TextHide: Tackling Data Privacy in Language Understanding Tasks

Yangsibo Huang[†] **Zhao Song**[†‡] **Danqi Chen**[†] **Kai Li**[†] **Sanjeev Arora**[†‡]
[†]Princeton University [‡]Institute for Advanced Study
{yangsibo, zhaos}@princeton.edu
{danqic, li, arora}@cs.princeton.edu

Abstract

An unsolved challenge in distributed or federated learning is to effectively mitigate privacy risks without slowing down training or reducing accuracy. In this paper, we propose *TextHide* aiming at addressing this challenge for natural language understanding tasks. It requires all participants to add a simple encryption step to prevent an eavesdropping attacker from recovering private text data. Such an encryption step is efficient and only affects the task performance slightly. In addition, *TextHide* fits well with the popular framework of fine-tuning pre-trained language models (e.g., BERT) for any sentence or sentence-pair task. We evaluate *TextHide* on the GLUE benchmark, and our experiments show that *TextHide* can effectively defend attacks on shared gradients or representations and the averaged accuracy reduction is only 1.9%. We also present an analysis of the security of *TextHide* using a conjecture about the computational intractability of a mathematical problem.[1]

1 Introduction

Data privacy for deep learning has become a challenging problem for many application domains including Natural Language Processing. For example, healthcare institutions train diagnosis systems on private patients' data (Pham et al., 2017; Xiao et al., 2018). Google trains a deep learning model for next-word prediction to improve its virtual keyboard using users' mobile device data (Hard et al., 2018). Such data are decentralized but moving them to a centralized location for training a model may violate regulations such as Health Insurance Portability and Accountability Act (HIPAA) (Act, 1996) and California Consumer Privacy Act (CCPA) (Legislature, 2018).

Federated learning (McMahan et al., year; Kairouz et al., 2019) allows multiple parties training a global neural network model collaboratively in a distributed environment without moving data to a centralized storage. It lets each participant compute a model update (i.e., gradients) on its local data using the latest copy of the global model, and then send the update to the coordinating server. The server then aggregates these updates (typically by averaging) to construct an improved global model.

Privacy has many interpretations depending on the assumed threat models (Kairouz et al., 2019). This paper assumes an eavesdropping attacker with access to all information communicated by all parties, which includes the parameters of the model being trained. With such a threat model, a recent work (Zhu et al., 2019) suggests that an attacker can reverse-engineer the private input.

Multi-party computation (Yao, 1982) or homomorphic encryption (Gentry, 2009) can ensure full privacy but they slow down computations by several orders of magnitude. Differential privacy (DP) approach (Dwork et al., 2006; Dwork, 2009) is another general framework to ensure certain amount of privacy by adding controlled noise to the training pipeline. However, it trades off data utility for privacy preservation. A recent work that applies DP to deep learning was able to reduce accuracy losses (Abadi et al., 2016) but they still remain relatively high.

The key challenge for distributed or federated learning is to ensure privacy preservation without slowing down training or reducing accuracy. In this paper, we propose *TextHide* to address this challenge for natural language understanding tasks. The goal is to protect *training data privacy* at a minimal cost. In other words, we want to ensure that an adversary eavesdropping on the communicated bits will not be able to reverse-engineer

[1]Our code is available at https://github.com/Hazelsuko07/TextHide.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1368–1382

training data from any participant.

TextHide requires each participant in a distributed or federated learning setting to add a simple encryption step with one-time secret keys to hide the hidden representations of its text data. The key idea was inspired by *InstaHide* (Huang et al., 2020) for computer vision tasks, which encrypts each training datapoint using a random pixel-wise mask and the *MixUp* technique (Zhang et al., 2018a) of data augmentation. However, application of *InstaHide* to text data is unclear because of the well-known dissimilarities between image and language: pixel values are real numbers whereas text is sequences of discrete symbols.

TextHide is designed to plug into the popular framework which transforms textual input into output vectors through pre-trained language models (e.g., BERT (Devlin et al., 2019)) and use those output representations to train a new shallow model (e.g., logistic regression) for any supervised single-sentence or sentence-pair task. The pre-trained encoder is fine-tuned as well while training the shallow model. We evaluate *TextHide* on the GLUE benchmark (Wang et al., 2019). Our results show that *TextHide* can effectively defend attacks on shared gradients or representations while the averaged accuracy reduction is only 1.9%.

Lastly, *TextHide* and *InstaHide* have completely different security arguments due to the new designs. To understand the security of the proposed approach, we also invent a new security argument using a conjecture about the computational intractability of a mathematical problem.

2 *InstaHide* and Its Challenges for NLP

InstaHide (Huang et al., 2020) has achieved good performance in computer vision for privacy-preserving distributed learning, by providing a cryptographic[2] security while incurring much smaller utility loss and computation overhead than the best approach based on differential privacy (Abadi et al., 2016).

InstaHide is inspired by the observation that a classic computation problem, k-VECTOR SUBSET SUM[3], also appears in the *MixUp* (Zhang et al., 2018a) method for data augmentation, which is used to improve accuracy on image data.

To encrypt an image $x \in \mathbb{R}^d$ from a private dataset, *InstaHide* first picks $k-1$ other images $s_2, s_3, \ldots, s_k$ from that private dataset, or a large public dataset of N images, and random nonnegative coefficients λ_i for $i = 1, .., k$ that sum to 1, and creates a composite image $\lambda_1 x + \sum_{i=2}^{k} \lambda_i s_i$ (k is typically small, e.g., 4). A composite label is also created using the same set of coefficients.[4] Then it adds another layer of security: pick a *random mask* $\sigma \in \{-1, 1\}^d$ and output the encryption $\widetilde{x} = \sigma \circ (\lambda_1 x + \sum_{i=2}^{k} \lambda_i s_i)$, where $\circ$ is coordinate-wise multiplication of vectors. The neural network is then trained on encrypted images, which look like random pixel vectors to the human eye and yet lead to good classification accuracy. Note that the "one-time secret key" $\sigma, s_2, \cdots, s_k$ used to encrypt x will not be reused to encrypt other images.

Challenges of applying *InstaHide* to NLP. There are two challenges to apply *InstaHide* to text data for language understanding tasks. The first is the discrete nature of text, while the encryption in *InstaHide* operates at continuous inputs. The second is that most NLP tasks today are solved by *fine-tuning* pretrained language models such as BERT on downstream tasks. It remains an open question how to add encryption into such a framework and what type of security argument it will provide. The following section presents our approach that overcomes these two challenges.

3 *TextHide*: Formal Description

There are two key ideas in *TextHide*. The first one is using the "one-time secret key" coming from *InstaHide* for encryption, and the second is a method to incorporate such encryption into the popular framework of fine-tuning a pre-trained language model e.g., BERT (Devlin et al., 2019).

In the following, we will describe how to integrate *TextHide* in the federated learning setting (Section 3.1), and then present two *TextHide* schemes (Section 3.2 and 3.3). We analyze the security of *TextHide* in Section 3.4.

3.1 Fine-tuning BERT with *TextHide*

In a federated learning setting, multiple participants holding private text data may wish to solve NLP tasks by using a BERT-style fine-tuning

[2]Cryptosystem design since the 1970s seeks to ensure any attack must solves a computationally expensive task.

[3]k-VECTOR SUBSET SUM is known to be hard: in the worst case, finding the secret indices requires $\geq N^{k/2}$ time (Abboud and Lewi, 2013) under the conjecture *Exponential Time Hypothesis (Impagliazzo et al., 1998). See Appendix A.*

[4]Only the labels of the examples from the private dataset will get combined. See (Huang et al., 2020) or Section 3 for more details.

