Hush! Protecting Secrets During Model Training: An Indistinguishability Approach

Arun Ganesh* Brendan McMahan† Milad Nasr‡ Thomas Steinke§
Abhradeep Thakurta¶

June 7, 2025

Abstract

We consider the problem of *secret protection*, in which a business or organization wishes to train a model on their own data, while attempting to not leak secrets potentially contained in that data via the model. The standard method for training models to avoid memorization of secret information is via differential privacy (DP). However, DP requires a large loss in utility or a large dataset to achieve its strict privacy definition, which may be unnecessary in our setting where the data curator and data owner are the same entity. We propose an alternate definition of secret protection that instead of targeting DP, instead targets a bound on the posterior probability of secret reconstruction. We then propose and empirically evaluate an algorithm for model training with this secret protection definition. Our algorithm solves a linear program to assign weights to examples based on the desired per-secret protections, and then performs Poisson sampling using these weights. We show our algorithm significantly outperforms the baseline of running DP-SGD on the whole dataset.

1 Introduction

We consider the problem of an organization training a model on an internal dataset containing some number of secrets. For example, consider an algorithmic trading company. They company could be interested in fine-tuning an internal code completion model used by all employees on their codebase because they are using their own internal libraries that a public code completion model would have no knowledge of. However, their code base likely contains information that they would not like their entire employee base to have access to, and hence want to prevent the model from memorizing. For example, they may have many teams working on different trading algorithms for different markets, and keep different teams' codebases and knowledge siloed to reduce the risk of leaking any of their trade secrets which give them a competitive advantage. In this case, if they trained the code completion model without any safeguards, it's possible the model could be prompted by any employee to reproduce code they should not be able to access. While the definitions and techniques in this paper could be applied broadly beyond this example, to keep the discussion concrete we will focus on this example throughout the paper.

A standard technique for preventing memorization of sensitive information is differential privacy (DP) [DMNS06], and we could use an algorithm like DP-SGD [SCS13, BST14] or its variants, which clip gradients to a bounded ℓ_2-norm and then add noise to prevent memorization of specific examples. To protect the secrets in a dataset, we could target e.g. 'secret-level' DP, i.e. ensure that the outputs of training on the dataset and a copy of the dataset with each example containing a certain secret being changed arbitrarily satisfy some indistinguishability guarantee. However, DP gives strong protections necessary for a setting where a data curator is using external data belonging to third-party users. The example trading company is both the model trainer and the owner of the codebase they are training on,

*Google Research arunganesh@google.com
†Google Research mcmahan@google.com
‡Google DeepMind. srxzr@google.com
§Google DeepMind steinke@google.com
¶Google DeepMind athakurta@google.com

hence these strong protections are unnecessary. In this paper, we propose alternatives to DP for defining formal secret protection while training models, propose methods for achieving these definitions, and give empirical evaluations of the practicality of these goals methods.

1.1 Our Contributions

Our main contributions are a definition of secret protection that is more suitable for the threat model we study, and algorithms tailored to this definition.

A definition for secret protection: In Section 3 we give our proposed definition of secret protection. Our definition differs from standard DP guarantees in the following ways: (i) It gives each secret a different degree of protection, rather than requiring a uniform protection (ii) It assumes the existence of secrets is public information, rather than trying to keep membership private (iii) It does not allow examples to arbitrarily change for two adjacent datasets.

Our definition bounds the posterior reconstruction probability for each secret given a prior reconstruction probability. While tools from [HBM23] do this tightly if we can exactly compute the privacy loss distribution of our algorithm, in our empirical setting they suffer from numerical instability issues. We found divergence bounds to be more robust to these issues, and give a divergence-based method for bounding reconstruction probabilities (Lemma 3.2) that is tight for a given divergence bound. This improves on the loose Rényi-divergence-based reconstruction bounds of [BCH22].

Algorithms tailored to our new definition: We next study algorithms for model training with secret protection. The main algorithmic challenge is that given a dataset, even if each secret has a uniform privacy guarantee we are targeting, some secrets may appear more frequently in the dataset and require more noise. Instead, we would like to bound the number of appearances of each secret in the dataset, so that each secret appears a number of times roughly proportional to its desired privacy guarantee. We propose a simple but novel algorithm for selecting a subset of the dataset to use in training that achieves this property. At a high level, our algorithm uses a linear program to assign each example a fractional weight, and uses these weights to choose sampling probabilities for each example when forming batches. We empirically demonstrate that our algorithm allows us to achieve a $\approx 8\times$ reduction in the noise and $\approx 7.5\%$ reduction in test loss compared to not doing any preprocessing of the dataset.

1.2 Related Work

The observation that DP even with weak privacy parameters protects against reconstruction and suggestion to relax the DP parameters when membership inference is not a concern are well-known. [BDF+19], motivated by this observation, give minimax optimal local ε-DP mechanisms in the high-ε regime for noising vectors, as opposed to past work which only considered the low-ε regime. [HBM23] give upper bounds on the probability of reconstructing an example included in DP-SGD, and an attack which gives a nearly matching empirical lower bound. They also show different algorithms with the same approximate DP parameters can give very different protections against reconstruction, showing standard DP definitions are not good predictors of protection against reconstruction. As we will discuss later in Section 3.3, the bounds of [HBM23] are tight but can require highly numerically unstable computations in some cases.

