One-shot Empirical Privacy Estimation for Federated Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Privacy estimation techniques for differentially private (DP) algorithms are useful for comparing against analytical bounds, or to empirically measure privacy loss in settings where known analytical bounds are not tight. However, existing privacy auditing techniques usually make strong assumptions on the adversary (e.g., knowledge of intermediate model iterates or the training data distribution), are tailored to specific tasks, model architectures, or DP algorithm, and/or require retraining the model many times (typically on the order of thousands). These shortcomings make deploying such techniques at scale difficult in practice, especially in federated settings where model training can take days or weeks. In this work, we present a novel "one-shot" approach that can systematically address these challenges, allowing efficient auditing or estimation of the privacy loss of a model during the same, single training run used to fit model parameters, and without requiring any *a priori* knowledge about the model architecture, task, or DP training algorithm. We show that our method provides provably correct estimates for the privacy loss under the Gaussian mechanism, and we demonstrate its performance on well-established FL benchmark datasets under several adversarial threat models.

1 Introduction

Federated learning (FL) [McMahan et al., 2017, Kairouz et al., 2021b] is a paradigm for training machine learning models on decentralized data. At each round, selected clients contribute model updates to be aggregated by a server, without ever communicating their raw data. FL incorporates *data minimization* principles to reduce the risk of compromising anyone's data: each user's data never leaves their device, the update that is transmitted contains only information necessary to update the model, the update is encrypted in transit, and the update exists only ephemerally before being combined with other clients' updates and then incorporated into the model [Bonawitz et al., 2022]. Technologies such as secure aggregation [Bonawitz et al., 2017, Bell et al., 2020] can be applied to ensure that even the central server cannot inspect individual updates, but only their aggregate.

However, these data minimization approaches cannot rule out the possibility that an attacker might learn some private information from the training data by directly interrogating the final model [Carlini et al., 2021, Balle et al., 2022, Haim et al., 2022]. To protect against this, *data anonymization* for the model is required. FL can be augmented to satisfy user-level differential privacy [Dwork and Roth, 2014, Abadi et al., 2016, McMahan et al., 2018], the gold-standard for data anonymization. DP can guarantee each user that a powerful attacker – one who knows all other users' data, all details about the algorithm (other than the values of the noise added for DP), and every intermediate model update – still cannot confidently infer the presence of that user in the population, or anything about their data. This guarantee is typically quantified by the parameter ε, with lower values corresponding to higher privacy (less confidence for the attacker).

DP is often complemented by *empirical privacy estimation* techniques, such as membership inference attacks [Shokri et al., 2017, Yeom et al., 2018, Carlini et al., 2022], which measure the success of an adversary at distinguishing whether a particular record was part of training or not.[1] Such methods have been used to audit the implementations of DP mechanisms or claims about models trained with DP [Jagielski et al., 2020, Nasr et al., 2021, Zanella-Béguelin et al., 2022, Lu et al., 2022]. They are also useful for estimating the privacy loss in cases where a tight analytical upper bound on ε is unknown, for example when clients are constrained to participate in at most some number of rounds, or when the adversary does not see the full trace of model iterates. However, existing privacy auditing techniques suffer from several major shortcomings. First, they require retraining the model many times (typically in the thousands) to provide reliable estimates of DP's ε [Jagielski et al., 2020, Nasr et al., 2021]. Second, they often rely on knowledge of the model architecture and/or the underlying dataset (or at least a similar, proxy dataset) for mounting the attack. For example, a common approach is to craft a "canary" training example on which the membership is being tested, which typically requires an adversary to have access to the underlying dataset and knowledge of the domain and model architecture. Finally, such techniques typically grant the adversary unrealistic power, for example (and in particular) the ability to inspect all model iterates during training [Maddock et al., 2022], something which may or may not be reasonable depending on the system release model.

Such assumptions are particularly difficult to satisfy in FL due to the following considerations:

- **Minimal access to the dataset, or even to proxy data.** A primary motivating feature of FL is is that it can make use of on-device data without (any) centralized data collection. In many tasks, on-device data is more representative of real-world user behavior than any available proxy data.
- **Infeasibility of training many times, or even more than one time.** FL training can take days or weeks, and expends resources on client devices. To minimize auditing time and client resource usage, an ideal auditing technique should produce an estimate of privacy during the same, single training run used to optimize model parameters, and without significant overhead from crafting examples or computing additional "fake" training rounds.
- **Lack of task, domain, and model architecture knowledge.** A scalable production FL platform is expected to cater to the needs of many diverse ML applications, from speech to image to language modeling tasks. Therefore, using techniques that require specific knowledge of the task and/or model architecture makes it hard to deploy those techniques at scale in production settings.

In this paper, we design an auditing technique tailored for FL usage with those considerations in mind. We empirically estimate ε efficiently under user-level DP federated learning by measuring the training algorithm's tendency to memorize arbitrary clients' updates. Our main insight is to insert multiple canary clients in the federated learning protocol with independent random model updates, and design a test statistic based on cosine angles of each canary update with the final model to test participation of a certain user in the protocol. The intuition behind the approach comes from the elementary result that in a high-dimensional space, isotropically sampled vectors are nearly orthogonal with high probability. So we can think of each canary as estimating the algorithm's tendency to memorize along a dimension of variance that is independent of the true model updates, and of the other canaries.

Our method has several favorable properties. It can be applied during the same, single training run which is used to train the federated model parameters, and therefore does not incur additional performance overhead. Although it does inject some extra noise into the training process, the effect on model quality is negligible, provided model dimensionality and number of clients are reasonably sized. We show that in the tractable case of a single application of the Gaussian mechanism, our method provably recovers the true, analytical ε in the limit of high dimensionality. We evaluate privacy loss for several adversarial models of interest, for which existing analytical bounds are not tight. In the case when all intermediate updates are observed and the noise is low, our method produces high values of ε, indicating that an attacker could successfully mount a membership inference attack. However, in the common and important case that only the final trained model is released, our ε estimate is far lower, suggesting that adding a modest amount of noise is sufficient to prevent leakage, as has been observed by practitioners. Our method can also be used to explore how leakage changes as aspects of the training protocol change, for which no tight theoretical analysis is known, for example if we limit client participation. The method we propose is model and dataset agnostic, so it can be easily applied without change to any federated learning task.

[1] Some prior work only applies to example-level DP, in which *records* correspond to examples, as opposed to user-level, in which *records* are users. We will describe our approach in terms of user-level DP, but it can be trivially modified to provide example-level DP.

2 Background and related work

Differential privacy. Differential privacy (DP) [Dwork et al., 2006, Dwork and Roth, 2014] is a rigorous notion of privacy that an algorithm can satisfy. DP algorithms for training ML models include DP-SGD [Abadi et al., 2016], DP-FTRL [Kairouz et al., 2021a], and DP matrix factorization [Denissov et al., 2022, Choquette-Choo et al., 2022]. Informally, DP guarantees that a powerful attacker observing the output of the algorithm A trained on one of two *adjacent* datasets (differing by addition or removal of one record), D or D', cannot confidently distinguish the two cases, which is quantified by the privacy parameter ϵ.

Definition 2.1. User-level differential privacy. The training algorithm $A : \mathcal{D} \to \mathcal{R}$ is user-level (ϵ, δ) differentially private if for all pairs of datasets D and D' from $\mathcal{D}$ that differ only by addition or removal of the data of one user and all output regions $R \subseteq \mathcal{R}$:

$$\Pr[A(D) \in R] \leq e^{\epsilon} \Pr[A(D') \in R] + \delta.$$

DP can be interpreted as a hypothesis test with the null hypothesis that A was trained on D and the alternative hypothesis that A was trained on D'. False positives (type-I errors) occur when the null hypothesis is true, but is rejected, while false negatives (type-II errors) occur when the alternative hypothesis is true, but is rejected. Kairouz et al. [2015] characterized (ϵ, δ)-DP in terms of the false positive rate (FPR) and false negative rate (FNR) achievable by an acceptance region. This characterization enables estimating the privacy parameter as:

$$\hat{\epsilon} = \max\{\log \frac{1 - \delta - \text{FPR}}{\text{FNR}}, \log \frac{1 - \delta - \text{FNR}}{\text{FPR}}\}. \tag{1}$$

We review and compare with related work in Appendix F.