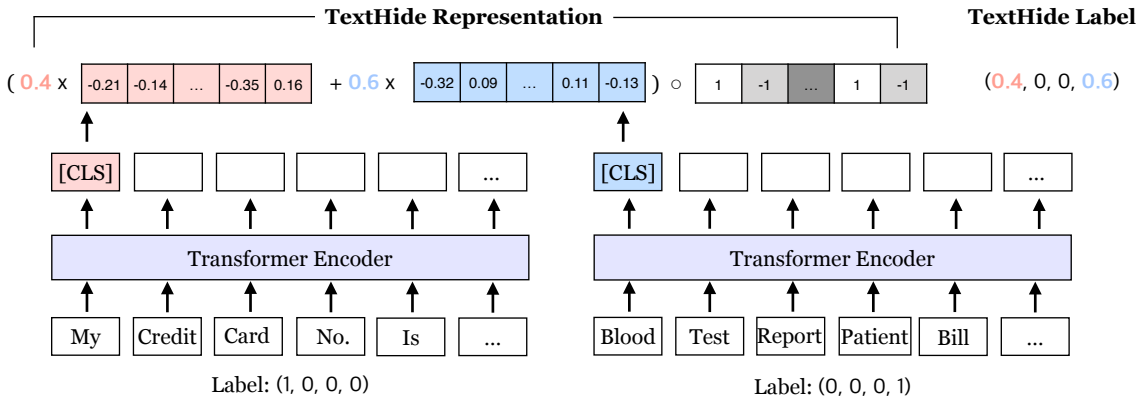


Figure 1: An illustration of *TextHide* encryption with $k = 2$, where k is the number of inputs (sentence or sentence-pair) got mixed in each *TextHide* representation. *TextHide* first encodes each text input using a transformer encoder, then linearly combines their output representations (i.e., `[CLS]` tokens), as well as their labels. Finally, an entry-wise mask is chosen from a randomly pre-generated pool and applied on the mixed representation. The entry-wise mask, together with the other datapoints to mix constitute the "one-time secret key" of the *TextHide* scheme. Note that training directly takes place on encrypted data and no decryption is needed.

pipeline, where *TextHide*, a simple *InstaHide*-inspired encryption step can be applied at its intermediate level to ensure privacy (see Figure 1).

The BERT fine-tuning framework assumes (input, label) pairs (x, y)'s, where x takes the form of `[CLS]`s_1 `[SEP]` for single-sentence tasks, or `[CLS]`s_1 `[SEP]`s_2 `[SEP]` for sentence-pair tasks. y is a one-hot vector for classification tasks, or a real-valued number for regression tasks.[5] For a standard fine-tuning process, federated learning participants use a BERT-style model f_{θ_1} to compute hidden representations $f_{\theta_1}(x)$'s for their inputs x's and then train a shallow classifier h_{θ_2} on $f_{\theta_1}(x)$, while also fine-tuning θ_1. The parameter vectors θ_1, θ_2 will be updated at the central server via pooled gradients. All participants hold current copies of the two models.

To ensure privacy of their individual inputs x's, federated learning participants can apply *TextHide* encryption at the *output* $f_{\theta_1}(x)$'s. The model h_{θ_2} will be trained on these encrypted representations. Each participant will compute gradients by back-propagating through their private encryption, and this is going to be the source of the secrecy: the attacker can see the communicated gradients but not the secret encryptions, which limits leakage of information about the input.

We then formally describe two *TextHide* schemes for fine-tuning BERT in the federated learning setting: *TextHide*$_{\text{intra}}$ which encrypts an input using other examples from the same dataset, and *TextHide*$_{\text{inter}}$ which utilizes a large public dataset to perform encryption. Due to a large public dataset, *TextHide*$_{\text{inter}}$ is more secure than *TextHide*$_{\text{intra}}$, but the latter is quite secure in practice when the training set is large.

[5]We will mainly use classification tasks as examples throughout the paper for brevity.

3.2 Basic *TextHide*: Intra-Dataset *TextHide*

In *TextHide*, we have a pre-trained text encoder f_{θ_1}, which takes x, a sentence or a sentence pair, and maps it to a representation $e = f_{\theta_1}(x) \in \mathbb{R}^d$ (e.g., $d = 768$ for BERT$_{\text{base}}$). We use $[b]$ to denote the set $\{1, 2, \cdots, b\}$. Given a dataset $\mathcal{D}$, we denote the set $\{x_i, y_i\}_{i\in[b]}$ an "input batch" by $\mathcal{B}$, where $x_1, \cdots, x_b$ are b inputs randomly drawn from $\mathcal{D}$, and $y_1, \cdots, y_b$ are their labels. For each x_i in the batch $\mathcal{B}$, $i \in [b]$, we can encode x_i using f_{θ_1}, and obtain a new set of $\{e_i = f_{\theta_1}(x_i), y_i\}_{i\in[b]}$. We refer to this set as an "encoding batch", and denote it by $\mathcal{E}$. Later in this section, we use $\widetilde{e}_i$ to denote the *TextHide* encryption of e_i for $i \in [b]$, and name the set $\widetilde{\mathcal{E}} = \{\widetilde{e}_i, y_i\}_{i\in[b]}$ as a "hidden batch" of $\mathcal{E}$.

We use $\sigma \in \{-1, +1\}^d$ to denote an entry-wise sign-flipping mask. For a *TextHide* scheme, $\mathcal{M} = \{\sigma_1, \cdots, \sigma_m\}$ denotes its randomly pre-generated mask pool of size m, and k denotes the number of sentences combined in a *TextHide* representation. We name such a parametrized scheme as (m, k)-*TextHide*.

(m, k)-*TextHide*. Algorithm 1 describes how (m, k)-*TextHide* encrypts an encoding batch $\mathcal{E} = \{e_i, y_i\}_{i\in[b]}$ into a hidden batch $\widetilde{\mathcal{E}}$, where b is the batch size. For each e_i in $\mathcal{E}$, *TextHide* linearly combines it with $k - 1$ other representations, as well as their labels. Then, *TextHide* randomly selects a mask σ_i from $\mathcal{M}$, the mask pool, and applies it on the combination using coordinate-wise

Algorithm 1 (m, k)-*TextHide*

procedure TEXTHIDE($\mathcal{E}, \mathcal{M}, k$)
  ▷ $\mathcal{E}$: the training batch, b: $|\mathcal{E}| = b$
  ▷ $\mathcal{M}$: the mask pool, m: $|\mathcal{M}| = m$
  ▷ k: number of training examples to be mixed
  ▷ Let $[b]$ denote the set $\{1, 2, \cdots, b\}$
  $\widetilde{\mathcal{E}} \leftarrow \emptyset$
  Generate π_1 such that $\pi_1(i) = i, \forall i \in [b]$
  Generate $k-1$ random permutations $\pi_2, \cdots, \pi_k : [b] \rightarrow [b]$
  Sample $\lambda_1, \cdots, \lambda_b \sim |\mathcal{N}(0, I_k)| \in \mathbb{R}^k$ uniformly at random, normalize s.t. $\sum_{j=1}^{k} (\lambda_i)_j = 1, \forall i \in [b]$.
  for $(e_i, y_i) \in \mathcal{E}$ **do**
    $\sigma_i \sim \mathcal{M}$
    $\widetilde{e}_i \leftarrow \sigma_i \circ \sum_{j=1}^{k} (\lambda_i)_j \cdot e_{\pi_j(i)}$
    $\widetilde{y}_i \leftarrow \sum_{j=1}^{k} (\lambda_i)_j \cdot y_{\pi_j(i)}$
    $\widetilde{\mathcal{E}} \leftarrow \widetilde{\mathcal{E}} \cup \{(\widetilde{x}_i, \widetilde{y}_i)\}$
  end for
  return $\widetilde{\mathcal{E}}$
end procedure

multiplication. This gives $\widetilde{e}_i$, the encryption of e_i (lines 12, 13 in Algorithm 1). Note that different e_i's in the batch get assigned to a fresh random σ_i's from the pool.