A key component of our definition of secret protection is that we offer different privacy protections to different secrets, whereas most of the DP literature studies uniform privacy protections. [YKK+23] observed that DP-SGD often provides different privacy protections to different individuals. However, this is an artifact of average-case training behaviors that occur in practice, and in their work DP-SGD still is implemented to target a uniform worst-case privacy guarantee, as opposed to our algorithms which explicitly target non-uniform worst-case protections.

Our definition of secret protection is somewhat analogous to user-level DP, where the privacy unit is all examples owned by a single user. How to fit a standard algorithm like DP-SGD to user-level DP is well-studied in the setting where each example is owned by a single user [CGM+24, CGH+24]. Across the broader DP literature, the setting where an example can be owned by multiple users has been indirectly studied in algorithms for queries on graphs. For example, [DFY+22] study the problem of graph pattern counting under node-level DP. Here, each instance of a graph pattern can be viewed as an example belonging to all the nodes in it. [GMM+25] recently proposed and studied a *multi-attribution* DP definition, similar to node-level DP but where the graph is assumed to be public, and with a focus on model training rather than graph queries.

Algorithm 1 DP-SGD with Poisson sampling

Inputs: Dataset $D = \{x_1, \ldots, x_n\}$, loss ℓ, optimizer update function $update$, secrets S, number of rounds T, target batch size B, clip norm C, noise multiplier σ, initial model θ_0.

1: $\mathsf{clip}(\mathbf{v}, C) := \mathbf{v} \cdot \min\{1, C/\|\mathbf{v}\|_2\}$
2: **for** $t \in [T]$ **do**
3: $\quad \mathcal{B}_i \leftarrow$ batch including x_i independently w.p. p.
4: $\quad \mathbf{g}_i \leftarrow \frac{1}{B} \sum_{x_i \in \mathcal{B}_i} \nabla \mathsf{clip}(\ell(\theta_{i-1}; x_i), C)$
5: $\quad \mathbf{z}_i \leftarrow N(0, \frac{C^2\sigma^2}{B^2}\mathbb{I})$
6: $\quad \theta_i \leftarrow update(\theta_{i-i}, \mathbf{g}_i + \mathbf{z}_i)$. ▷ e.g. for SGD, $update(\theta, \mathbf{g}) = \theta - \eta\mathbf{g}$
7: **end for**

2 Background on DP-SGD and Privacy Accounting

DP-SGD [SCS13, BST14, ACG+16], formally stated as Algorithm 1, is a canonical private learning algorithm. In each round it samples a batch of examples, computes the gradient of the loss on these examples, clips the gradients to a bounded ℓ_2-norm, and adds Gaussian noise to privatize their sum. The privatized sum is then passed to a first-order method such as SGD or Adam.

DP-SGD is commonly analyzed via *privacy loss distributions*. Given two distributions P, Q, their *privacy loss random variable* is given by $\log(P(x)/Q(x)), x \sim P$. The privacy loss distribution is the distribution of this random variable. The *blow-up function* is closely related to the privacy loss distribution. Given two distributions P, Q, their blow-up function is the function $B_{P,Q}(x) := \max_{E:Q(E)\leq x} P(E)$. In other words, the blow-up function gives the maximum true positive rate of a binary classifier to identify samples from Q, with false positive rate at most x on P. [ZDW22] observed that the privacy loss distribution and blow-up function are equivalent representations, i.e. the privacy loss distribution of P, Q is easily computable from its blow-up function and vice-versa.

Under a given definition of adjacency $\simeq$ (i.e., a relation that defines what pairs of datasets are adjacent), a *dominating pair* P, Q for a mechanism M is a pair such that $B_{P,Q}(x) \geq B_{M(D),M(D')}(x)$ for all $x \in [0, 1], D \simeq D'$. A dominating pair P, Q is *tight* if there exists $D \simeq D'$ such that $B_{P,Q} = B_{M(D),M(D')}$. For DP-SGD when $D \simeq D'$ if D, D' differ in a group of examples of bounded size, tight dominating pairs are well-known [CGM+24] and their privacy loss distributions are easily computable via open-source software [DP 22].

3 Secret Protection

We now introduce our proposed definition of secret protection:

Definition 3.1. *Suppose as input, an algorithm A receives*

- *A dataset $D = \{x_1, \ldots, x_n\} \in \mathcal{D}^n$ in data universe $\mathcal{D}$,*
- *A list of secrets $S = \{y_1, \ldots, y_m\}$, and for each secret y_j a predicate $T_j : \mathcal{D}^2 \to \{0, 1\}$, where $T_j(x_i, x_i') = 1$ means that example x_i is equivalent to x_i' with only the secret y_j transformed.*
- *A function $E : D \to 2^S$, where $E(x_i)$ represents all secrets in S contained in x_i.*

We define inputs $I = (D = \{x_1, \ldots, x_n\}, S, E)$, $I' = (D' = \{x_1', \ldots, x_n'\}, S, E)$, as differing only in secret j, denoted $I \simeq_j I'$, if for all $i \in [n]$ either $y_j \notin E(x_i)$ or $T_j(x_i, x_i') = 1$. In other words, D and D' differ only in examples that contain the secret y_j, where they only differ by information pertaining to the secret, and otherwise the inputs (including S and E) don't differ.