3 One-shot privacy estimation for the Gaussian mechanism

As a warm-up, we start by considering the problem of estimating the privacy of the Gaussian mechanism, the fundamental building block of DP-SGD and DP-FedAvg. To be precise, given $D = (x_1, \cdots, x_n)$, with $\|x_i\| \leq 1$ for all $i \in [n]$, the output of the Gaussian vector sum query is $A(D) = \bar{x} + \sigma Z$, where $\bar{x} = \sum_i x_i$ and $Z \sim \mathcal{N}(0, I)$. Without loss of generality, we can consider a neighboring dataset D' with an additional vector x with $\|x\| \leq 1$. Thus, $A(D) \sim \mathcal{N}(\bar{x}, \sigma^2 I)$ and $A(D') \sim \mathcal{N}(\bar{x} + x, \sigma^2 I)$. For the purpose of computing the DP guarantees, this mechanism is equivalent to analyzing $A(D) \sim \mathcal{N}(0, \sigma^2)$ and $A(D') \sim \mathcal{N}(1, \sigma^2)$ due to spherical symmetry.

The naive approach for estimating the ε of an implementation of the Gaussian mechanism would run it many times (say 1000 times), with half of the runs on D and the other half on D'. Then the outputs of these runs are shuffled and given to an "attacker" who attempts to determine for each output whether it was computed from D or D'. Finally, the performance of the attacker is quantified and Eq. (1) is used to obtain an estimate of the mechanism's ε at a target δ.

We now present an approach for estimating ε by running the mechanism *only once*. The basic idea behind our approach is to augment the original dataset with k canary vectors $c_1, \ldots, c_k$, sampled i.i.d uniformly at random from the unit sphere, obtaining $D = (x_1, \cdots, x_n, c_1, \cdots, c_k)$. We consider k neighboring datasets, each excluding one of the canaries, i.e., $D'_i = D \setminus \{c_i\}$ for $i \in [k]$. We run the Gaussian mechanism once on D and use its output to compute k test statistics $\{g_i\}_{i \in [k]}$, the cosine of the angles between the output and each one of the k canary vectors. We use these k cosines to estimate the distribution of test statistic on D by computing the sample mean $\hat{\mu} = \frac{1}{k} \sum_{j=1}^{k} g_j$ and sample variance $\hat{\sigma}^2 = \frac{1}{k} \sum_{j=1}^{k} (g_j - \hat{\mu})^2$ and fitting a Gaussian $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$. To estimate the distribution of the test statistic on D', we need to run the mechanism on each D'_i and compute the cosine of the angle between the output vector and c_i. This is where our choice of (i) independent isotropically distributed canaries and (ii) cosine angles as our test statistic are particularly useful. The distribution of the cosine of the angle between an isotropically distributed unobserved canary and the mechanism output (or any independent vector) can be described in a closed form; there is no need to approximate this distribution with samples. We will show in Theorems 3.1 and 3.2 that this distribution can be well approximated by $\mathcal{N}(0, 1/d)$. Now that we have the distribution of the test statistic on D and D', we estimate the ε of the mechanism using the method given in Appendix B which allows us to compute the ε when the null and alternate hypotheses are two arbitrary Gaussians. Our approach is summarized in Algorithm 1.

Algorithm 1 One-shot privacy estimation for Gaussian mechanism.

Input: Vectors $x_1, \cdots, x_n$ with $\|x_i\| \leq 1$, DP noise variance σ^2, and target δ
$\rho \leftarrow \sum_{i \in [n]} x_i$
for $j \in [k]$ **do**
  Draw random $c_j \in \mathbb{S}^{d-1}$ unit sphere
  $\rho \leftarrow \rho + c_j$
Release $\rho \leftarrow \rho + \mathcal{N}(0, \sigma^2 I)$
for $j \in [k]$ **do**
  $g_j \leftarrow \langle c_j, \rho \rangle / ||\rho||$
$\hat{\mu}, \hat{\sigma} \leftarrow \mathbf{mean}(\{g_j\}), \mathbf{std}(\{g_j\})$
$\hat{\varepsilon} \leftarrow \varepsilon(\mathcal{N}(0, 1/d) \,||\, \mathcal{N}(\hat{\mu}, \hat{\sigma}^2); \delta)$

We argue that the approach given in Algorithm 1 gives an estimate of ε that approaches the exact value when d is high. To do so, we will prove (Theorems 3.1 and 3.2) that distribution of the test statistic on D' is indeed well approximated by $\mathcal{N}(0, 1/d)$, and (Theorem 3.3) that $\sqrt{d}\hat{\mu} \xrightarrow{p} 1/\sigma$ and $d\hat{\sigma}^2 \xrightarrow{p} 1$ as $d \to \infty$, i.e., the distribution of test statistic on D is $\mathcal{N}(\frac{1}{\sigma\sqrt{d}}, \frac{1}{d})$, asymptotically. Since these two distributions are just a scaling of $A(D) \sim \mathcal{N}(0, \sigma^2)$ and $A(D') \sim \mathcal{N}(1, \sigma^2)$ by a factor of $\frac{1}{\sigma\sqrt{d}}$, the ε is the same, which proves our claim.[2] (All proofs in Appendix A.)

Theorem 3.1. *Let S be sampled uniformly from the unit sphere in $\mathbb{R}^d$, and let $\tau = \langle S, v \rangle / \|v\| \in [-1, 1]$ be the cosine similarity between S and some arbitrary independent nonzero vector v. Then, the probability density function of τ is*

$$f_d(\tau) = \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}} (1 - \tau^2)^{\frac{d-3}{2}}.$$

Theorem 3.2. *In the setting of Theorem 3.1, we have that $\tau\sqrt{d}$ converges in distribution to $\mathcal{N}(0, 1)$ as $d \to \infty$, i.e., $\lim_{d \to \infty} \mathbb{P}(\tau \leq \lambda/\sqrt{d}) = \mathbb{P}_{Z \sim \mathcal{N}(0,1)}(Z \leq \lambda)$.*

Theorem 3.3. *For $d \in \mathbb{N}$, let $k = o(d)$, but $k = \omega(1)$. For $i = 1 \ldots k$, let c_i sampled i.i.d. from the unit sphere in d dimensions. Let $Z \sim \mathcal{N}(0; I_d)$. Let $\sigma > 0$, and define the mechanism result $\rho = \sum_{j=1}^{k} c_j + \sigma Z$, and the cosine values $g_j = \frac{\langle c_j, \rho \rangle}{\|\rho\|}$. Write the empirical mean of the cosines $\hat{\mu} = \frac{1}{k} \sum_{j=1}^{k} g_j$, and the empirical variance $\hat{\sigma}^2 = \frac{1}{k} \sum_{j=1}^{k} \big(g_j - \hat{\mu}\big)^2$. Then as $d \to \infty$, $\sqrt{d}\hat{\mu} \xrightarrow{p} 1/\sigma$ and $d\hat{\sigma}^2 \xrightarrow{p} 1$.*

We note that running the algorithm with moderate values of d and k already yields a close approximation. We simulated the case where $d = 10^6$, $k = 10^3$, and the results are shown in Table 1. The estimated ε_{est} is very close to the true value of ε, with small standard deviation, which demonstrates that the random canary method provides tight estimates for the Gaussian mechanism.

σ	analytical ε	ε_{est}
4.22	1.0	0.972 ± 0.148
1.54	3.0	3.04 ± 0.137
0.541	10.0	9.98 ± 0.190

Table 1: One-shot auditing of the Gaussian mechanism with $d = 10^6$, $k = 10^3$, and $\delta = 10^{-6}$. For each value of ε, we set σ using the optimal calibration of Balle and Wang [2018], and then use the random canary method to output the estimate ε_{est}. Shown is the mean and std ε_{est} over 50 simulations.