Plug into federated BERT fine-tuning. Algorithm 2 shows how to incorporate (m, k)-*TextHide* in federated learning, to allow a centralized server and C distributed clients collaboratively fine-tune a language model (e.g., BERT) for any downstream tasks, without sharing raw data. Each client (indexed by c) holds its own private data $\mathcal{D}_c$ and a private mask pool $\mathcal{M}_c$, and $\sum_{c=1}^{C} |\mathcal{M}_c| = m$.

The procedure takes a pre-trained BERT f_{θ_1} and an initialized task-specific classifier h_{θ_2}, and runs T steps of *global* updates of both θ_1 and θ_2. In each *global* update, the server aggregates *local* updates of C clients. For a *local* update at client c, the client receives the latest copy of f_{θ_1} and h_{θ_2} from the server, samples a random input batch $\{x_i, y_i\}_{i \in [b]}$ from its private dataset $\mathcal{D}_c$, and encodes it into an encoding batch $\mathcal{E} = \{e_i = f_{\theta_1}(x_i), y_i\}_{i \in [b]}$ (line 21 in Algorithm 2).

To protect privacy, each client will run (m, k)-*TextHide* with its own mask pool $\mathcal{M}_c$ to encrypt the encoding batch $\mathcal{E}$ into a hidden batch $\widetilde{\mathcal{E}}$ (line 22 in Algorithm 2). The client then uses the hidden batch $\widetilde{\mathcal{E}}$ to calculate the model updates (i.e., gradients) of both the BERT encoder f_{θ_1} and the shallow classifier h_{θ_2}, and returns them to the server (line 23 in Algorithm 2). The server averages all updates from C clients, and runs a *global* update for f_{θ_1} and h_{θ_2} (lines 12, 13 in Algorithm 2).

Algorithm 2 Federated fine-tuning BERT using (m, k)-*TextHide* with C clients (indexed by c)

1: m: size of each client's mask pool
2: k: number of training samples to be mixed
3: d: hidden size (e.g., 768 in BERT)
4: **procedure** SERVEREXECUTION($f_{\theta_1}, h_{\theta_2}$)
5: ▷ f_{θ_1}: the pre-trained BERT; h_{θ_2}: a shallow classifer
6: ▷ T: number of model updates, η: learning rate
7: $f_{\theta_1^1} \leftarrow f_{\theta_1}, h_{\theta_2^1} \leftarrow h_{\theta_2}$
8: **for** $t = 1 \rightarrow T$ **do**
9: **for** each client c **in parallel do**
10: $\nabla_{\theta_1^t, c}, \nabla_{\theta_2^t, c} \leftarrow$ CLIUPDATE($c, f_{\theta_1^t}, h_{\theta_2^t}$)
11: **end for**
12: $\theta_1^{t+1} \leftarrow \theta_1^t - \frac{\eta}{C} \sum_{c=1}^{C} \nabla_{\theta_1^t, c}$
13: $\theta_2^{t+1} \leftarrow \theta_2^t - \frac{\eta}{C} \sum_{c=1}^{C} \nabla_{\theta_2^t, c}$
14: **end for**
15: **return** $f_{\theta_1}^{T+1}, h_{\theta_2}^{T+1}$
16: **end procedure**
17: **procedure** CLIUPDATE($c, f_{\theta_1}, h_{\theta_2}$) ▷ Run on Client c
18: ▷ b: batch size; $\mathcal{D}^c$: private train set of client c
19: ▷ $\mathcal{M}_c$: the mask pool of size m owned by client c, masks are sampled i.i.d. from $\{-1, +1\}^d$
20: Sample a random batch $\{x_i, y_i\}_{i \in [b]}$ from $\mathcal{D}_c$
21: $\mathcal{E} = \{f_{\theta_1}(x_i), y_i\}_{i \in [b]}$
22: $\widetilde{\mathcal{E}} \leftarrow$ *TextHide*$(\mathcal{E}, \mathcal{M}_c, k)$
23: **return** $\nabla_{\theta_1} \mathcal{L}(f_{\theta_1}, h_{\theta_2}; \widetilde{\mathcal{E}}), \nabla_{\theta_2} \mathcal{L}(f_{\theta_1}, h_{\theta_2}; \widetilde{\mathcal{E}})$
24: **end procedure**

3.3 Inter-dataset *TextHide*

Inter-dataset *TextHide* encrypts private inputs with text data from a *second* dataset, which can be a large public corpus (e.g., Wikipedia). The large public corpus plays a role reminiscent of the *random oracle* in cryptographic schemes (Canetti et al., 2004).

Assume we have a private dataset D_{private} and a large public dataset D_{public}, *TextHide*$_{\text{inter}}$ randomly chooses $\lceil k/2 \rceil$ sentences from D_{private} and the other $\lfloor k/2 \rfloor$ from D_{public}, mixes their representations, and applies on it a random mask from the pool. A main difference between *TextHide*$_{\text{inter}}$ and *TextHide*$_{\text{intra}}$ is, *TextHide*$_{\text{intra}}$ mixes all labels of inputs used in the combination, while in *TextHide*$_{\text{inter}}$, only the labels from D_{private} will be mixed (there is usually no label from the public dataset). Specifically, for an original datapoint $\{x_i, y_i\} \in \mathcal{E}$, let $S \subset [b]$ denote the set of data points' indices that its *TextHide* encryption combines, and $|S| = k$. Then its *TextHide*$_{\text{inter}}$ label is given by

$$\frac{\sum_{j=1}^{k} (\lambda_i)_j \cdot y_{\pi_j(i)} \cdot \mathbf{1}[\pi_j(i) \in D_{\text{private}} \cap S]}{\sum_{j=1}^{k} (\lambda_i)_j \cdot \mathbf{1}[\pi_j(i) \in D_{\text{private}} \cap S]},$$

where $\mathbf{1}[f]$ is a variable that $\mathbf{1}[f] = 1$ if f holds, and $= 0$ otherwise. For each $j \in [k]$, $\pi_j : [b] \rightarrow [b]$ is a permutation.

3.4 On Security of *TextHide*

The encrypted representations produced by *TextHide* themselves are secure — i.e., do not allow any efficient way to recover the text x — from the security framework of *InstaHide* (see Appendix A for k-VECTOR SUBSET SUM). However, an additional source of information leakage is the shared gradients during federated learning, as shown by (Zhu et al., 2019). We mitigate this by ensuring that the secret mask σ used to encrypt the representation of input x is changed each epoch. The pool of masks is usually much larger than the number of epochs, which means that each mask gets used only once for an input (with negligible failure probability). The gradient-matching attack of (Zhu et al., 2019) cannot work in this scenario. In the following section, we will show that it does not even work with a fixed mask.

4 Experiments

We evaluate the utility and privacy of *TextHide* in our experiments. We aim to answer the following questions in our experiments:

- What is the accuracy when using *TextHide* for sentence-level natural language understanding tasks (Section 4.2)?
- How effective is *TextHide* in terms of hiding the gradients (Section 4.3) and the representations of the original input (Section 4.4)?

4.1 Experimental Setup

Dataset. We evaluate *TextHide* on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019), a collection of 9 sentence-level language understanding tasks:

- Two *sentence-level classification* tasks including Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2019), and Stanford Sentiment Treebank (SST-2) (Socher et al., 2013).
- Three *sentence-pair similarity* tasks including Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), Semantic Textual Similarity Benchmark (STS-B) (Cer et al., 2017), and Quora Question Pairs (QQP)[6].
- Four *natural language inference* (NLI) tasks including Multi NLI (MNLI) (Williams et al., 2018), Question NLI (QNLI) (Rajpurkar et al., 2016), Recognizing Textual Entailment (RTE) (Dagan et al., 2005; Bar Haim et al., 2006; Giampiccolo et al., 2007), and Winograd NLI (WNLI) (Levesque et al., 2011).