Then, A satisfies $\{(p_j, r_j)\}$-secret protection if it satisfies the following: For any j, given a distribution $\mathcal{P}$ over $[k]$ such that $\mathcal{P}(i) \leq p_j$ for all i and a set of candidate inputs $\{I_1, I_2, \ldots, I_k\}$ such that for any i, i', $I_i \simeq_j I_{i'}$, for any adversary B, $Pr_{i\sim\mathcal{P},A}[B(A(I_i)) = i] \leq r_j$.

Suppose each all p_j were equal and all r_j were equal, T_j was always equal to 1, and we enforced the bound of r_j via a divergence (such as a Rényi or hockey-stick divergence) bound on $A(I), A(I')$ for all input pairs $I, I' : \exists j : I \simeq_j I'$.

If we treated each secret y_j as a user, and D_j as the set of examples owned by that user Definition 3.1 would recover a DP definition for user-level DP, where an example can be owned by multiple users, and which examples are owned by which users is public information. Hence, the key differences between Definition 3.1 and user-level DP are as follows:

- **Public membership:** We do not protect against knowing about the existence of secrets and which examples contain which secrets, just the contents of examples (i.e., the secrets themselves). This is encoded in the definition by the fact that the structure of the secrets given by $\{D_1, \ldots, D_m\}$ is shared between adjacent inputs. We note that this is analogous to the fixed-graph DP definition recently proposed in [GMM+25].
- **Non-uniform protection:** We do not require uniform "privacy parameters" for the secrets, as not all secrets are equally sensitive.
- **Privacy guarantee:** Our DP definition requires us to protect against reconstruction attacks, rather than bound the divergence of all pairs of outputs given by adjacent inputs. A divergence bound usually implies a reconstruction bound as we will demonstrate in Lemma 3.2, so this can be viewed as a loosening of the privacy guarantee.
- **Restricted adjacency:** The restriction of adjacency via T_j allows us to release public information within examples that also contain secrets without violating the privacy definition.

3.1 Why Is DP Overambitious?

We identify reasons why DP may be overly protective for the needs of secret protection, and state weaker goals for model training encoded in Definition 3.1 that allow us to improve utility.

First, as previously mentioned, differential privacy often targets a setting where the model trainer (e.g., a data curator working for a hospital) is a different entity than the data owners with privacy concerns (e.g., the hospital's patients). In turn, strong privacy protection is usually an ethical and/or legal requirement for using the data. In contrast, we are focused on a setting where *the model trainer and data owner are the same entity*, and hence they may be willing to forgo the strong protection of DP in favor of utility. For example, a trading company may be willing to tolerate lower privacy standards for training on its own codebase, if they believe the benefits of having a well-trained model outweigh the losses of forgoing very strong privacy guarantees.

Second, differential privacy often provides a uniform privacy protection to all involved users (with some exceptions we discuss later). However, *not all secrets are equally sensitive*. By blindly applying the standard definitions of DP (which provide a uniform privacy protection to all secrets) the company might sacrifice utility to protect information that the company is willing to risk a higher chance of leaking. For example, in the trading company's codebase, consider the following two Python files:

Example of `secret_investment.py`

```
"""Trains an investment model
for a secret purpose."""

import file_utils
import torch

def training_loop():
    ...
```

Example of `file_utils.py`

```
"""Utils for accessing file
system during model
training."""

def load_dataset(name: str):
    ...
```

`secret_investment.py` is a file which contains details of how a trading model is defined and trained for an investment the trading company wishes to keep secret, and hence is likely highly sensitive. `file_utils.py` contains tools for accessing the file system, which may be still be mildly sensitive but are less likely to be damaging if leaked. So we might tolerate weaker privacy protections for `file_utils.py` than for `secret_investment.py`. We note that non-uniform privacy protections have appeared in various forms in the literature, and we do not claim to be the first to propose this relaxation of privacy. For example [JYC15] proposed "personalized" DP, where each

user specifies their own privacy budget. However, an important qualitative difference between our secret protection definition and these definitions is the assumptions required for a non-uniform privacy protection to offer improvements in utility. When we have data owners who are a different entity than the model trainer and have no investment in the success of the model trainer, data owners who act in their own self-interest will always opt for the maximum privacy budget available to them. Furthermore, legal obligations such as GDPR compliance may obligate model trainers to achieve a target DP guarantee for all users [CD18], even if some users may be unconcerned with their privacy. Hence in these settings, relaxations such as personalized DP may be inapplicable, or may not offer any improvements in utility when they are applicable. In contrast, in our setting the model trainer and data owner are the same entity, and hence the legal requirement does not exist, and loosening privacy of some secrets may be in the data owner's own best interest. So, the gains in utility from allowing non-uniform protections are more likely to be realizable.

Finally, differential privacy usually protects against leaking any bit of information from an example, whereas we are only concerned with protecting the secrets, which are typically larger than one bit. This allows us to relax the privacy definition in several different ways. First, while DP protects against membership inference (i.e., knowing whether or not an example is in a dataset), the mere existence of a business secret may not be a privacy concern. For example, an employee might be able to see that there is a `secret_investment` library, but without knowing more details this is unproblematic. In this setting, our privacy definition does not need to protect against membership inference, and we can tailor the definition to protecting against stronger privacy violations like reconstructing the contents of `secret_investment.py`. Second, even for the problem of reconstruction, not all parts of a training example may be problematic to reconstruct. For example, if `secret_investment.py` is used as an example in training, the fact that `file_utils` and `torch` are imported together is fine to release even if the learning algorithm in the same file is a secret.