4 One-shot privacy estimation for FL with random canaries

We now extend this idea to DP Federated Averaging to estimate the privacy of releasing the final model parameters in one shot, during model training. We propose adding k canary clients to the

[2] Our Theorem 3.3 as written does not include the data vectors x_i in the sum ρ. It can be extended to do so if we also assume that $n = o(d)$.

training population who participate exactly as real users do. Each canary client generates a random model update sampled from the unit sphere, which it returns at every round in which it participates, scaled to have norm equal to the clipping norm for the round. After training, we collect the set of canary/final-model cosines, fit them to a Gaussian, and compare them to the null hypothesis distribution $\mathcal{N}(0, 1/d)$ just as we did for the basic Gaussian mechanism. The procedure is described in Algorithm 2. (Algorithms 2 and 3 can be found in Appendix C.)

FL is an optimization procedure in which each model iterate is a linear combination of all updates received thus far, plus Gaussian noise. Our threat model allows the attacker to control the updates of a client, and the ability to inspect the final model. We argue that it is a powerful (perhaps optimal, under some assumptions) strategy to return a very large update that is essentially orthogonal to all other updates, and then measure the dot product (or cosine) to the final model. Here we use the fact that randomly sampled canary updates are nearly orthogonal to all the true client updates and also to each other. Unlike many works that only produce correct estimates when clients are sampled uniformly and independently at each round, our method makes no assumptions on the pattern of client participation. Clients may be sampled uniformly at each round, shuffled and processed in batches, or even participate according to the difficult-to-characterize *de facto* pattern of participation of real users in a production system. Our only assumption is that canaries can be inserted according to the same distribution that real clients are. In production settings, a simple and effective strategy would be to designate a small fraction of real clients to have their model updates replaced with the canary update whenever they participate. If the participation pattern is such that memorization is easier, for whatever reason, the distribution of canary/final-model cosines will have a higher mean, leading to higher ε estimates.

We stress that our empirical ε_{est} estimate should not be construed as a formal bound on the worst-case privacy leakage. Rather, a low value of ε_{est} can be taken as evidence that an adversary implementing this particular, powerful attack will have a hard time inferring the presence of any given user upon observing the final model. If we suppose that the attack is strong, or even optimal, then we can infer that *any* attacker will not be able to perform MI successfully, and therefore our ε_{est} is a justifiable metric of the true privacy when the final model is released. Investigating conditions under which this could be proven would be a valuable direction for future work.

Aside from quantifying the privacy of releasing only the final model, our method allows us to explore how privacy properties are affected by varying aspects of training for which we have no tight formal analysis. As an important example (which we explore in experiments) we consider how the estimate changes if clients are constrained to participate a fixed number of times.

We also propose a simple extension to our method that allows us to estimate ε under the threat model where all model updates are observed. We use as the test statistic the *maximum over rounds* of the angle between the canary and the model delta at that round. Unfortunately in this case we can no longer express in closed form the distribution of max-over-rounds cosine of an canary that did not participate in training, because it depends on the trajectory of partially trained models, which is task and model specific. Our solution is to sample a set of unobserved canaries that are never included in model updates, but we still keep track of their cosines with each model delta and finally take the max. We approximate both the distributions of observed and unobserved maximum canary/model-delta cosines using Gaussian distributions and compute the optimal ε. The pseudocode for this modified procedure is provided in Algorithm 3. We will see that this method provides estimates of ε close to the analytical bounds under moderate amounts of noise, providing evidence that our attack is strong.

5 Experiments

In this section we present the results of experiments estimating the privacy leakage while training a model on a large-scale public federated learning dataset: the stackoverflow word prediction data/model of Reddi et al. [2020]. The model is a word-based LSTM with 4.1M parameters. We train the model for 2048 rounds with 167 clients per round, where each of the m=341k clients participates in exactly one round, amounting to a single epoch over the data. We use the adaptive clipping method of Andrew et al. [2021]. With preliminary manual tuning, we selected a client learning rate of 1.0 and server learning rate of 0.56 for all experiments because the choice gives good performance over a range of levels of DP noise. We always use 1k canaries for each set of cosines; experiments with intermediate iterates use 1k observed and 1k unobserved canaries. We fix $\delta = m^{-1.1}$. We consider

Noise	analytical ε	ε_{lo}-all	ε_{est}-all	ε_{lo}-final	ε_{est}-final
0	∞	6.240	45800	2.88	4.60
0.0496	300	6.238	382	1.11	1.97
0.0986	100	5.05	89.4	0.688	1.18
0.2317	30	0.407	2.693	0.311	0.569

Table 2: Comparing ε estimates using all model deltas vs. using the final model only. ε_{lo} is the empirical 95% lower bound from our modified Jagielski et al. [2020] method. For moderate noise, ε_{est}-all is in the ballpark of the analytical ε, providing evidence that the attack is strong and therefore the ε estimates are reliable. On the other hand, ε_{est}-final is far lower, indicating that when the final model is observed, privacy is better.

noise multipliers[3] in the range 0.0496 to 0.2317, corresponding to analytical ε estimates from 300 down to 30.[4] We also include experiments with clipping only (noise multiplier is 0). We note that across the range of noise multipliers, the participation of 1k canaries had no significant impact on model accuracy – at most causing a 0.1% relative decrease.

We also report a high-probability lower bound on ε that comes from applying a modified version of the method of Jagielski et al. [2020] to the set of cosines. That work uses Clopper-Pearson upper bounds on the achievable FPR and FNR of a thresholding classifier to derive a bound on ε. We make two changes: following Zanella-Béguelin et al. [2022], we use the tighter and more centered Jeffreys confidence interval for the upper bound on FNR at some threshold a, and we use the exact CDF of the null distribution for the FPR as described in Section 4. We refer to this lower bound as ε_{lo}. We set $\alpha = 0.05$ to get a 95%-confidence bound.

We first consider the case where the intermediate updates are released as described in Algorithm 3. The middle columns of Table 2 shows the results of these experiments over a range of noise multipliers. For the lower noise multipliers, our method easily separates the cosines of observed vs. unobserved canaries, producing very high estimates ε_{est}-all, which are much higher than lower bounds ε_{lo}-all estimated by previous work. This confirms our intuition that intermediate model updates give the adversary significant power to detect the presence of individuals in the data. It also provides evidence that the canary cosine attack is strong, increasing our confidence that the ε estimates assuming a weakened adversary that observes only the final model is not a severe underestimate.

The rightmost columns of Table 2 show the results of restricting the adversary to observe only the final model, as described in Algorithm 2. Now ε_{est} is significantly smaller than when the adversary has access to all intermediary updates. With clipping only, our estimate is 4.60, which is still essentially vacuous from a rigorous privacy perspective.[5] But with even a small amount of noise, we approach the high-privacy regime of $\varepsilon \sim 1$, confirming observations of practitioners that a small amount of noise is sufficient to prevent most memorization.

In Appendix D we provide analogous experiments on the federated EMNIST dataset, with similar results. In Appendix E we present further experiments to highlight the ability of our method to estimate privacy when we vary aspects of the training algorithm that may reasonably be expected to change privacy properties, but for which no tight analysis has been obtained.

6 Conclusion

We have introduced a novel method for empirically estimating the privacy loss during training of a model with DP-FedAvg. For natural production-sized problems (millions of parameters, hundreds of thousands of clients), it produces reasonable privacy estimates during the same single training run used to estimate model parameters, without degrading the utility of the model, and does not require any prior knowledge of the task, data or model.

[3]The noise multiplier is the ratio of the noise to the clip norm. When adaptive clipping is used, the clip norm varies across rounds, and the noise scales proportionally.

[4]Since each user participates once, we bound ε as the unamplified Gaussian mechanism applied once with no composition.

[5]An ε of 5 means that an attacker can go from a small suspicion that a user participated (say, 10%) to a very high degree of certainty (94%) [Desfontaines, 2018].

References

M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

B. Balle and Y.-X. Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

B. Balle, G. Cherubin, and J. Hayes. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1138–1156. IEEE, 2022.

J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

K. Bonawitz, P. Kairouz, B. Mcmahan, and D. Ramage. Federated learning and privacy. *Commun. ACM*, 65(4):90–97, mar 2022. ISSN 0001-0782. doi: 10.1145/3500240. URL `https://doi.org/10.1145/3500240`.

N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 2021)*, 2021.