Following previous work (Devlin et al., 2019; Joshi et al., 2020), we exclude WNLI in the evaluation. Table 1 summarizes the data size, tasks and evaluation metrics of all the datasets. All tasks are single-sentence or sentence-pair classification tasks except that STS-B is a regression task.

Implementation. We fine-tune the pre-trained cased $\text{BERT}_{\text{base}}$ model released by (Devlin et al., 2019) on each dataset. We notice that generalizing to different masks requires a more expressive classifier, thus instead of adding a linear classifier on top of the `[CLS]` token, we use a multilayer perceptron of hidden-layer size (768, 768, 768) to get better performance under *TextHide*. We use AdamW (Kingma and Ba, 2015) as the optimizer, and a linear scheduler with a warmup ratio of 0.1. More details of hyperparameter selection are given in Appendix B.3. To show *TextHide*'s compatibility with the state-of-the-art model, we also test with the $\text{RoBERTa}_{\text{base}}$ model released by (Liu et al., 2019) and report the results in Appendix B.2.

4.2 Accuracy Results of *TextHide*

To answer the first question, we compare the accuracy of *TextHide* to the BERT baseline without any encryption.

We the vary *TextHide* scheme as follows:

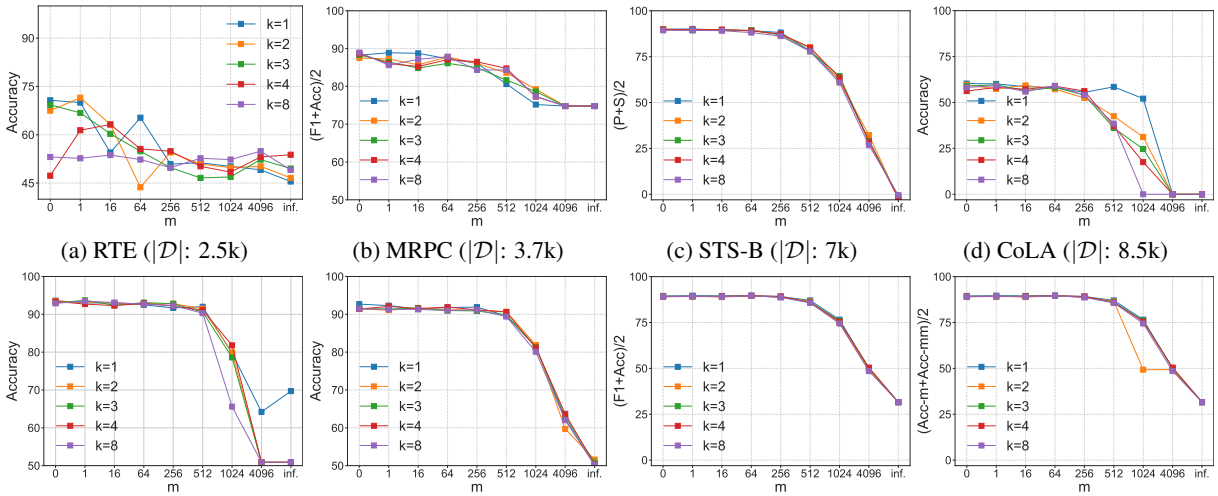
- Evaluate different (m, k) combinations, where m (the size of mask pool) is chosen from $\{0, 1, 16, 64, 256, 512, 1024, 4096, \infty\}$, and k (the number of inputs to combine) is chosen from $\{1, 2, 3, 4, 8\}$. $(m, k) = (0, 1)$ is equivalent to the baseline.
- Test both $\textit{TextHide}_{\text{intra}}$ and $\textit{TextHide}_{\text{inter}}$. We use MNLI train set (around 393k examples and all the labels are removed) as the "public dataset" in the inter-dataset setting and run BERT fine-tuning with $\textit{TextHide}_{\text{inter}}$ on the other 7 datasets. Here we use MNLI simply for convenience as it is the largest dataset in GLUE and one can use any public corpora (e.g., Wikipedia) in principle.

Results with different (m, k) pairs. Figure 2 shows the performance of $\textit{TextHide}_{\text{intra}}$ parameterized with different (m, k)'s. When m is fixed,

[6] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

Dataset	$\lvert\mathcal{D}\rvert$	Task	Metric	Baseline	*TextHide*$_{\text{intra}}$	*TextHide*$_{\text{inter}}$
RTE	2.5k	NLI	Acc.	$72.0_{(0.86)}$	$65.2_{(1.71)}$	$54.4_{(1.82)}$
MRPC	3.7k	Paraphrase	F1 / Acc.	$90.2_{(0.80)}$ / $86.2_{(1.40)}$	$89.7_{(0.56)}$ / $85.6_{(0.96)}$	$88.1_{(0.52)}$ / $82.6_{(0.75)}$
STS-B	7k	Similarity	P / S corr.	$90.1_{(0.12)}$ / $89.7_{(0.17)}$	$87.0_{(0.25)}$ / $87.0_{(0.27)}$	$86.0_{(0.27)}$ / $86.2_{(0.19)}$
CoLA	8.5k	Acceptability	MCC	$58.9_{(1.00)}$	$56.3_{(0.86)}$	$52.3_{(0.80)}$
SST-2	67k	Sentiment	Acc.	$92.4_{(0.76)}$	$91.7_{(0.51)}$	$91.3_{(0.41)}$
QNLI	108k	NLI	Acc.	$91.7_{(0.70)}$	$91.0_{(0.31)}$	$89.8_{(0.56)}$
QQP	364k	Paraphrase	F1 / Acc.	$87.9_{(0.39)}$ / $91.0_{(0.30)}$	$87.3_{(0.41)}$ / $90.5_{(0.33)}$	$86.5_{(0.28)}$ / $89.8_{(0.14)}$
MNLI	393k	NLI	m/mm	$86.1_{(0.36)}$ / $85.6_{(0.23)}$	$84.0_{(0.15)}$ / $84.1_{(0.23)}$	-

Table 1: Performance on the GLUE tasks for both baseline (standard finetuning) and *TextHide* with BERT$_{\text{base}}$, measured on the development sets. We report the mean results across 5 runs, with $(m, k) = (16, 4)$ for RTE and $(m, k) = (256, 4)$ for all the other datasets (see text for more details). Standard deviations are reported in parentheses. $\lvert\mathcal{D}\rvert$ denotes the number of training examples. *TextHide* only suffers minor utility loss: $< 3\%$ in most cases for both *TextHide*$_{\text{intra}}$and *TextHide*$_{\text{inter}}$. 'P / S corr.' is Pearson/Spearman correlation and 'MCC' is Matthew's correlation.



(a) RTE ($|\mathcal{D}|$: 2.5k) (b) MRPC ($|\mathcal{D}|$: 3.7k) (c) STS-B ($|\mathcal{D}|$: 7k) (d) CoLA ($|\mathcal{D}|$: 8.5k)

(e) SST-2 ($|\mathcal{D}|$: 67k) (f) QNLI ($|\mathcal{D}|$: 108k) (g) QQP ($|\mathcal{D}|$: 364k) (h) MNLI ($|\mathcal{D}|$: 393k)

Figure 2: Performance of *TextHide*$_{\text{intra}}$ on the GLUE tasks of different (m, k) pairs, measured on the development sets. $(m, k) = (0, 1)$ is equivalent to the baseline. Metrics are marked on the y-axis. $\lvert\mathcal{D}\rvert$ denotes the number of training examples. *TextHide* with $m = 256$ achieves good utility on all datasets (except RTE). Larger dataset can work with larger m.

the network performs consistently with different k's, suggesting that *MixUp* (Zhang et al., 2018a) also works for language understanding tasks.