3.2 Defining S, E, T

In our secret protection definition we have deliberately left the definition of S, E, T_j, i.e. the secrets, which examples include which secrets, and what constitutes changing only a secret in an example, to be specified by the model trainer. We emphasize that while we have treated these as givens, defining these quantities is perhaps the most challenging part of using our secret protection definition in practice. The main challenge is that in contrast with e.g. user-level DP, where an example may come with metadata (e.g., the username of the account that uploaded the example) that easily lets us identify users and associations between examples and users, a definition of a secret and identifying an example as containing a secret are both likely more subjective.

We believe designing effective techniques for identifying secrets and examples containing secrets in various types of datasets is an important open problem. For our specific motivating example of a code dataset, we believe there may be hope in automating the process of identifying secrets and examples containing them, using structure built into the dataset. Dependencies between code libraries can hint at where in the codebase a secret may reappear. For example, if an algorithm which is a trade secret appears in a library that is commonly imported, it may be reasonable to tag other libraries importing this library as containing the secret. Similarly, code is often siloed by user groups that could hint at secret containment, and also has a commit history that associates code with certain contributors. By identifying user groups and code contributors working on sensitive projects, we could infer that files which they have modified contain a secret associated with the sensitive project.

Unfortunately, because datasets containing actual secrets cannot be made public or used for public research, we did not pursue an empirical investigation of techniques for defining these quantities as part of this work. We hope that future work can make progress on this problem.

3.3 Bounding the Posterior Probability

We now discuss different strategies for bounding the posterior reconstruction probability r_j given the prior probability p_j. Recall that the *blow-up function* of two distributions P, Q is $B_{P,Q}(x) := \max_{E:Q(E)\leq x} P(E)$. Theorem 2 of [HBM23] shows that if for all j and any inputs $I \simeq_j I'$ that $B_{A(I),A(I')}(p_j) \leq r_j$, this suffices for $\{(p_j, r_j)\}$-secret protection, and this is tight, i.e. one cannot achieve smaller r_j than $\max_{I\simeq I'} B_{A(I),A(I')}(p_j)$. Also recall that for DP-SGD, a "worst-case" privacy loss distribution and hence blow-up function are known, even for group privacy. Hence, in principle we can use this worst-case blow-up function to compute tight r_j values for DP-SGD.

However, a practical issue with working with privacy loss distributions is that the distribution is continuous, and hence we need to (i) discretize the distribution at some cost in accuracy, and (ii) deal with numerical errors due to working with floating-point precision. This can be handled in a pessimistic manner, i.e. any errors will only result in over-estimating parameters that measure privacy leakage such as the posterior reconstruction probabilities r_j, so the reported parameters are safe to use in practice. However, as we will discuss in further detail in Section 5.1, in some settings this makes the computation r_j via blow-up functions extremely unstable, to the point where the over-estimation is often too large to be usable. However, we found that the KL divergence, which is the mean of the privacy loss distribution, remained relatively stable. We can use the following lemma to bound posterior reconstruction probabilities via KL divergence and other divergences:

Lemma 3.2. *Let $\mathcal{P}$ be a prior distribution over $[k]$ such that $\mathcal{P}(i) \leq p$ for all $i \in [k]$, let $\mathcal{F}$ be an f-divergence function for strictly convex f, i.e. $\mathcal{F}(P,Q) := \mathbb{E}_{x\sim Q}\left[f(P(x)/Q(x))\right]$ for some convex f such that $f(1) = 0$. Let $P_1, \ldots, P_k$ be distributions such that for all i, j pairs $\mathcal{F}(P_i, P_j) \leq \mathcal{F}(Bern(q), Bern(p))$ where $q \geq p$. Then for any adversary B, $\mathbf{Pr}_{i\sim\mathcal{P}, y\sim P_i}[B(y) = i] \leq q$.*

Proof. This is a generalization of Fano's inequality, and we prove it similarly. Consider the sampling process $i \sim \mathcal{P}, y \sim P_i, i' \sim \mathcal{P}, y' \sim P_{i'}$. By (1) the post-processing property of f-divergences $\mathcal{F}(P,Q) \geq \mathcal{F}(g(P), g(Q))$ and (2) convexity of f-divergences: for $w_i \in (0,1)$, $\mathcal{F}(\sum_i w_i P_i, \sum_i w_i Q_i) < \sum_i w_i \mathcal{F}(P_i, Q_i)$, we have (abusing notation to let a random variable denote a distribution):

$$\begin{aligned}
\mathcal{F}(\mathbb{1}(B(y) = i), \mathbb{1}(B(y') = i)) &\overset{(1)}{\leq} \mathcal{F}((y,i),(y',i)) \\
&= \mathcal{F}(\sum_{i,i'} \mathcal{P}(i)\mathcal{P}(i')(P_i, i), \sum_{i,i'} \mathcal{P}(i)\mathcal{P}(i')(P_{i'}, i)) \\
&\overset{(2)}{\leq} \max_{i,i'} \mathcal{F}(P_i, P_{i'}). \\
&\leq \mathcal{F}(Bern(q), Bern(p)).
\end{aligned}$$