N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy (SP)*, pages 1519–1519, Los Alamitos, CA, USA, May 2022. IEEE Computer Society. doi: 10.1109/SP46214.2022.00090. URL `https://doi.ieeecomputersociety.org/10.1109/SP46214.2022.00090`.

C. A. Choquette-Choo, H. B. McMahan, K. Rush, and A. Thakurta. Multi-epoch matrix factorization mechanisms for private machine learning. *arXiv preprint arXiv:2211.06530*, 2022.

S. Denissov, H. B. McMahan, J. K. Rush, A. Smith, and A. G. Thakurta. Improved differential privacy for SGD via optimal private linear operators on adaptive streams. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=i9XrHJoyLqJ`.

D. Desfontaines. Differential privacy in (a bit) more detail. `https://desfontain.es/privacy/differential-privacy-in-more-detail.html`, 2018. Accessed 2023-09-28.

Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 475–489, 2018.

C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL `http://dx.doi.org/10.1561/0400000042`.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Conference on Theory of Cryptography*, TCC '06, pages 265–284, New York, NY, USA, 2006.

A. C. Gilbert and A. McMillan. Property Testing for Differential Privacy. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 249–258. IEEE, 2018.

N. Haim, G. Vardi, G. Yehudai, michal Irani, and O. Shamir. Reconstructing training data from trained neural networks. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=Sxk8Bse3RKO`.

M. Jagielski, J. Ullman, and A. Oprea. Auditing differentially private machine learning: How private is private sgd? In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

M. Jagielski, S. Wu, A. Oprea, J. Ullman, and R. Geambasu. How to combine membership-inference attacks on multiple updated machine learning models. *Proceedings on Privacy Enhancing Technologies*, 3:211–232, 2023.

P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1376–1385, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/kairouz15.html`.

P. Kairouz, B. McMahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*, pages 5213–5225. PMLR, 2021a.

P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021b.

X. Liu and S. Oh. Minimax Optimal Estimation of Approximate Differential Privacy on Neighboring Databases. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

F. Lu, J. Munoz, M. Fuchs, T. LeBlond, E. V. Zaresky-Williams, E. Raff, F. Ferraro, and B. Testa. A general framework for auditing differentially private machine learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=AKM3C3tsSx3`.

S. Maddock, A. Sablayrolles, and P. Stock. Canife: Crafting canaries for empirical privacy measurement in federated learning, 2022. URL `https://arxiv.org/abs/2210.02912`.

B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017. URL `https://proceedings.mlr.press/v54/mcmahan17a.html`.

H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

M. Nasr, S. Songi, A. Thakurta, N. Papemoti, and N. Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 866–882. IEEE, 2021.

M. Nasr, J. Hayes, T. Steinke, B. Balle, F. Tramèr, M. Jagielski, N. Carlini, and A. Terzis. Tight auditing of differentially private machine learning. *arXiv preprint arXiv:2302.07956*, 2023.

K. Pillutla, G. Andrew, P. Kairouz, H. B. McMahan, A. Oprea, and S. Oh. Unleashing the power of randomization in auditing differentially private ml. *arXiv preprint arXiv:2305.18447*, 2023.

S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

T. Steinke. Composition of differential privacy privacy amplification by subsampling, 2022.

T. Steinke, M. Nasr, and M. Jagielski. Privacy auditing with one (1) training run. *arXiv preprint arXiv:2305.08846*, 2023.

S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

S. Zanella-Beguelin, L. Wutschitz, S. Tople, A. Salem, V. Rühle, A. Paverd, M. Naseri, B. Köpf, and D. Jones. Bayesian estimation of differential privacy. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 40624–40636. PMLR, 23–29 Jul 2023.

S. Zanella-Béguelin, L. Wutschitz, S. Tople, A. Salem, V. Rühle, A. Paverd, M. Naseri, B. Köpf, and D. Jones. Bayesian estimation of differential privacy, 2022. URL `https://arxiv.org/abs/2206.05199`.

A Proofs of theorems from the main text

Theorem 3.1. *Let S be sampled uniformly from the unit sphere in $\mathbb{R}^d$, and let $\tau = \langle S, v\rangle / \|v\| \in [-1, 1]$ be the cosine similarity between S and some arbitrary independent nonzero vector v. Then, the probability density function of τ is*

$$f_d(\tau) = \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}}(1-\tau^2)^{\frac{d-3}{2}}.$$

Proof. Due to the rotational symmetry of the distribution of S, without loss of generality, we can take v to be constant. First we describe the distribution of the angle $\theta \in [0, \pi]$ between s and v, then change variables to get the distribution of its cosine τ. Consider the spherical cap of points on the d-sphere with angle to v less or equal to θ, having $(d{-}1)$-measure $A_d(\theta)$. The boundary of $A_d(\theta)$ is a $(d{-}1)$-sphere with radius $\sin\theta$ and $(d{-}2)$-measure $M_d(\theta) = S_{d-1}\sin^{d-2}\theta$, where $S_d = 2\pi^{\frac{d}{2}}/\Gamma(\frac{d}{2})$ is the surface area of the unit d-sphere. (For example, the boundary of the 3-d spherical cap with maximum angle θ is a circle (2-sphere) with radius $\sin\theta$ and circumference $2\pi\sin(\theta)$.) Normalizing by the total area of the sphere S_d, the density of the angle is

$$\begin{aligned}
\phi_d(\theta) &= S_d^{-1}\frac{d}{d\theta}A_d(\theta) \\
&= \left(\frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}\right)^{-1}\left(\frac{2\pi^{\frac{d-1}{2}}}{\Gamma(\frac{d-1}{2})}\right)\sin^{d-2}\theta \\
&= \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}}\sin^{d-2}\theta.
\end{aligned}$$

Now change variables to express it in terms of the angle cosine $\tau = \cos(\theta) \in [-1, 1]$:

$$\begin{aligned}
f_d(\tau) &= \phi_d(\arccos\tau)\cdot\left|\frac{d}{d\tau}\arccos(\tau)\right| \\
&= \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}}\left[\sin(\arccos\tau)\right]^{d-2}\left|-\frac{1}{\sqrt{1-\tau^2}}\right| \\
&= \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}}\frac{\left(\sqrt{1-\tau^2}\right)^{d-2}}{\sqrt{1-\tau^2}} \\
&= \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}}(1-\tau^2)^{\frac{d-3}{2}}.
\end{aligned}$$

□

Theorem 3.2. *In the setting of Theorem 3.1, we have that $\tau\sqrt{d}$ converges in distribution to $\mathcal{N}(0,1)$ as $d \to \infty$, i.e., $\lim_{d\to\infty}\mathbb{P}(\tau \le \lambda/\sqrt{d}) = \mathbb{P}_{Z\sim\mathcal{N}(0,1)}(Z \le \lambda)$.*

Proof. The distribution function of $t \in [-\sqrt{d}, \sqrt{d}]$ is

$$\hat{f}_d(t) = \frac{1}{\sqrt{d}} \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi}} \left(1 - (t/\sqrt{d})^2\right)^{\frac{d-3}{2}}$$
$$= \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi d}} \left(1 - t^2/d\right)^{\frac{d-3}{2}}.$$

Taking the limit:

$$\lim_{d\to\infty} \hat{f}_d(t) = \left(\lim_{d\to\infty} \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})\sqrt{\pi d}}\right) \cdot \left(\lim_{d\to\infty} \left(1 - t^2/d\right)^{\frac{d}{2}}\right) \cdot \left(\lim_{d\to\infty} \left(1 - t^2/d\right)^{-\frac{3}{2}}\right)$$
$$= \frac{1}{\sqrt{2\pi}} \cdot e^{-t^2/2} \cdot 1,$$

where we have used the fact that $\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} \sim \sqrt{d/2}$.