Increasing m makes learning harder since the network needs to generalize to different masking patterns. However, for most datasets (except for RTE), *TextHide* with $m = 256$ only reduces accuracy slightly comparable to the baseline. Our explanation for the poor performance on RTE is that we find training on this small dataset (even without encryption) to be quite unstable. This has been observed in (Dodge et al., 2020) before. In general, *TextHide* can work with larger m (better security) when the training corpus is larger (e.g., $m = 512$ for data size > 100k).

TextHide$_{\text{intra}}$ **vs.** ***TextHide***$_{\text{inter}}$**.** *TextHide*$_{\text{intra}}$ mixes the representations from the same private dataset, whereas *TextHide*$_{\text{inter}}$ combines representations of private inputs with representations of random inputs from a large public corpus (MNLI in our case).

Table 1 shows the results of the baseline and *TextHide* (both *TextHide*$_{\text{intra}}$ and *TextHide*$_{\text{inter}}$) on the GLUE benchmark, with $(m, k) = (256, 4)$ except for RTE with $(m, k) = (16, 4)$. The averaged accuracy reduction of *TextHide*$_{\text{intra}}$ is 1.9%, when compared to the baseline model. With the same (m, k), *TextHide*$_{\text{inter}}$ incurs an additional 2.5% accuracy loss on average, but as previously suggested, the large public corpus gives a stronger notion of security.

4.3 Security of Gradients in *TextHide*

We test *TextHide* against the gradients matching attack in federated learning (Zhu et al., 2019), which has been shown effective in recovering private inputs from public gradients.

Gradients matching attack. Given a public model and the gradients generated by private data from a client, the attacker aims to recover the private data: he starts with some randomly initialized dummy data and dummy labels (i.e., a dummy batch). In each iteration of attack, he calculates the ℓ_2-distance between gradients generated by the dummy batch and the real gradients, and back-propagates that loss to update the dummy batch (see Algorithm 3 in Appendix C for details).

The original attack is infeasible in the *TextHide* setting, because the attacker can't backpropagate the loss of the dummy batch through the *secret mask* of each client. Thus, we enhance the attack by allowing the attacker to learn the mask: at the beginning of the attack, he also generates some dummy masks and back-propagates the loss of gradient to update them.

Setup and metric. We use the code[7] of the original paper (Zhu et al., 2019) for evaluation. Due to the unavailability of their code for attacks in text data, we adapted their setting for computer vision (see Appendix C for more details). We use the success rate as the metric: an attack is said to be successful if the mean squared error between the original input and the samples recovered from gradients is ≤ 0.001. We vary two key variables in the evaluation: k and d, where d is the dimensionality of the representation (768 for $\text{BERT}_{\text{base}}$).

Test the leakage upper bound. We run the attack in a much easier setting for the attacker to test the upper bound of privacy leakage:

- The *TextHide* scheme uses a single mask throughout training (i.e., $m = 1$).
- The batch size is 1.[8]
- The attacker knows the true label for each private input.[9]

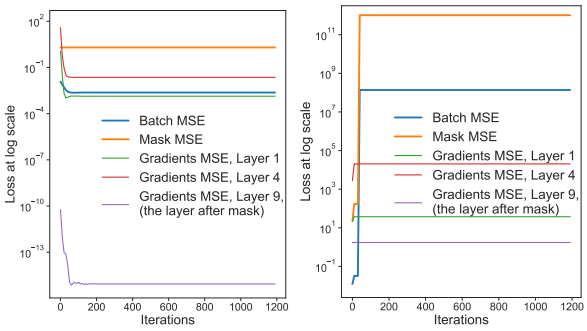
[7] https://github.com/mit-han-lab/dlg

[8] The original paper (Zhu et al., 2019) pointed out that attacking a larger batch is more difficult.

[9] As suggested by Zhao et al. (2020), guessing the correct label is crucial for success in the attack.

Baseline	k \ d	4	16	64	256	1024
0.82	1	0.76	0.56	0.30	0.22	0.08
	2	0.00	0.00	0.00	0.00	0.00
	4	0.00	0.00	0.00	0.00	0.00

Table 2: Success rate of 50 independent gradients matching attacks. Baseline is the vanilla architecture without *TextHide*. d: the dimensionality of the representation. Increasing k and d makes attack harder.



(a) Success (b) Failure

Figure 3: Loss over iterations of a succeeded (a) and a failed (b) attacks. When the mean square error (MSE) between real and dummy masks gets smaller, both the gradients' distance and the MSE between leaked image and the original image gets smaller.

***TextHide* makes gradients matching harder.** As shown in Table 2, increasing d, greatly increases the difficulty of attack — for no mixing ($k = 1$), a representation with $d = 1024$ reduces the success rate of 82% (baseline) to only 8%. The defense becomes much stronger when combined with mixing: a small mask of 4 entries combined with $k = 2$ makes the attack infeasible in the tested setting. Figure 3 suggests that the success of this attack largely depends on whether the mask is successfully matched, which is aligned with the security argument of *TextHide* in Section 3.4.

4.4 Effectiveness of Hiding Representations

We also design an attack-based evaluation to test whether *TextHide* representations effectively "hide" its original representations, i.e., how 'different' the *TextHide* representation is from its original representation. In Appendix C, we present another attack, which suggests that a deep architecture can not be trained to reconstruct the original representations from the *TextHide* representation.

Representation-based Similarity Search (RSS). Given a corpus of size n, and

1) a search index: $\{x_i, e_i\}_{i=1}^n$, where x_i is the i-th example in the training corpus, e_i is x_i's

	Baseline	*Mix-only*	*TextHide*	Rand
Identity	0.993	0.111	**0.000**	0.000
JC_{dist}	0.999	0.184	**0.023**	0.024
$\text{TF-IDF}_{\text{sim}}$	1.000	0.194	**0.015**	0.015
Label	0.998	0.759	**0.494**	0.542
$\text{SBERT}_{\text{sim}}$	0.991	0.280	**0.102**	0.101

(a) CoLA

	Baseline	*Mix-only*	*TextHide*	Rand
Identity	0.992	0.064	**0.000**	0.000
JC_{dist}	0.999	0.168	**0.100**	0.096
$\text{TF-IDF}_{\text{sim}}$	1.000	0.080	**0.007**	0.008
Label	1.000	0.714	**0.503**	0.501
$\text{SBERT}_{\text{sim}}$	1.000	0.275	**0.202**	0.209

(b) SST-2

Table 3: Averaged similarity score of five metrics over 1,000 independent RSS attacks on CoLA (a) and SST-2 (b). For each score, the scheme with the worst similarity (best hiding) is marked in **bold**. Rand: random baseline. As shown, attacker against *TextHide* gives similar performance to random guessing.

encoded representation $f_{\theta_1}(x_i)$;

2) a query $\widetilde{e}$: *TextHide* representation of any input x in the corpus,

RSS returns x_v from the index such that $v = \arg\min_{i\in[n]} \cos(e_i, \widetilde{e})$. If x_v is dramatically different from x, then $\widetilde{e}$ hides e (the original representation of x) effectively. To build the search index, we dump all (x_i, e_i) pairs of a corpus by extracting each sentence's `[CLS]` token from the baseline BERT model. We use Facebook's FAISS library (Johnson et al., 2017) for efficient similarity search to implement RSS.

Metrics. The evaluation requires measuring the similarity of a sentence pair, (x, x^*), where x is a sample in corpus, and x^* is RSS's answer given x's encoding $\widetilde{e}$ as query. Our evaluation uses three explicit leakage metrics:

- Identity: 1 if x^* is identical to x, else 0.
- JC_{dist}: Jaccard distance $|\text{words in } x \cap \text{words in } x^*|/|\text{words in } x \cup \text{words in } x^*|$
- $\text{TF-IDF}_{\text{sim}}$: cosine similarity between x's and x^*'s TF-IDF representation in the corpus

and two implicit (semantic) leakage metrics:

- Label: 1 if x^*, x have the same label, else 0.
- $\text{SBERT}_{\text{sim}}$: cosine similarity between x's and x^*'s SBERT representations pretrained on NLI-STS[10] (Reimers and Gurevych, 2019).