Now assume that $Pr[B(y) = i] > q$. Since f is strictly convex, this implies that for $q' > q \geq p \geq p'$, $\mathcal{F}(Bern(q'), Bern(p')) > \mathcal{F}(Bern(q), Bern(p))$. For any B, we have $Pr[B(y') = i] \leq p$, since B has no knowledge of i. So if $Pr[B(y) = i] > q$, we have $Pr[B(y) = i] > q \geq p \geq Pr[B(y') = i]$ and hence $\mathcal{F}(\mathbb{1}(B(y) = i), \mathbb{1}(B(y') = i)) > \mathcal{F}(Bern(q), Bern(p))$, which is a contradiction. □

The KL divergence is the f-divergence given by $f(y) = y \log y$, which is strictly convex. So another strategy for bounding the posterior reconstruction probability of y_j, r_j, given a worst-case privacy loss distribution for $A(I), A(I'), I \simeq_j I'$ is to compute the mean of the privacy loss distribution μ, then compute the $r_j > p_j$ such that $KL(Bern(r_j), Bern(p_j)) = \mu$. If μ is computed from an algorithm with a noise level, we can use binary search to invert this procedure, finding the minimum noise level that achieves KL divergence bound $KL(Bern(r_j), Bern(p_j))$ for our target r_j. Unlike using the blow-up function to compute r_j, this is not tight (and cannot be, as the blow-up function $T : [0,1] \to [0,1]$ contains strictly more information than the single parameter μ), but in our experiments proved to be more numerically stable.

4 Algorithms for Secret Protection

Given Definition 3.1, we can now consider algorithms for model training that achieve secret protection. As discussed in Section 3.3, it is straightforward to derive posterior reconstruction probability bounds for DP-SGD. However, if we train on the entire dataset with DP-SGD, some secrets may appear many more times than others (relative to their desired level of secret protection), and for these secrets we may unnecessarily increase the noise multiplier in DP-SGD, worsening its utility. Hence to achieve $\{(p_j, r_j)\}$-secret protection at minimal cost in utility, we want to preprocess the dataset such that each secret y_j appears in the dataset a number of times roughly equal to an appropriate function

of its corresponding (p_j, r_j). At the same time, we want to make sure that when bounding the number of appearances of each secret, we do not discard too much of the data.

Our main algorithmic idea is to formulate a linear program for the problem of selecting a subset of the data such that we bound the number of appearances of each secret while keeping as many examples as possible. The linear program solution gives weights for each example, and then we can use Poisson sampling to form batches where each example's sampling probability is proportional to its assigned weight. That is, we will first compute a weight w_i for each example i. Then, if we sample example i with probability $\frac{Bw_i}{\sum_{i'} w_{i'}}$ (which is in $[0,1]$ assuming $\max_i w_i \leq \sum_{i'} w_{i'}/B$), our expected batch size is B. Once we've decided on the sampling probabilities, we can set the noise multiplier to the minimum value such that each secret's target privacy guarantee is achieved.

We formulate the following linear program for assigning weights to examples:

$$\max \sum_{i \in D} w_i, \qquad s.t. \forall j \in S : \sum_{i:(i,j)\in E} w_i \leq c\mu_j, \qquad \forall i \in D : w_i \in [0,1]. \tag{1}$$

The objective is to maximize the sum of the weights, i.e. include as many examples as possible. We constrain w_i to be in $[0,1]$, which ensures that $\frac{Bw_i}{\sum_{i'} w_{i'}}$ is a probability as long as we find a solution to the LP where $\sum_{i'} w_{i'} > B$. For each secret j, we write a constraint saying the sum of the weights assigned to it is at most a target μ_j, times a constant c which we set before solving the program.

We set μ_j to be the KL divergence that guarantees posterior reconstruction probability at most r_j by Lemma 3.2. This is based on the observation by past work that DP-SGD's group privacy parameter is roughly linear in the group size [CGM+24], which $\sum_{i:(i,j)\in E} w_i$ is a fractional version of. We note that ideally, we would include the noise multiplier σ as a variable in the program and encode the downstream accounting that computes μ_j as a constraint. However, this accounting is computationally expensive and potentially would give a non-convex constraint.

The choice of the constant c serves to trade off between preventing discarding too many examples and avoiding overrepresented secrets. When c is sufficiently large, the constraints $\sum_{i:(i,j)\in E} w_i \leq c\mu_j$ can never be saturated, i.e. only the constraint $w_i \in [0,1]$ matters and hence the optimal solution is just to set all $w_i = 1$. This is equivalent to running DP-SGD on the full dataset, which includes as many examples as possible but will require a high noise multiplier due to some secrets being overrepresented. On the other hand, when c is sufficiently small, the constraint $w_i \leq 1$ can never be saturated because the constraints $\sum_{i:(i,j)\in E} w_i \leq c\mu_j$ are stricter. By well-known properties of basic feasible solutions to linear programs, $n = |D|$ constraints must be tight for the optimal solution, and there are only $m = |S|$ constraints of the form $\sum_{i:(i,j)\in E} w_i \leq c\mu_j$, which means $n - m$ constraints of the form $w_i \geq 0$ will be tight. In other words, if c is too small, the solution will assign $w_i = 0$ to all but m examples, i.e. discard a large fraction of examples. As we interpolate between large and small values of c, we will trade off between these two extreme solutions. In our experiments, we include c as a hyperparameter to be tuned. Note that under our adjacency definition, the linear program (and its solution) can be treated as public information since we assume G is public. Hence, sweeping c has the same privacy ramifications as sweeping other parameters like learning rate, whose privacy cost is commonly ignored both in research settings and in practice.