□

Lemma A.1. *If τ is distributed according to the cosine angle distribution described in Theorem 3.1, then Var$[\tau] = 1/d$.*

Proof. Let $x = (x_1, \ldots, x_d)$ be uniform on the unit d-sphere. Then $x_1 = \langle x, e_1 \rangle$ has the required distribution, where e_1 is the first standard basis vector. $\mathbb{E}[x_1]$ is zero, so we are interested in Var$[x_1] = \mathbb{E}\big[x_1^2\big]$. Since $\sum_i x_i^2 = 1$, we have that $\mathbb{E}\big[\sum_i x_i^2\big] = \sum_i \mathbb{E}\big[x_i^2\big] = 1$. But all of the x_i have the same distribution, so $\mathbb{E}\big[x_1^2\big] = 1/d$. □

Theorem 3.3. *For $d \in \mathbb{N}$, let $k = o(d)$, but $k = \omega(1)$. For $i = 1 \ldots k$, let c_i sampled i.i.d. from the unit sphere in d dimensions. Let $Z \sim \mathcal{N}(0; I_d)$. Let $\sigma > 0$, and define the mechanism result $\rho = \sum_{j=1}^k c_j + \sigma Z$, and the cosine values $g_j = \frac{\langle c_j, \rho \rangle}{\|\rho\|}$. Write the empirical mean of the cosines $\hat{\mu} = \frac{1}{k}\sum_{j=1}^k g_j$, and the empirical variance $\hat{\sigma}^2 = \frac{1}{k}\sum_{j=1}^k \big(g_j - \hat{\mu}\big)^2$. Then as $d \to \infty$, $\sqrt{d}\hat{\mu} \xrightarrow{p} 1/\sigma$ and $d\hat{\sigma}^2 \xrightarrow{p} 1$.*

Proof. Rewrite

$$\sqrt{d}\hat{\mu} = \sqrt{d}\left(\frac{1}{k}\sum_i \frac{\langle c_i, \rho \rangle}{||\rho||}\right)$$
$$= \left(\frac{||\rho||}{\sqrt{d}}\right)^{-1} \left(\frac{1}{k}\sum_i \langle c_i, \rho \rangle\right).$$

We will show that $\frac{||\rho||^2}{d} \xrightarrow{p} \sigma^2$, while $\frac{1}{k}\sum_i \langle c_i, \rho \rangle \xrightarrow{p} 1$.

Note that $||Z||^2$ is Chi-squared distributed with mean d and variance $2d$. Also, for all $i \neq j$, $\langle c_i, c_j \rangle$ is distributed according to the cosine distribution discussed in Theorem 3.1 and Lemma A.1 with mean 0 and variance $1/d$. Therefore,

$$\mathbb{E}\left[\frac{||\rho||^2}{d}\right] = \frac{1}{d}\left\langle \sum_i c_i + \sigma Z,\ \sum_i c_i + \sigma Z \right\rangle$$
$$= \frac{1}{d}\left(\sum_{i,j} \mathbb{E}[\langle c_i, c_j \rangle] + 2\sigma \sum_i \mathbb{E}[\langle Z, c_i \rangle] + \sigma^2 \mathbb{E}\big[||Z||^2\big]\right)$$
$$= \frac{1}{d}\left(k + \sigma^2 d\right)$$
$$= \sigma^2 + o(1).$$

Next, note that all of the following dot products are pairwise uncorrelated: $\langle Z, Z\rangle$, $\langle Z, c_i\rangle$ (for all i), and $\langle c_i, c_j\rangle$ (for all i, j). Therefore the variance decomposes:

$$\begin{aligned}\mathrm{Var}\left[\frac{||\rho||^2}{d}\right] &= \frac{1}{d^2}\left(\sum_{i,j}\mathrm{Var}[\langle c_i, c_j\rangle] + 2\sigma^2\sum_i \mathrm{Var}[\langle Z, c_i\rangle] + \sigma^4 \mathrm{Var}\big[||Z||^2\big]\right) \\ &= \frac{1}{d^2}\left(\frac{k(k-1)}{d} + 2k\sigma^2 + 2\sigma^4 d\right), \\ &= O(d^{-1}).\end{aligned}$$

Taken together, these imply that $\frac{||\rho||}{\sqrt{d}} \xrightarrow{p} \sigma$.

Now reusing many of the same calculations,

$$\mathbb{E}\left[\frac{1}{k}\sum_i \langle c_i, \rho\rangle\right] = \frac{1}{k}\left(k + \sum_{i\neq j}\mathbb{E}[\langle c_i, c_j\rangle] + \sigma\sum_i \mathbb{E}[\langle Z, c_i\rangle]\right) = 1,$$

and

$$\begin{aligned}\mathrm{Var}\left[\frac{1}{k}\sum_i \langle c_i, \rho\rangle\right] &= \frac{1}{k^2}\left(\sum_{i\neq j}\mathrm{Var}[\langle c_i, c_j\rangle] + \sigma^2\sum_i \mathrm{Var}[\langle Z, c_i\rangle]\right) \\ &= \frac{1}{k^2}\left(\frac{k(k-1)}{d} + k\sigma^2\right) \\ &= o(1),\end{aligned}$$

which together imply that $\frac{1}{k}\sum_i \langle c_i, \rho\rangle \xrightarrow{p} 1$.

Now consider

$$\begin{aligned}d\hat{\sigma}^2 &= d\left(\frac{1}{k}\sum_i g_i^2 - \left(\frac{1}{k}\sum_i g_i\right)^2\right) \\ &= \frac{d}{||\rho||^2}\left(\frac{1}{k}\sum_i \langle c_i, \rho\rangle^2 - \left(\frac{1}{k}\sum_i \langle c_i, \rho\rangle\right)^2\right).\end{aligned}$$

We already have that $\frac{d}{||\rho||^2} \xrightarrow{p} \frac{1}{\sigma^2}$ and $\frac{1}{k}\sum_i \langle c_i, \rho\rangle \xrightarrow{p} 1$, so it will be sufficient to show $\frac{1}{k}\sum_i \langle c_i, \rho\rangle^2 \xrightarrow{p} 1 + \sigma^2$. Again using the uncorrelatedness of all pairs of dot products under consideration,

$$\begin{aligned}\mathbb{E}\left[\frac{1}{k}\sum_i \langle c_i, \rho\rangle^2\right] &= \frac{1}{k}\sum_i \mathbb{E}\left[\left(\sum_j \langle c_i, c_j\rangle + \langle c_i, \sigma Z\rangle\right)^2\right] \\ &= \frac{1}{k}\sum_i\left(\sum_{j,\ell}\mathbb{E}[\langle c_i, c_j\rangle\langle c_i, c_\ell\rangle] + 2\sum_j \mathbb{E}[\langle c_i, c_j\rangle\langle c_i, \sigma Z\rangle] + \mathbb{E}\big[\langle c_i, \sigma Z\rangle^2\big]\right) \\ &= \frac{1}{k}\sum_i\left(\sum_{j,\ell}\mathbb{I}\{j = i, \ell = i\} + 0 + \sigma^2\right) \\ &= 1 + \sigma^2.\end{aligned}$$

Also,

$$\mathrm{Var}\left[\frac{1}{k}\sum_i \langle c_i, \rho\rangle^2\right] = \frac{1}{k^2}\sum_i\left(\sum_{j,\ell}\mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, c_\ell\rangle] + 2\sum_j \mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, \sigma Z\rangle] + \mathrm{Var}\big[\langle c_i, \sigma Z\rangle^2\big]\right).$$

We'll bound each of these terms to show the sum is $o(1)$.