For all five metrics above, a larger value indicates a higher similarity between x and x^*, i.e., worse 'hiding'.

Query1 (CoLA): Some people consider the noisy dogs dangerous. (✓)
Baseline: Some people consider the noisy dogs dangerous. (✓) *Mix-only*: Some people consider the noisy dogs dangerous. (✓) *TextHide*: I know a man who hates myself. (×)
Query2 (SST-2): otherwise excellent (😊)
Baseline: otherwise excellent (😊) *Mix-only*: worthy (😊) *TextHide*: passive-aggressive (☹)

Table 4: Example queries and answers of RSS with different representation schemes. We mark words with similar meanings in the same color. We annotate the acceptability for CoLA ('✓': yes, '×': no) and sentiment for SST-2 ('😊': positive, '☹': negative). Querying with a *Mix-only* representation still retrieve the original sentence (Query1), or sentence with similar meanings (Query2).

Test Setup. For an easier demonstration, we run RSS on two single-sentence datasets CoLA and SST-2 with $\textit{TextHide}_{\text{intra}}$. The results presumably can generalize to larger datasets and $\textit{TextHide}_{\text{inter}}$, since attacking a small corpus with a weaker security is often easier than attacking a larger one with a stronger security. For each task, we test three (m, k) variants: baseline $(m = 0, k = 1)$, mix-only $(m = 0, k = 4)$, and *TextHide* $(m = 256, k = 4)$. We report a random baseline for reference — for each query, the attacker returns an input randomly selected from the index.

Baseline. The result with original representation as query can be viewed as an upper bound of privacy leakage where no defense has been taken. As shown in Table 3 and Table 4, RSS almost returns the correct answer all the time (i.e., *Identity* close to 1), which is a severe explicit leakage.

Mix-only. *Mix-only* representation greatly reduces both explicit leakage (i.e., gives much lower similarity on all first 3 metrics) compared to the undefended baseline. However, RSS still can query back the original sentence with Mix-only representations (see Query1 in Table 4).

[10] We use SBERT as an off-the-shelf similarity scorer since it has been demonstrated great performance in semantic textual similarity tasks.

Also, semantic leakage, measured by *Label* and $SBERT_{\text{sim}}$, is higher than the random baseline.

TextHide *TextHide* works well in protecting both explicit and semantic information: sample attacks on *TextHide* (see Table 4) return sentences seemingly irrelevant to the original sentence hidden in the query representation. Note that the sophisticated attacker (RSS) against *TextHide* gives similar performance to a naive random guessing attacker.

5 Related Work

Differential privacy. Differential privacy (Dwork et al., 2006; Dwork and Roth, 2014) adds noise drawn from certain distributions to provide guarantees of privacy. Applying differential privacy techniques in distributed deep learning is interesting but non-trivial. Shokri and Shmatikov (2015) proposed a distributed learning scheme by directly adding noise to the shared gradients. Abadi et al. (2016) proposed to dynamically keep track of privacy spending based on the composition theorem (Dwork, 2009), and McMahan et al. (2018) adapted this approach to train large recurrent language models. However, the amount of privacy guaranteed drops with the number of training epochs and the size of shared parameters (Papernot et al., 2020), and it remains unclear how much privacy can still be guaranteed in practical settings.

Cryptographic methods. Homomorphic encryption (Gentry, 2009; Graepel et al., 2012; Li et al., 2017) or secure multi-party computation (MPC) (Yao, 1982; Beimel, 2011; Mohassel and Zhang, 2017; Dolev et al., 2019) allow multiple data cites (clients) to jointly train a model over their private inputs in distributed learning setting. Recent work proposed to use cryptographic methods to secure federated learning by designing a secure gradients aggregation protocol (Bonawitz et al., 2017) or encrypting gradients (Aono et al., 2017). However, these approaches shared the same key drawback: slowing down the computation by several orders of magnitude, thus currently impractical for deep learning.

***InstaHide*.** See Section 2.

Privacy in NLP. Training with user-generated language data raises privacy concerns: sensitive information can take the form of key phrases explicitly contained in the text (Harman et al., 2012; Hard et al., 2018); it can also be implicit (Coavoux et al., 2018; Pan et al., 2020), e.g., text data contains latent information about the author and situation (Hovy and Spruit, 2016; Elazar and Goldberg, 2018). Recently, Song and Raghunathan (2020) suggests that text embeddings from language models such as BERT can be inverted to partially recover some of the input data.

To deal with explicit privacy leakage in NLP, Zhang et al. (2018b) added DP noise to TF-IDF (Salton and McGill, 1986) textual vectors, and Hu et al. (2020) obfuscated the text by substituting each word with a new word of similar syntactic role. However, both approaches suffer large utility loss when trying to ensure practical privacy.

Adversarial learning (Li et al., 2018; Hu et al., 2020) has been used to address implicit leakage to learn representations that are invariant to private-sensitive attributes. Similarly, Mosallanezhad et al. (2019) used reinforcement learning to automatically learn a strategy to reduce private-attribute leakage by playing against an attribute-inference attacker. However, these approaches does not defend explicit leakage.

6 Conclusion

We have presented *TextHide*, a practical approach for privacy-preserving NLP training with a pre-train and fine-tuning framework in a federated learning setting. It requires all participants to add a simple encryption step with an one-time secret key. It imposes a slight burden in terms of computation cost and accuracy. Attackers who wish to break such encryption and recover user inputs have to pay a large computational cost.

We see this as a first step in using cryptographic ideas to address privacy issues in language tasks. We hope our work motivates further research, including applications to other NLP tasks. An important step could be to successfully train language models directly on encrypted texts, as is done for image classifiers.

Acknowledgements

This project is supported in part by the Graduate Fellowship at Princeton University, Ma Huateng Foundation, Schmidt Foundation, Simons Foundation, NSF, DARPA/SRC, Google and Amazon AWS. Arora and Song were at the Institute for Advanced Study during this research.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318.

Amir Abboud and Kevin Lewi. 2013. Exact weight subgraphs and the k-sum conjecture. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–12.

Amir Abboud, Kevin Lewi, and Ryan Williams. 2014. Losing weight by gaining edges. In *European Symposium on Algorithms (ESA)*, pages 1–12.

Accountability Act. 1996. Health insurance portability and accountability act of 1996. *Public law*, 104:191.

Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge. In *Manuscript*.

Amos Beimel. 2011. Secret-sharing schemes: a survey. In *International conference on coding and cryptology (IWCC)*, pages 11–46.

Arnab Bhattacharyya, Piotr Indyk, David P Woodruff, and Ning Xie. 2011. The complexity of linear dependence problems in vector spaces. In *Innovations in Computer Science (ICS)*, pages 496–508.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1175–1191.

Ran Canetti, Oded Goldreich, and Shai Halevi. 2004. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL)*, pages 1–14.

Maximin Coavoux, Shashi Narayan, and Shay B Cohen. 2018. Privacy-preserving neural representations of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–10.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment (MLCW)*, pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the International Workshop on Paraphrasing*.

Shlomi Dolev, Peeyush Gupta, Yin Li, Sharad Mehrotra, and Shantanu Sharma. 2019. Privacy-preserving secret shared computations using mapreduce. *IEEE Transactions on Dependable and Secure Computing*.

Cynthia Dwork. 2009. The differential privacy frontier. In *Theory of Cryptography Conference (TCC)*, pages 496–502.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503.

Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 11–21.

Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC)*, pages 169–178.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing (IWP)*, pages 1–9.