Accounting for secret protection: A slight variation of Theorem 1 of [CGM+24] shows that if examples in $E^{-1}(y_j) := \{x \in D : y_j \in E(x)\}$ have sampling probabilities $\rho_1, \ldots, \rho_k$ and we do T rounds of DP-SGD with Poisson sampling, then $N(\sum_{i=1}^{k} Bern(\rho_i), \sigma^2)^T$, $N(0,\sigma^2)^T$ is a dominating pair for DP-SGD with noise multiplier σ and pairs of inputs differing only in $E^{-1}(y_j)$. The open source `dp_accounting` library supports computing the privacy loss distribution for this pair of distributions, and hence the KL divergence between the dominating pair which is an upper bound on the KL divergence between the outputs of DP-SGD. Hence using Lemma 3.2, given the sampling probabilities ρ_j from the linear program, we can easily compute an indistinguishability guarantee for each secret. Then given the sampling probabilities, we can perform binary search on σ to find (approximately) the minimum σ such that all target indistinguishability guarantees are met, and run DP-SGD with these sampling probabilities and noise multiplier. We summarize our end-to-end training procedure, including the noise calibration, in Algorithm 2.

Algorithm 2 Secret protection training pipeline

Inputs: Dataset $D = \{x_1, \ldots, x_n\}$, loss ℓ, optimizer update function $update$, secrets S, secret protection parameters (p_j, r_j), number of rounds T, target batch size B, clip norm C, noise multiplier σ, initial model θ_0.

For each secret j, $\mu_j \leftarrow KL(Bern(r_j), Bern(p_j))$.
$\{w_i\} \leftarrow$ solution to (1) using the chosen μ_j values.
For each example i, $\rho_i = \frac{Bw_i}{\sum_{i'} w_{i'}}$
For each secret j, $\sigma_j = \min\{\sigma : KL(N(\sum_{x_i \in D_j} Bern(\rho_i), \sigma^2)^T, N(0, \sigma^2)^T) \leq \mu_j\}$
$\sigma = \max_j \sigma_j$
$\mathsf{clip}\,(\mathbf{v}, C) := \mathbf{v} \cdot \min\{1, C/\left\|\mathbf{v}\right\|_2\}$
for $t \in [T]$ **do**
    $\mathcal{B}_i \leftarrow$ batch including x_i independently w.p. ρ_i.
    $\mathbf{g}_i \leftarrow \frac{1}{B} \sum_{x_i \in \mathcal{B}_i} \nabla \mathsf{clip}\,(\ell(\theta_{i-1}; x_i), C)$
    $\mathbf{z}_i \leftarrow N(0, \frac{C^2\sigma^2}{B^2}\mathbb{I})$
    $\theta_i \leftarrow update(\theta_{i-i}, \mathbf{g}_i + \mathbf{z}_i)$. ▷ e.g. for SGD, $update(\theta, \mathbf{g}) = \theta - \eta\mathbf{g}$
end for

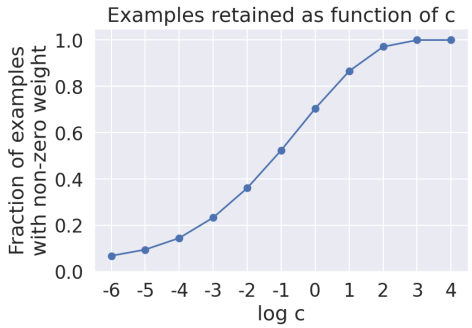
5 Experiments

5.1 Secret Protection

We now empirically investigate the effectiveness of our algorithm for training tailored to Definition 3.1. Again, since we wish to use a public dataset (without actual business secrets), we use the arXiv dataset [CBOA19] as a benchmark, as it still represents a collection of knowledge on specialized topics. The arXiv dataset consists of 1.9 million papers uploaded to arXiv.org, including their abstract. We randomly split the dataset into a training set of 1.7 million examples, a test set of 0.1 million examples, and a validation set of 0.1 million examples. We fine-tune a pretrained BERT-Tiny model on the abstracts of examples in the arXiv dataset for the masked language modeling task as defined in [DCLT19]. We sort all words appearing in any abstract by the number of abstracts they appear in, take the 50001st to 150000th words in decreasing order, and treat these as secrets. We treat each abstract as containing a secret if it contains the corresponding word. This set of secrets each appears roughly 50 to 100 times each. We use a fixed $p_j = 10^{-10}$ and assign each secret a uniformly random $r_j \in [2 \cdot 10^{-4}, 10^{-3}]$. We discard any examples not containing secrets for simplicity. We fix a batch size of 2048 and 2000 iterations of training, and sweep the learning rate. We fix the clip norm to be 1.0 as we found this to be reasonably effective in most settings. To simulate training with more resources on a larger dataset, in our privacy accounting we assume our batch size and dataset size are 10 times larger, which is equivalent to reducing the noise multiplier by a factor of 10[1].