First, look at $\mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, c_\ell\rangle]$. If $i = j = \ell$, it is 0. If $j \neq \ell$,

$$\begin{aligned}
\mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, c_\ell\rangle] &= \mathbb{E}\big[\langle c_i, c_j\rangle^2\langle c_i, c_\ell\rangle^2\big] - \mathbb{E}[\langle c_i, c_j\rangle\langle c_i, c_\ell\rangle]^2 \\
&= \mathbb{E}\big[\mathbb{E}\big[\langle c_i, c_j\rangle^2\langle c_i, c_\ell\rangle^2 \mid c_i\big]\big] \\
&= \mathbb{E}\big[\mathbb{E}\big[\langle c_i, c_j\rangle^2 \mid c_i\big]\,\mathbb{E}\big[\langle c_i, c_\ell\rangle^2 \mid c_i\big]\big] \\
&= \begin{cases} 1/d & j = i \text{ or } \ell = i, \\ 1/d^2 & j \neq i \text{ and } \ell \neq i, \end{cases}
\end{aligned}$$

and if $j = \ell \neq i$,

$$\begin{aligned}
\mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, c_\ell\rangle] &= \mathrm{Var}\big[\langle c_i, c_j\rangle^2\big] \\
&= \mathbb{E}\big[\langle c_i, c_j\rangle^4\big] - \mathbb{E}\big[\langle c_i, c_j\rangle^2\big]^2 \\
&\leq \mathbb{E}\big[\langle c_i, c_j\rangle^4\big] \\
&\leq \mathbb{E}\big[\langle c_i, c_j\rangle^2\big] \\
&= 1/d.
\end{aligned}$$

Together we have

$$\frac{1}{k^2}\sum_{i,j,\ell}\mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, c_\ell\rangle] \leq \frac{1}{k^2}\left(\frac{2k(k-1)}{d} + \frac{k(k-1)(k-2)}{d^2} + \frac{k(k-1)}{d}\right) = O(d^{-1}).$$

Now for $\mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, \sigma Z\rangle]$. If $i = j$, then it is σ^2. If $i \neq j$,

$$\begin{aligned}
\mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, \sigma Z\rangle] &= \mathbb{E}\big[\langle c_i, c_j\rangle^2\langle c_i, \sigma Z\rangle^2\big] - \mathbb{E}[\langle c_i, c_j\rangle\langle c_i, \sigma Z\rangle]^2 \\
&= \mathbb{E}\big[\mathbb{E}\big[\langle c_i, c_j\rangle^2 | c_i\big]\,\mathbb{E}\big[\langle c_i, \sigma Z\rangle^2 | c_i\big]\big] \\
&= \sigma^2/d.
\end{aligned}$$

So,

$$\frac{2}{k^2}\sum_{i,j}\mathrm{Var}[\langle c_i, c_j\rangle\langle c_i, \sigma Z\rangle] = \frac{2}{k^2}\left(k\sigma^2 + \frac{k(k-1)\sigma^2}{d}\right) = o(1).$$

Finally, $\langle c_i, Z\rangle^2$ is Chi-squared distributed with one degree of freedom, so $\mathrm{Var}\big[\langle c_i, \sigma Z\rangle^2\big] = 2\sigma^4$, and

$$\frac{1}{k^2}\sum_{i}\mathrm{Var}\big[\langle c_i, \sigma Z\rangle^2\big] = \frac{2\sigma^4}{k} = o(1).$$

□

B Algorithm for exact computation of ε comparing two Gaussian distributions

In this section we give the details of the computation for estimating ε when $A(D)$ and $A(D')$ are both Gaussian-distributed with different variances.

Let distribution under $A(D)$ be $P_1 = \mathcal{N}(\mu_1, \sigma_1^2)$ and the distribution under $A(D')$ be $P_2 = \mathcal{N}(\mu_2, \sigma_2^2)$ with densities p_1 and p_2 respectively. Define $f_{P_1||P_2}(x) = \log\frac{p_1(x)}{p_2(x)}$. Now $Z_1 = f_{P_1||P_2}(X_1)$ with $X_1 \sim P_1$ is the privacy loss random variable. Symmetrically define $Z_2 = f_{P_2||P_1}(X_2)$ with $X_2 \sim P_2$.

From Steinke [2022] Prop. 7 we have that (ϵ, δ)-DP implies

$$\Pr[Z_1 > \varepsilon] - e^{\varepsilon}\Pr[-Z_2 > \varepsilon] \leq \delta \text{ and } \Pr[Z_2 > \varepsilon] - e^{\varepsilon}\Pr[-Z_1 > \varepsilon] \leq \delta.$$

Now we can compute

$$
\begin{aligned}
f_{P_1||P_2}(x) &= \log \frac{p_1(x)}{p_2(x)} \\
&= \log \left(\frac{\sigma_2}{\sigma_1} \exp \left(-\frac{1}{2} \left[\frac{(x-\mu_1)^2}{\sigma_1^2} - \frac{(x-\mu_2)^2}{\sigma_2^2} \right] \right) \right) \\
&= \log \sigma_2 - \log \sigma_1 + \frac{(x-\mu_2)^2}{2\sigma_2^2} - \frac{(x-\mu_1)^2}{2\sigma_1^2} \\
&= ax^2 + bx + c
\end{aligned}
$$

where

$$
\begin{aligned}
a &= \frac{1}{2} \left(\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2} \right), \\
b &= \frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}, \\
\text{and } c &= \frac{1}{2} \left(\left(\frac{\mu_2}{\sigma_2} \right)^2 - \left(\frac{\mu_1}{\sigma_1} \right)^2 \right) + \log \sigma_2 - \log \sigma_1.
\end{aligned}
$$

To compute $\Pr[Z_1 > \varepsilon]$, we need $\Pr\left[aX_1^2 + bX_1 + (c-\varepsilon) > 0\right]$ with $X_1 \sim P_1$. To do so, divide the range of X_1 into intervals according to the zeros of $R(x) = ax^2 + bx + (c-\varepsilon)$. For example, if R has roots $r_1 < r_2$ and a is positive, we can compute $\Pr[Z_1 > \varepsilon] = \Pr[X_1 < r_1] + \Pr[X_1 > r_2]$, using the CDF of the Normal distribution. This requires considering a few cases, depending on the sign of a and the sign of the determinant $b^2 - 4a(c-\varepsilon)$.

Now note that $f_{P_2||P_1} = -f_{P_1||P_2}$, so

$$
\Pr[-Z_2 > \varepsilon] = \Pr\left[-f_{P_2||P_1}(X_2) > \varepsilon\right] = \Pr[aX_2^2 + bX_2 + (c-\varepsilon) > 0].
$$

So the two events we are interested in ($Z_1 > \varepsilon$ and $-Z_2 > \varepsilon$) are the same, only when we compute their probabilities according to P_1 vs. P_2 we use different values for μ and σ.

For numerical stability, the probabilities should be computed in the log domain. So we get

$$
\begin{aligned}
\log \delta &\geq \log\left(\Pr[Z_1 > \varepsilon] - e^{\varepsilon} \Pr[-Z_2 > \varepsilon]\right) \\
&= \log \Pr[Z_1 > \varepsilon] + \log\left(1 - \exp(\varepsilon + \log \Pr[-Z_2 > \varepsilon] - \log \Pr[Z_1 > \varepsilon])\right).
\end{aligned}
$$

Note it can happen that $\Pr[Z_1 > \varepsilon] < e^{\varepsilon} \Pr[-Z_2 > \varepsilon]$ in which case the corresponding bound is invalid. A final trick we suggest for numerical stability is if $X \sim \mathcal{N}(\mu, \sigma^2)$ to use $\Pr(X < \mu; t, \sigma^2)$ in place of $\Pr(X > t; \mu, \sigma^2)$.

Now to determine ε at a given target δ, one can perform a line search over ε to find the value that matches.

C Algorithms

Algorithms 1-3 illustrate the methods described in the main text.

D Supplementary experimental results on EMNIST dataset

In the main paper we presented results on the Stackoverflow federated word prediction task. Here we present similar results on the EMNIST character recognition dataset. It contains 814k characters written by 3383 users. The model is a CNN with 1.2M parameters. The users are shuffled and we train for five epochs with 34 clients per round. The optimizers on client and server are both SGD, with learning rates 0.031 and 1.0 respectively, and momentum of 0.9 on the server. The client batch size is 16.

