CoLA dataset. Note that TAG performs 2500 steps, as opposed to LAMP_{Cos} which terminates at 2000, as this is usually sufficient for convergence.

Iteration	TAG	LAMP_{Cos}
0	billie icaohwatch press former spirit technical	trinity jessie maps extended evidence private peerage whatever
500	enough stadium six too 20 le was,	many marbles have six. too.
1000	have stadium seven too three le. marble	; have six too many marbles.
1500	have respect six too manys, marble	have six. too many marbles.
2000	have... six too many i, marble	have six. too many marbles.
2500	have... six too manys, marble	
Reference	i have six too many marbles.	i have six too many marbles.

Table 7: In this experiment we reconstruct 100 random selected sentences with our methods and the baselines on the CoLA dataset and the $\text{BERT}_{\text{BASE}}$ (B=1) model 10 times with 10 different randomly selected set of sentences. We report the mean and standard deviation of all ROUGE measures.

	R-1	R-2	R-L
DLG	56.2 ± 5.0	6.5 ± 1.6	45.0 ± 2.6
TAG	74.4 ± 3.1	10.7 ± 1.8	53.0 ± 2.1
LAMP_{Cos}	87.8 ± 2.6	48.4 ± 5.5	74.6 ± 2.9
$\text{LAMP}_{L_2+L_1}$	83.1 ± 3.7	40.7 ± 5.7	69.3 ± 3.6

- Finally, our discrete optimization is alternated with the continuous, allowing for both take advantage of the result of the other which ultimately results in better token order reconstruction (See our ablation in Sec. 5).

The other major difference between our method and existing work is the choice of the error function $\mathcal{L}_{\text{rec}}$ used in the continuous part of the optimization. Our choice of reconstruction loss results in better reconstruction of individual tokens and thus increases R-1. In particular, we show that the cosine error function, previously applied in the image domain, can often outperform the error function suggested by TAG for text reconstruction and introduce a regularization term $\mathcal{L}_{\text{reg}}$ that helps the continuous optimization to converge faster to more accurate embeddings using prior knowledge about the vector sizes of embeddings.

B Detailed Text Reconstruction Example

In Table 6, we show the intermediate steps of text reconstruction for a real example taken from our experiments presented in Table 3. We can observe that LAMP_{Cos} reaches convergence significantly faster than the TAG baseline, and that after only 500 iterations most words are already reconstructed by our method.

C Additional Experiments

C.1 Dependency of Experimental Results to the Chosen Sentences

Throughout this paper, we conducted our experiments on the same 100 sequences randomly chosen from the test portion of the datasets we attack. In this experiment, we show that our results are consistent when different sets of 100 sequences are used. To achieve this, we ran the $\text{BERT}_{\text{BASE}}$ CoLA experiment with B=1 on additional 10 different sets of 100 randomly chosen sentences from the COLA test set. We report the mean $\pm$ one standard deviation of the resulting R-1, R-2 and R-L metrics averaged across the 10 sets in Table 7. We see that the results are consistent with our original findings.

Table 8: This experiment shows the trade-off between the final network accuracy (measured by MCC) and the reconstruction quality from gradients with different percentages of zeroed-out gradient entries on the CoLA dataset on $\text{BERT}_{\text{BASE}}$ (B=1).

Zeroed %	MCC	R-1	R-2	R-L
0	0.557	89.6	51.9	76.2
75	0.557	79.4	34.5	66.3
90	0.534	61.9	20.1	53.9
95	0.515	39.0	5.8	37.4
99	0.371	24.7	0.0	24.7

Table 9: This experiment shows the effect on reconstruction of the chosen number of initializations n_{init} used in LAMP_{Cos} on all 3 datasets for $\text{BERT}_{\text{BASE}}$ (B=1).

	CoLA			SST-2			RottenTomatoes		
n_{init}	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
1	87.3	48.1	73.2	87.4	**60.8**	**78.8**	63.7	**16.6**	**43.8**
500	**89.6**	**51.9**	**76.2**	**88.8**	56.9	77.7	**64.7**	16.3	43.1

C.2 Attacking Gradient Masking Defense

We experimented with a defense which zeroes out a percentage of elements in the gradient vector. In Table 8 we vary the percentage and report MCC, R-1 and R-2. While zeroing out most gradients weakens the attack, it also reduces utility (MCC) of the model.

C.3 Dependence on the Number of Initializations

In this section, we investigate the influence of our proposed initialization on the reconstructions of LAMP_{Cos} by comparing a single random initialization ($n_{\text{init}} = 1$) with using our two-step initialization procedure with $n_{\text{init}} = 500$ on the $\text{BERT}_{\text{BASE}}$ model and batch size of 1. The results are shown in Table 9. We observe that the two-step initialization scheme consistently improves individual token recovery (measured in terms of R-1) but may in some cases slightly degrade token ordering results (measured in terms of R-2). Even though we used two-step initialization in the paper (it is strictly better on one dataset and non-comparable to single random initialization on the remaining datasets), it is indeed sometimes possible to get slightly better R-2 results with the latter.

D Additional Experimental Details

We run all of our experiments on a single NVIDIA RTX 2080 Ti GPU with 11 GB of RAM, except for the experiments on $\text{BERT}_{\text{LARGE}}$ for which we used a single NVIDIA RTX 3090 Ti GPU with 24 GB of RAM instead.