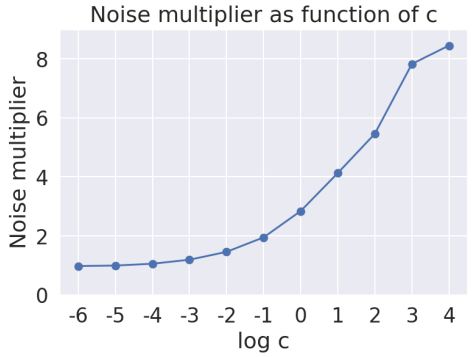
We compare our linear program-based preprocessing with the baseline of no preprocessing. For simplicity in Definition 3.1 we let the predicates T_j be always 1, i.e. two adjacent datasets can differ in the examples containing the sensitive secret s_j arbitrarily. In Figure 1a we first show for each $c = 2^k$ where $k \in \{-6, -5, \ldots, 4\}$ what fraction of examples are retained. For setting the constant in the linear program to be $c = 2^4$, our LP solution assigns all examples a weight of 1, i.e. does no preprocessing of the dataset. So we will compare these choices of c, and demonstrate improvements over $c = 2^4$. In Figure 1b we compare the test losses achieved by training without noise on the resulting datasets, where we see only a small ≈ 0.04 increase in test loss due to the pre-processing step even though it discards many examples. In contrast, in Figure 1c, we compare the noise multipliers we need to achieve the desired bounds on posterior reconstruction probabilities via KL divergence bounds. We see that our preprocessing can lead to a significant multiplicative reduction of ≈ 8 in the noise multiplier compared to no preprocessing. In Figure 1d we plot the test losses and see that the noise multiplier reduction from preprocessing strongly outweighs the small benefits from having the whole dataset, leading to a substantial decrease in the loss.

[1] Since we decrease the scale of the noise, this should only make any comparisons more unfavorable for our LP-based approach.



(a) The fraction of examples in the dataset which are assigned non-zero weight.

(b) The test loss achieved by the different LP solutions, without any noise.



(c) The noise multiplier needed for the desired reconstruction probability bounds.

(d) The test loss achieved by the different LP solutions.

Figure 1: Comparison of LP solutions for different values of c. In Fig. 1d, the 95% confidence intervals have radius at most .001.

Infeasibility of the Blow-Up Function: In our experiments we attempted to use the privacy loss distribution to compute the blow-up function, which would allow us to use the tight bounds on reconstruction probability of [HBM23]. Using the `dp_accounting` tools for computing the privacy loss distribution, we get a discrete privacy loss distribution with support $v_1, \ldots, v_k$ and probabilities $p_1, \ldots, p_k$. The corresponding blow-up function evaluated at $\sum_{i=j}^{k} p_i$ has a value of $\sum_{i=j}^{k} e^{v_i} p_i$, and linearly interpolates between these points. In particular, by definition of the blow-up function it should be the case $\sum_{i=1}^{k} e^{v_i} p_i = 1$. In our experiments, when computing the privacy loss distribution for two datasets differing in a single example at the noise multipliers in Figure 1c, we did find the equality $\sum_{i=1}^{k} e^{v_i} p_i = 1$ held. However when we did the same calculation for two datasets differing in a secret (≈ 100 examples), we found instances where $\sum_{i=1}^{k} e^{v_i} p_i \geq 2 \cdot 10^4$. In other words, the numerical instability and pessimism in PLD accounting (specifically for *Mixture of Gaussians mechanisms*, i.e. Gaussian mechanisms whose sensitivity's support is larger than $\{0, 1\}$) heavily inflates the blowup function. For this reason we opted to use the KL divergence-based reconstruction bounds of Lemma 3.2 instead.

5.2 Reconstructing Pre-Training data

While our primary focus is on protecting pre-defined secrets, the underlying principle—that moderate indistinguishability guarantees, potentially weaker than standard DP, can suffice to limit reconstruction attacks can also apply to the broader problem of preventing training data extraction from large language models. Large language models (LLMs) are known to memorize portions of their training data, which can then be extracted [NCH+23, CTW+21, CIJ+]. This section focuses on reconstruction attacks (also known as extraction attacks) against large language models to validate

that moderate indistinguishability guarantees suffice to protect against reconstruction. These attacks aim to extract training data from the model. We adopt a similar experimental setup and definition of memorization.

Definition 5.1 ((k-memorization) [CIJ+]). *A string s is extractable with k tokens of context from a model f if there exists a (length-k) string p, such that the concatenation $[p||s]$ is contained in the training data for f, and $f(p) = s$ using greedy decoding.*

It's important to note that this definition might not encompass all possible forms of memorization [ITN+22]. However, it serves as a valuable proxy for measuring the extent of data extraction and provides a benchmark for comparison. This metric is widely used in various contexts, including copyright infringement cases, to assess the unauthorized reproduction of training data.

Pretraining Setup: For our experiments, we pretrained a series of causal language models using the Gemma-2 architecture [TRP+24] and the fineweb-edu dataset [PKa+24] tokenized using the Gemma-2 tokenizer. We explored various model sizes within the Gemma-2 family.