Table 3 shows the empirical epsilon estimates using either all model iterates or only the final model. As with Stackoverflow next word prediction, using all iterates and a low amount of noise gives us

Algorithm 2 Privacy estimation via random canaries

Input: Client selection function **clients**, client training functions τ_i, canary selection function **canaries**, set of canary updates c_j, number of rounds T, initial parameters θ_0, noise generator Z, ℓ_2 clip norm function S, privacy parameter δ, server learning rate η
for $t = 1, \ldots, T$ **do**
  $\rho = \vec{0}$
  for $i \in \textbf{clients}(t)$ **do**
    $\rho \leftarrow \rho + \text{CLIP}(\tau_i(\theta_{t-1}); S(t))$
  for $j \in \textbf{canaries}(t)$ **do**
    $\rho \leftarrow \rho + \text{PROJ}(c_j; S(t))$
  $m = |\textbf{clients}(t)| + |\textbf{canaries}(t)|$
  $\theta_t \leftarrow \theta_{t-1} + \eta(\rho + Z(t))/m$
for all canaries j **do**
  $g_j \leftarrow \langle c_j, \theta_T \rangle / (||c_j|| \cdot ||\theta_T||)$
$\mu, \sigma \leftarrow \textbf{mean}(\{g_j\}), \textbf{std}(\{g_j\})$
$\varepsilon \leftarrow \varepsilon(\mathcal{N}(0, 1/d) \,||\, \mathcal{N}(\mu, \sigma^2); \delta)$

function $\text{CLIP}(x; \kappa)$
  return $x \cdot \min(1, \kappa/||x||)$

function $\text{PROJ}(x; \kappa)$
  return $x \cdot \kappa/||x||$

Algorithm 3 Privacy estimation via random canaries using all iterates

Input: As in Algorithm 2, but with *unobserved* canary updates c_j^0 and *observed* canary updates c_j^1.
for $t = 1, \ldots, T$ **do**
  $\rho = \vec{0}$
  for $i \in \textbf{clients}(t)$ **do**
    $\rho \leftarrow \rho + \text{CLIP}(\tau_i(\theta_{t-1}); S(t))$
  for $j \in \textbf{canaries}(t)$ **do**
    $\rho \leftarrow \rho + \text{PROJ}(c_j^1; S(t))$
  $m = |\textbf{clients}(t)| + |\textbf{canaries}(t)|$
  $\bar{\rho} \leftarrow (\rho + Z(t))/m$
  for all canaries j **do**
    $g_{t,j}^0 = \langle c_j^0, \bar{\rho} \rangle / (||c_j^0|| \cdot ||\bar{\rho}||)$
    $g_{t,j}^1 = \langle c_j^1, \bar{\rho} \rangle / (||c_j^1|| \cdot ||\bar{\rho}||)$
  $\theta_t \leftarrow \theta_{t-1} + \eta\bar{\rho}$
for all canaries j **do**
  $g_j^0 \leftarrow \max_t g_{t,j}^0$
  $g_j^1 \leftarrow \max_t g_{t,j}^1$
$\mu_0, \sigma_0 \leftarrow \textbf{mean}(\{g_j^0\}), \textbf{std}(\{g_j^0\})$
$\mu_1, \sigma_1 \leftarrow \textbf{mean}(\{g_j^1\}), \textbf{std}(\{g_j^1\})$
$\varepsilon \leftarrow \varepsilon(\mathcal{N}(\mu_0, \sigma_0^2) \,||\, \mathcal{N}(\mu_1, \sigma_1^2); \delta)$

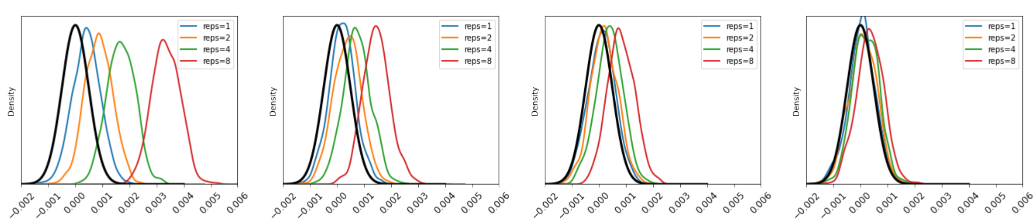

Figure 1: Density plots of cosine values with four values of noise corresponding to analytical epsilons (∞, 300, 100, 30) and four values of canary repetitions (1, 2, 4, 8). The black curve in each plot is the pdf of the null distribution $\mathcal{N}(0, 1/d)$. With no noise ($\varepsilon = \infty$), the distributions are easily separable, with increasing separation for more canary repetitions. At higher levels of noise, distributions are less separable, even with several repetitions.

Noise	analytical ε	ε_{lo}-all	ε_{est}-all	ε_{lo}-final	ε_{est}-final
0.0	∞	6.25	48300	3.86	5.72
0.16	33.8	2.87	17.9	1.01	1.20
0.18	28.1	2.32	12.0	0.788	1.15
0.195	24.8	2.02	8.88	0.723	1.08
0.25	16.9	0.896	3.86	0.550	0.818
0.315	12.0	0.315	1.50	0.216	0.737

Table 3: Comparing ε estimates using all model deltas vs. using the final model only. ε_{lo} is the empirical 95% lower bound from our modified Jagielski et al. [2020] method. The high values of ε_{est}-all indicate that membership inference is easy when the attacker has access to all iterates. On the other hand, when only the final model is observed, ε_{est}-final is far lower.

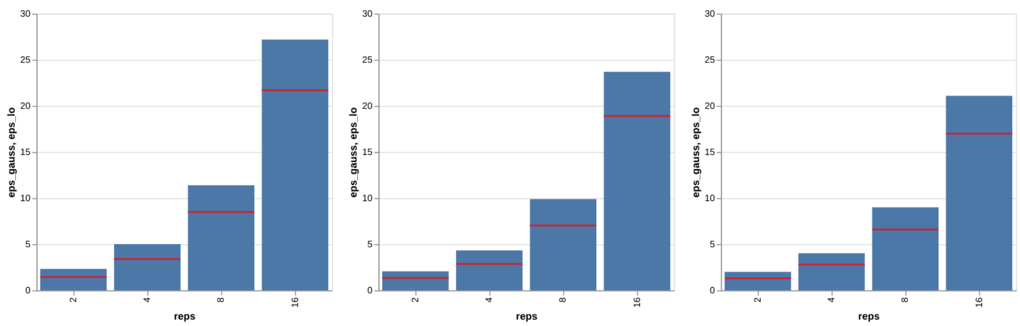

Figure 2: Blue bars are our ε_{est} and red ticks are the ε_{lo} 95%-confidence lower bound for three noise multipliers (0.16, 0.18, 0.195) and four numbers of canary repetitions. Our estimate of epsilon increases sharply with the number of canary repetitions, confirming that limiting client participation improves privacy.

estimates close to the analytical upper bound, while using only the final model gives a much smaller estimate.

Figure 2 demonstrates the effect of increasing the number of canary repetitions for EMNIST. The results are qualitatively similar to the case of Stackoverflow.

E Experiments with multiple canary presentations

Here we highlight the ability of our method to estimate privacy when we vary not only the threat model, but also aspects of the training algorithm that may reasonably be expected to change privacy properties, but for which no tight analysis has been obtained. We consider presenting each canary a fixed multiple number of times, modeling the scenario in which clients are only allowed to check in for training every so often. In practice, a client need not participate in every period, but to obtain worst-case estimates, we present the canary in every period.

In Figure 1 we show kernel density estimation plots of the canary cosine sets. As the number of presentations increases in each plot, the distributions become more and more clearly separated. On the other hand as the amount of noise increases across the three plots, they converge to the null distribution. Also visible on this figure is that the distributions are roughly Gaussian-shaped, justifying the Gaussian approximation that is used in our estimation method. In Appendix G we give quantitative evidence for this observation. Finally we compare ε_{lo} to our ε_{est} with multiple canary presentations in Figure 3. For each noise level, ε_{est} increases dramatically with increasing presentations, confirming our intuition that seeing examples multiple times dramatically reduces privacy.

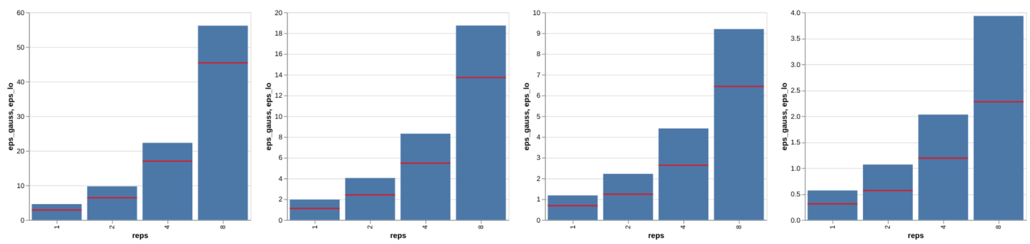

Figure 3: Blue bars are our ε_{est} and red ticks are the ε_{lo} 95%-confidence lower bound for four values of noise corresponding to analytical epsilons (∞, 300, 100, 30) and four values of canary repetitions (1, 2, 4, 8). Note the difference of y-axis scales in each plot. Our estimate of epsilon increases sharply with the number of canary repetitions, confirming that limiting client participation improves privacy.