As we explain in Sec. 5, we choose the hyperparameters of our methods using a grid search approach on the CoLA and RottenTomatoes datasets. For CoLA, we first evaluated 50 hyperparameter combinations on 10 randomly selected (in a stratified way with respect to length) sequences from the training set (after removing the 100 test sequences). Then, we further evaluated the best 10 combinations on different 20 sequences from the training set. For RottenTomatoes, we picked the hyperparameters from the same 10 best combinations and evaluated them on the same 20 additional sequences. For both LAMP_{Cos} and the baselines, we investigated the following ranges for the hyperparameters: $\alpha_{\text{lm}} \in [0.05, 0.2]$, $\alpha_{\text{reg}} \in [0.01, 1]$, $\lambda \in [0.001, 0.5]$, $\gamma \in [0.8, 1]$, and $\alpha_{\text{tag}} \in [10^{-5}, 10^{2}]$. For $\text{LAMP}_{L_2+L_1}$, we consider $\alpha_{\text{lm}} \in [30, 240]$ and $\alpha_{\text{reg}} \in [10, 100]$, as the scale of the loss values is orders of magnitude larger than in LAMP_{Cos}. We experimentally found that our algorithm is robust with respect to the exact values of n_c and n_d, provided that they are sufficiently large. To this end, we select $n_d = 200$ because we found that the selected token order from the 200 random transformations is close to the optimal one according to $\mathcal{L}_{\text{rec}}(\boldsymbol{x}) + \alpha_{\text{lm}}\mathcal{L}_{\text{lm}}(\boldsymbol{t})$ for the sentence lengths present in our datasets. Similarly, we set $n_c = 75$ ($n_c = 200$ for $\text{BERT}_{\text{LARGE}}$) which allows

our continuous optimization to significantly change the embeddings before applying the next discrete optimization step in the process. Finally, we also observed the performance of our algorithm is robust with respect to n_{init}, so we set it to 500 throughout the experiments. We note that compared to TAG and DLG, the only additional hyperparameters we have to search over are α_{reg} and α_{lm} which makes the grid search feasible for our methods.

The resulting hyperparameters for LAMP_{Cos} are $\alpha_{\text{lm}} = 0.2$, $\alpha_{\text{reg}} = 1.0$, $\lambda = 0.01$, $\gamma = 0.89$. In contrast, the best hyperparameters for $\text{LAMP}_{L_2+L_1}$ are $\alpha_{\text{tag}} = 0.01$, $\alpha_{\text{lm}} = 60$, $\alpha_{\text{reg}} = 25$, $\lambda = 0.01$, $\gamma = 0.89$, as the loss $\mathcal{L}_{\text{tag}}$ is on a different order of magnitude compared to $\mathcal{L}_{\text{cos}}$. In our experiments, TAG's best hyperparameters are $\alpha_{\text{tag}} = 0.01$, $\lambda = 0.1$, $\gamma = 1.0$ (no decay), which we also use for DLG (with $\alpha_{\text{tag}} = 0.0$).

To account for the different optimizer used in $\text{BERT}_{\text{LARGE}}$ experiments, we tuned the learning rate λ for all methods separately in this setting by evaluating each method on 5 different learning rates in the range $[0.01, 0.1]$ on 10 randomly selected sentences from the CoLA dataset. This resulted in $\lambda = 0.1$ for DLG, and $\lambda = 0.01$ for TAG, LAMP_{Cos}, and $\text{LAMP}_{L_2+L_1}$. We applied the chosen values of λ to all 3 datasets. Additionally, following Geiping et al. [8] we clip the gradient magnitudes $\|\nabla_{\boldsymbol{x}} \mathcal{L}_{\text{rec}}(\boldsymbol{x})\|_2$ for our $\text{BERT}_{\text{LARGE}}$ experiments to 1.0 for DLG and TAG and 0.5 for LAMP_{Cos} and $\text{LAMP}_{L_2+L_1}$.

E Total Runtime of the Experiments

The $\text{BERT}_{\text{LARGE}}$ model experiments in Table 1 were the most computationally expensive to execute. They took between 50 hours per experiment for the LAMP_{Cos} and $\text{LAMP}_{L_2+L_1}$ methods and 70 hours for TAG, which executes two times more continuous optimization steps. Our experiments on the rest of the networks for both the baselines and our methods on batch size 1 ($B = 1$) all took between 8 and 16 hours to execute on a single GPU with our methods being up to 2x slower due to the additional computational cost of our discrete optimization. Additionally, our experiments on batch size 4 ($B = 4$) took between 8 and 36 hours to execute on a single GPU with our methods being up to 4x slower due to the additional computational cost of our discrete optimization.

F Potential Negative Societal Impact of This Work

Our work is closely related to the existing works on gradients leakage attacks (See Sec. 2) which are capable of breaking the privacy promise of FL e.g., Zhao et al. [41], Geiping et al. [7], Yin et al. [40], Deng et al. [3]. Similar to these works, LAMP can be used to compromise the privacy of client data in real-world FL setups, especially when no defenses are used by the clients. Our attack emphasizes that text data, which is commonly used in federated settings [30], is highly vulnerable to gradient leakage attacks, similarly to data in other domains, and that when FL is applied in practice extra steps need to be taken to mitigate the potential risks. Further, in line with the related work, we study a range of possible mitigations to our attack in Tables 1, 5 and 8 in the paper, thus promoting possible practical FL implementations that will be less vulnerable including those defended with Gaussian noise and gradient pruning and those using bigger batch sizes.