Thore Graepel, Kristin Lauter, and Michael Naehrig. 2012. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology (ICISC)*, pages 1–21.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

Laurinda B Harman, Cathy A Flite, and Kesa Bond. 2012. Electronic health records: privacy, confidentiality, and security. *AMA Journal of Ethics*, 14(9):712–719.

Dirk Hovy and Shannon L Spruit. 2016. The social impact of natural language processing. In *Association for Computational Linguistics (ACL)*, pages 591–598.

Zhifeng Hu, Serhii Havrylov, Ivan Titov, and Shay B Cohen. 2020. Obfuscation for privacy-preserving syntactic parsing. In *The 16th International Conference on Parsing Technologies (IWPT)*, pages 62–72.

Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. 2020. Instahide: Instance-hiding schemes for private distributed learning. In *Internation Conference on Machine Learning (ICML)*.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 1998. Which problems have strongly exponential complexity? In *Proceedings. 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 653–662.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. In *Transactions of the Association of Computational Linguistics (TACL)*, pages 64–77.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

California State Legislature. 2018. California consumer privacy act (ccpa). https://oag.ca.gov/privacy/ccpa.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47.

Ping Li, Jin Li, Zhengan Huang, Tong Li, Chong-Zhi Gao, Siu-Ming Yiu, and Kai Chen. 2017. Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*, 74:76–85.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Association for Computational Linguistics (ACL)*, pages 25–30.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. year. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning differentially private recurrent language models. In *International Conference on Learning Representations (ICLR)*.

Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (S&P)*, pages 19–38.

Ahmadreza Mosallanezhad, Ghazaleh Beigi, and Huan Liu. 2019. Deep reinforcement learning-based text anonymization against private-attribute inference. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2360–2369.

Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (S&P)*, pages 1314–1331.

Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. 2020. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy. In *Manuscript*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035.

Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. 2017. Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of Biomedical Informatics*, 69:218 – 229.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese bert-networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Gerard Salton and Michael J McGill. 1986. Introduction to modern information retrieval. *McGraw-Hil, New York*.

Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (CCS)*, pages 1310–1321.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Congzheng Song and Ananth Raghunathan. 2020. Information leakage in embedding models. In *ACM Conference on Computer and Communications Security (CCS)*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. In *Transactions of the Association of Computational Linguistics (TACL)*, pages 625–641.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's transformers: State-of-the-art natural language processing. In *arXiv preprint*.

Cao Xiao, Edward Choi, and Jimeng Sun. 2018. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 25(10):1419–1428.

A. C. Yao. 1982. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018a. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*.

Jinxue Zhang, Jingchao Sun, Rui Zhang, Yanchao Zhang, and Xia Hu. 2018b. Privacy-preserving social media data outsourcing. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 1106–1114.

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

Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*.

A k-VECTOR SUBSET SUM

Cryptosystem design since the 1970s seeks to ensure that attackers can violate privacy only by solving a computationally expensive task. A simple example is the VECTOR SUBSET SUM problem (Bhattacharyya et al., 2011; Abboud et al., 2014). Here a set of N vectors $v_1, v_2, \ldots, v_k \in \mathbb{R}^d$ are publicly released. The defender picks secret indices $i_1, i_2, \ldots, i_k \in [N] \stackrel{\text{def}}{=} \{1, \cdots, N\}$ and publicly releases the vector $\sum_j v_{i_j}$. Given this released vector the attacker has to find secret indices $i_1, i_2, \ldots, i_k$. In worst cases even when the answer happens to be unique, finding the secret indices requires $\geq N^{k/2}$ time (Abboud and Lewi, 2013) under the famous conjecture, Exponential Time Hypothesis (ETH) (Impagliazzo et al., 1998). Note that ETH is a stronger notion than NP $\neq$ P, and ETH is widely accepted computational complexity community.

B Experiment details

B.1 Implementation

The implementation uses the PyTorch framework (Paszke et al., 2019) based on HuggingFace's codebase (Wolf et al., 2019). We ran all experiments on 24 NVIDIA RTX 2080 Ti GPUs.

B.2 More Evaluations

Compatibility with the state-of-the-art model. To test if *TextHide* is also compatible with state-of-the-art models, we repeat our accuracy evaluation in Section 4.2 but replace the $\text{BERT}_{\text{base}}$ model with the $\text{RoBERTa}_{\text{base}}$ model (Liu et al., 2019).

As shown in Table 5, *TextHide* behaves consistently for $\text{BERT}_{\text{base}}$ and $\text{RoBERTa}_{\text{base}}$: when incorporated with $\text{RoBERTa}_{\text{base}}$, the averaged accuracy reduction of $\textit{TextHide}_{\text{intra}}$ is 1.1% when compared with the baseline model (was 1.9% for $\text{BERT}_{\text{base}}$). $\textit{TextHide}_{\text{inter}}$ incurs an additional 2.6% accuracy loss on average (was 2.5% for $\text{BERT}_{\text{base}}$).

$\textit{TextHide}_{\text{inter}}$ with different public corpora: A case study of SST-2. We investigate whether using different public corpora affects the performance of $\textit{TextHide}_{\text{inter}}$. We fix SST-2 as the private dataset, set $(m, k) = (256, 4)$, and choose the public corpora from *unlabeled* {QNLI, QQP, MNLI}. We intentionally make the public corpora larger than the private dataset (SST-2 in this test), since $\textit{TextHide}_{\text{inter}}$ was designed to use a *large* public corpus as the source of randomness to provide useful security.

Table 6 suggests that for our case study of SST-2, the choice of the public corpus does *not* have a major impact on the final accuracy of $\textit{TextHide}_{\text{inter}}$. However, this may not be true for every dataset.

Algorithm 3 Gradients matching attack (Zhu et al., 2019) in *TextHide*

1: **Require :**
2: The function $F(x; W)$ can be thought of as a neural network
3: For each $l \in [L]$, we define $W_l \in \mathbb{R}^{m_l \times m_{l-1}}$ to be the weight matrix in l-th layer, and $m_0 = d_i$ and $m_l = d_o$
4: Let $W = \{W_1, W_2, \cdots, W_L\}$ denote the weights over all layers
5: Let $\mathcal{L} : \mathbb{R}^{d_o \times d_o} \to \mathbb{R}$ denote loss function
6: Let $g(x, y) = \nabla \mathcal{L}(F(x; W), y)$ denote the gradients of loss function
7: Let $\widehat{g} = g(\sigma, x, y)|_{\sigma=\sigma_0, x=x_0, y=y_0}$ denote the gradients computed on x_0 with label y_0, and secret mask σ_0
8: **procedure** INPUTRECOVERYFROMGRADIENTS
9: $x^{(1)} \leftarrow \mathcal{N}(0, 1)$, $y^{(1)} \leftarrow \mathcal{N}(0, 1)$, $\sigma^{(1)} \leftarrow \mathcal{N}(0, 1)$ ▷ Random initialization of the input, label and mask
10: **for** $t = 1 \to T$ **do**
11: Let $D_g(\sigma, x, y) = \|g(\sigma, x, y) - \widehat{g}\|_2^2$
12: $x^{(t+1)} \leftarrow x^{(t)} - \eta \cdot \nabla_x D_g(\sigma, x, y)|_{x=x^{(t)}}$
13: $y^{(t+1)} \leftarrow y^{(t)} - \eta \cdot \nabla_y D_g(\sigma, x, y)|_{y=y^{(t)}}$
14: $\sigma^{(t+1)} \leftarrow \sigma^{(t)} - \eta \cdot \nabla_y D_g(\sigma, x, y)|_{\sigma=\sigma^{(t)}}$
15: **end for**
16: **return** $x^{(T+1)}, y^{(T+1)}, \sigma^{(T+1)}$
17: **end procedure**

B.3 Fine-tuning Hyperparameters

For results in Table 1 and 5 (including our baseline), we chose the best parameters with learning rate = $\{5e{-}6, 1e{-}5, 2e{-}5, 3e{-}5, 5e{-}5\}$, epochs = $\{5, 10, 15, 20, 25, 30\}$, batch size = $\{16, 32\}$, dropout rate = $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ based on the validation performance (10% from the training set). We used more epochs for fine-tuning since training with random masking takes longer to converge.