Extraction Experiments: To evaluate the reconstruction attack on our trained models, we fix a subset of the dataset. We prompted each model with the first n tokens of a sequence from the subset and observed whether it could generate the subsequent m tokens. Our evaluation focused exclusively on greedy sampling, where the model selects the most probable token at each step.

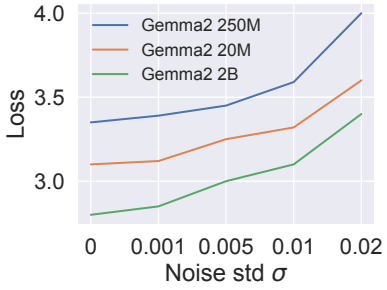


Figure 2: Effect of noise on the validation loss

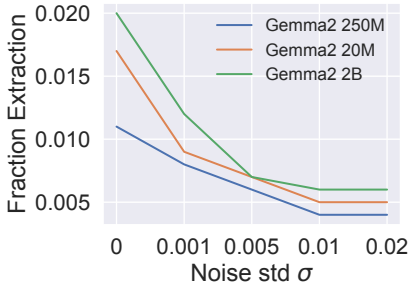


Figure 3: Fraction memorized

Results: Figure 2 demonstrates the impact noise addition has on training. With small noise multiplier, the effect on model utility is minimal. Figure 3 highlights the effectiveness of small noise multipliers in mitigating memorization. A small amount of noise leads to a substantial reduction in memorized instances. Interestingly, we see even with large noise multiplier (≥ 0.01) there is some remaining memorization. We manually looked at the memorized examples for higher noise multipliers and observed that remaining instances are primarily attributed to generalization. For example, prompting the model with "12345" results in the continuation "6789", even with increased noise levels. This suggests the model learns underlying patterns rather than simply memorizing specific sequences.

References

[ACG+16] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS'16), pages 308–318, 2016.

[BCH22] Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. pages 1138–1156, 05 2022.

[BDF+19] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning, 2019.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In Proc. of the 2014 IEEE 55th Annual Symp. on Foundations of Computer Science (FOCS), pages 464–473, 2014.

[CBOA19] Colin B. Clement, Matthew Bierbaum, Kevin P. O'Keeffe, and Alexander A. Alemi. On the use of arxiv as a dataset. CoRR, abs/1905.00075, 2019.

[CD18] Rachel Cummings and Deven R. Desai. The role of differential privacy in gdpr compliance. 2018.

[CGH+24] Lynn Chua, Badih Ghazi, Yangsibo Huang, Pritish Kamath, Ravi Kumar, Daogao Liu, Pasin Manurangsi, Amer Sinha, and Chiyuan Zhang. Mind the privacy unit! user-level differential privacy for language model fine-tuning. In First Conference on Language Modeling, 2024.

[CGM+24] Zachary Charles, Arun Ganesh, Ryan McKenna, H. Brendan McMahan, Nicole Mitchell, Krishna Pillutla, and Keith Rush. Fine-tuning large language models with user-level differential privacy, 2024.

[CIJ+] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In The Eleventh International Conference on Learning Representations.

[CTW+21] 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), pages 2633–2650. USENIX Association, August 2021.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[DFY+22] Wei Dong, Juanru Fang, Ke Yi, Yuchao Tao, and Ashwin Machanavajjhala. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In Proceedings of the 2022 International Conference on Management of Data, SIGMOD '22, page 759–772, New York, NY, USA, 2022. Association for Computing Machinery.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Proc. of the Third Conf. on Theory of Cryptography (TCC), pages 265–284, 2006.

[DP 22] DP Team. Google's differential privacy libraries., 2022. `https://github.com/google/differential-privacy`.

[GMM+25] Arun Ganesh, Ryan McKenna, Brendan McMahan, Adam Smith, and Fan Wu. It's my data too: Private ml for datasets with multi-user training examples, 2025.

[HBM23] Jamie Hayes, Borja Balle, and Saeed Mahloujifar. Bounding training data reconstruction in DP-SGD. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

[ITN+22] Daphne Ippolito, Florian Tramèr, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher A Choquette-Choo, and Nicholas Carlini. Preventing verbatim memorization in language models gives a false sense of privacy. arXiv preprint arXiv:2210.17546, 2022.

[JYC15] Zach Jorgensen, Ting Yu, and Graham Cormode. Conservative or liberal? personalized differential privacy. volume 2015, 04 2015.

[NCH+23] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. arXiv preprint arXiv:2311.17035, 2023.

[PKa+24] Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024.

[SCS13] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In 2013 IEEE Global Conference on Signal and Information Processing, pages 245–248. IEEE, 2013.

[TRP+24] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024.

[YKK+23] Da Yu, Gautam Kamath, Janardhan Kulkarni, Tie-Yan Liu, Jian Yin, and Huishuai Zhang. Individual privacy accounting for differentially private stochastic gradient descent. Transactions on Machine Learning Research, 2023.

[ZDW22] Yuqing Zhu, Jinshuo Dong, and Yu-Xiang Wang. Optimal accounting of differential privacy via characteristic function. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, volume 151 of Proceedings of Machine Learning Research, pages 4782–4817. PMLR, 28–30 Mar 2022.