C Details of attacks

C.1 Gradients matching attack

Algorithm 3 describes the gradients matching attack (Zhu et al., 2019) in TextHide setting. This attack aims to recover the original image from model gradients computed on it. As discussed in Section 2, masks are kept private in TextHide setting, thus the attacker also need to start from a dummy mask (line 9) and iteratively update it to compromise the real mask (line 14). In our experiment, we made this attack much easier for the

Datasets	$\lvert\mathcal{D}\rvert$	Task	Metric	Baseline	*TextHide*$_{\text{intra}}$	*TextHide*$_{\text{inter}}$
RTE	2.5k	NLI	Acc.	$78.8_{(0.69)}$	$77.9_{(0.99)}$	$70.8_{(0.78)}$
MRPC	3.7k	Paraphrase	F1 / Acc.	$92.3_{(0.56)}$ / $89.3_{(0.66)}$	$91.1_{(0.68)}$ / $87.6_{(0.34)}$	$90.5_{(0.68)}$ / $87.1_{(0.89)}$
STS-B	7k	Similarity	P / S corr.	$91.3_{(0.13)}$ / $91.0_{(0.19)}$	$90.4_{(0.19)}$ / $90.3_{(0.16)}$	$82.6_{(0.65)}$ / $84.2_{(0.52)}$
CoLA	8.5k	Acceptability	MCC	$63.0_{(1.24)}$	$59.1_{(1.01)}$	$57.2_{(0.90)}$
SST-2	67k	Sentiment	Acc.	$94.1_{(0.52)}$	$93.5_{(0.21)}$	$92.8_{(0.47)}$
QNLI	108k	NLI	Acc.	$92.7_{(0.21)}$	$92.3_{(0.29)}$	$91.7_{(0.48)}$
QQP	364k	Paraphrase	F1 / Acc.	$88.8_{(0.21)}$ / $91.6_{(0.15)}$	$88.1_{(0.24)}$ / $91.0_{(0.31)}$	$87.7_{(0.36)}$ / $90.7_{(0.22)}$
MNLI	393k	NLI	m/mm	$87.2_{(0.39)}$ / $86.8_{(0.21)}$	$86.4_{(0.21)}$ / $86.0_{(0.15)}$	-

Table 5: Performance on the GLUE tasks for both baseline (standard finetuning) and *TextHide* with RoBERTa$_{\text{base}}$ model (Liu et al., 2019), measured on the development sets. We report the mean results across 5 runs, with $(m, k) = (16, 4)$ for RTE and $(m, k) = (256, 4)$ for all the other datasets. Standard deviations are reported in parentheses. $|\mathcal{D}|$ denotes the number of training samples. *TextHide* only suffers minor utility loss ($\sim 3\%$). 'P / S corr.' is Pearson/Spearman correlation. 'MCC' is Matthew's correlation.

Private dataset: SST-2 ($\lvert\mathcal{D}\rvert$: 67k)			
Public Corpora	$\lvert\mathcal{D}\rvert$	**Task**	**Acc.**
QNLI	108k	NLI	$91.2_{(0.68)}$
QQP	364k	Paraphrase	$91.0_{(0.45)}$
MNLI	393k	NLI	$91.3_{(0.41)}$

Table 6: Dev set performance of SST-2 for *TextHide*$_{\text{inter}}$ with different public corpora, $(m, k) = (256, 4)$. $|\mathcal{D}|$ denotes the number of samples. Standard deviations are annotated as subscripts. The choice of the public corpus does not have a major impact on the final accuracy of SST-2.

Q1(CoLA): The magazines were sent to herself by Mary. (×)
Baseline: The magazines were sent to herself by Mary. (×) Mix-only: The company sent China its senior mining engineers to help plan the new mines. (×) TextHide: Hierarchy of Projections: (✓)
Q2(SST-2): an exquisitely crafted and acted tale. (😊)
Baseline: an exquisitely crafted and acted tale. (😊) Mix-only: to make their way through this tragedy (☹) TextHide: fails to live up to – or offer any new insight into – its chosen topic (☹)

Table 7: Example queries and answer of RepRecon with different representation schemes. Words with similar meanings are marked in the same color. For CoLA examples, we annotate the acceptability ('✓' for yes, '×' for no); for SST-2 examples, we annotate sentiment ('😊' for positive, '☹' for negative).

attacker, by revealing to him the real ground truth label (y_0 in line 7), which means he simply sets $y^{(t)} = y_0$ throughout the attack.

Dataset and architecture. We used CIFAR-10 (Krizhevsky, 2009) as the dataset and LeNet-5 (LeCun et al., 1998) as the architecture to mimic *TextHide*.

	Baseline	Mix-only	TextHide	Rand
ID	0.982	0.002	**0.000**	0.000
JC$_{\text{dist}}$	0.992	0.033	**0.029**	0.028
TF-IDF$_{\text{sim}}$	0.993	0.018	**0.014**	0.018
Label	0.998	0.818	**0.638**	0.620
SBERT$_{\text{sim}}$	0.994	0.111	**0.051**	0.104

(a) RepRecon, CoLA

	Baseline	Mix-only	TextHide	Rand
ID	0.948	**0.000**	**0.000**	0.000
JC$_{\text{dist}}$	0.953	0.065	**0.064**	0.080
TF-IDF$_{\text{sim}}$	0.949	0.019	**0.013**	0.014
Label	0.968	0.464	**0.472**	0.452
SBERT$_{\text{sim}}$	0.959	0.268	**0.266**	0.211

(b) RepRecon, SST-2

Table 8: Similarity score of five metrics for RepRecon on CoLA (a) and SST-2 (b) datasets. We report the average score over 500 independent queries. Test queries come from only the dev set. For each score, the scheme with the worst similarity (best hiding) is marked in **bold**. As shown, attacker against *TextHide* gives similar performance to random guessing.

Given the original LeNet-5, we firstly removed the last linear layer with output size d_o, which gives us a new network. We use d_c to denote the size of output in the new network. Then, we appended an MLP with hidden-layer size d_m and output size d_o to the new architecture. As in an (m, k)-*TextHide* scheme, for each private input, we first gets its *TextHide* representations by extracting the output from the hidden-layer, and mixes it with representations of other datapoints. We then apply a mask on this combination. Note: in this mimic setting, the mask's dimension is d_m.

Hyper-parameters and running-time. Following (Zhu et al., 2019), we use L-BFGS (Liu and Nocedal, 1989) optimizer (learning-rate 1, history-size 100 and max-iterations 20) and optimize for 1,200 iterations. Each run takes 97 seconds (single V100 GPU, averaged across 20 runs).

C.2 Representation-based Similarity Search (RSS)

Running-time. For CoLA, building the search index takes 267 seconds; each search takes < 0.1 seconds. For SST-2, building the index takes $1,576$ seconds; each search takes < 0.1 seconds.

C.3 Representation Reconstruction (RepRecon)

RepRecon tests whether a deep architecture can learn to disrupt our 'hiding' scheme. For an representation $e \in \mathbb{R}^d$, and its *TextHide* version $\widetilde{e} \in \mathbb{R}^d$, RepRecon tries to reconstruct e from $\widetilde{e}$ by training a network $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that $\|e - f(\widetilde{e})\|_2$ is minimized.

We use a multi-layer perception of hidden-layer size (1024, 1024) as the reconstruction architecture. We train the network on the train set of a benchmark for 20 epochs, and run evaluation using the dev set. We then run RSS to map the recovered representation to its closet sentence in the index, and measure the privacy leakage.

Quantitative and qualitative results of RepRecon are shown in Table 8 and Table 7.