F Related Work

Private federated fearning. DP Federated Averaging (DP-FedAvg) [McMahan et al., 2018] is a user-level DP version of the well-known Federated Averaging (FedAvg) algorithm [McMahan et al., 2017] for training ML models in a distributed fashion. In FedAvg, a central server interacts with a set of clients to train a global model iteratively over multiple rounds. In each round, the server sends the current global model to a subset of clients, who train local models using their training data, and send the model updates back to the server. The server aggregates the model updates via the Gaussian mechanism, in which each update is clipped to bound its ℓ_2 norm before averaging and adding Gaussian noise proportional to the clipping norm sufficient to mask the influence of individual users, and incorporates the aggregate update into the global model. DP-FedAvg can rely on privacy amplification from the sampling of clients at each round, but more sophisticated methods can handle arbitrary participation patterns [Kairouz et al., 2021a, Choquette-Choo et al., 2022].

Privacy auditing. Privacy auditing [Ding et al., 2018, Liu and Oh, 2019, Gilbert and McMillan, 2018, Jagielski et al., 2020] provides techniques for empirically auditing the privacy leakage of an algorithm. The main technique used for privacy auditing is mounting a membership inference attack [Shokri et al., 2017, Yeom et al., 2018, Carlini et al., 2022], and translating the success of the adversary into an ε estimate using Eq. (1) directly.

Most privacy auditing techniques [Jagielski et al., 2020, Nasr et al., 2021, Lu et al., 2022, Zanella-Béguelin et al., 2022] have been designed for centralized settings, with the exception of CANIFE [Maddock et al., 2022], suitable for privacy auditing of federated learning deployments. CANIFE operates under a strong adversarial model, assuming knowledge of all intermediary model updates, as well as local model updates sent by a subset of clients in each round of training. CANIFE crafts data poisoning canaries adaptively, with the goal of generating model updates orthogonal to updates sent by other clients in each round. We argue that when the model dimensionality is sufficiently high, such crafting is unnecessary, since a randomly chosen canary update will already be essentially orthogonal to the true updates with high probability. CANIFE also computes a *per-round* privacy measure which it extrapolates into a measure for the entire training run by estimating an equivalent per-round noise $\hat{\sigma}_r$ and then composing the RDP of the repeated Poisson subsampled Gaussian mechanism. However in practice FL systems do not use Poisson subsampling due to the infeasiblity of sampling clients i.i.d. at each round. Our method flexibly estimates the privacy loss in the context of arbitrary participation patterns, for example passing over the data in epochs, or the difficult-to-characterize *de facto* pattern of participation in a deployed system, which may include techniques intended to amplify privacy such as limits on client participation within temporal periods such as one day.

We empirically compare our approach with CANIFE in Appendix H and discuss the assumptions on the auditor's knowledge and capabilities for all recent approaches (including ours) in Appendix I.

G Gaussianity of cosine statistics

To our knowledge, there is no way of confidently inferring that a set of samples comes from a given distribution, or even that they come from a distribution that is close to the given distribution in some metric. To quantify the error of our approximation of the cosine statistics with a Gaussian distribution,

Noise	1 rep	2 reps	3 reps	5 reps	10 reps
0	0.464	**0.590**	0.396	0.407	0.196
0.1023	**0.976**	0.422	0.340	0.432	0.116
0.2344	0.326	0.157	0.347	**0.951**	0.401

Table 4: Anderson statistics for each set of canary-cosine samples whose densities are shown in Figure 1. The Anderson test rejects at a 1% significance level if the statistic is greater than 1.088, and rejects at 15% significance if the statistic is greater than 0.574. Under the null hypothesis that all 15 (independent) distributions are Gaussian, we can compute that the probability of observing three or more values with Anderson test statistic greater than 0.574 is 68%.

we apply the Anderson test to each set of cosines in Table 4. It gives us some confidence to see that a strong goodness-of-fit test cannot rule out that the distributions are Gaussian. This is a more quantitative claim than visually comparing a histogram or empirical CDF, as is commonly done.

H Empirical comparison with CANIFE

As discussed in the main text, our method is significantly more general than the CANIFE method of Maddock et al. [2022]. CANIFE periodically audits individual rounds to get a per-round $\hat{\varepsilon}_r$, estimates the noise for the round $\hat{\sigma}_r$ by inverting the computation of ε for the Gaussian mechanism, and uses standard composition theorems to determine a final cumulative epsilon. Therefore if the assumptions of those composition theorems do not strictly hold (for example, if clients are not sampled uniformly and independently at each round), the estimate will be inaccurate. Also the step of crafting canaries is model/dataset specific, and computationally expensive.

It is still interesting to see how the methods compare in the limited setting where CANIFE's assumptions do hold. We trained a two-layer feedforward network on the fashion MNIST dataset. Following experiments in Maddock et al. [2022], we used a canary design pool size of 512, took 2500 canary optimization steps to find a canary example optimizing pixels and soft label, ran auditing every 100 rounds with 100 attack scores on each auditing round. We trained with a clip norm of 1.0 and noise multiplier of 0.2 for one epoch with a batch size of 128, which corresponds to an analytical ε of 34.5.

CANIFE output a ε of mean 0.879 and standard deviation 0.124. Our method (using 1000 seen and 1000 unseen model canaries) estimated ε with a mean of 6.76 and std of 1.06, much closer to the analytical bound.

I Comparison of assumptions and requirements of empirical privacy estimation methods

As discussed in Section F, related work in privacy auditing/estimation rely on various assumptions on the auditor's knowledge and capability. Here we summarize the major differences.

		auditor controls	auditor receives
Central	Jagielski et al. [2020]	train data	final model
	Zanella-Beguelin et al. [2023]	train data	final model
	Pillutla et al. [2023]	train data	final model
	Steinke et al. [2023]	train data	final model
	Jagielski et al. [2023]	train data	intermediate models
	Nasr et al. [2023]	train data, privacy noise, minibatch	intermediate models
FL	Algorithm 2	client model update	final model
	Algorithm 3	client model update	intermediate models
	CANIFE [Maddock et al., 2022]	client sample, privacy noise, minibatch	intermediate models

Table 5: Examples of different assumptions in the literature. For each paper, we state the most relaxed condition the technique can be applied to, since they can be generalized in straightforward manner to scenarios with more strict assumptions on the auditor's control and observation. Within each category of {central training, federated learning}, the lists are ordered from least to most strict assumptions.

Noise	runs/canaries	0.1	0.3	0.5	0.7	0.9
0.0496	1/1000	1.59	1.75	1.97	2.15	2.46
0.0496	10/100	1.65	1.80	1.92	2.16	2.32
0.0986	1/1000	0.81	1.06	1.18	1.33	1.54
0.0986	10/100	0.87	1.03	1.16	1.36	1.78

Table 6: Quantiles of $\hat{\varepsilon}$ over fifty experiments using either one run with 1000 canaries or ten runs with 100 canaries each. For both noise multipliers, the distributions are very close.

Standard assumption in auditing centralized private training algorithm is a black-box setting where the auditor only get to control the training data and observes the final model output. In practice, many private training algorithms guarantee privacy under releasing all the intermediate model checkpoints. One can hope to improve the estimate of privacy by using those check points as in [Jagielski et al., 2023]. If the auditor can use information about how the minibatch sequence is drawn and the distribution of the privacy noise, which is equivalent to assuming that the auditor controls the privacy noise and the minibatch, one can further improve the estimates.

In the federated learning scenario, we assume canary client can return any model update. Note that while CANIFE only controls the sample of the canary client and not the model update directly, CANIFE utilises the Renyi-DP accountant with Poisson subsampling implemented via the Opacus library, which is equivalent to the auditor fully controlling the sequence of minibatches (cohorts in FL terminology). Further, the privacy noise is assumed to be independent spherical Gaussian, which is equivalent to the auditor fully controlling the noise.

J Experiments comparing when multiple runs are used

In the limit of high model dimensionality, canaries are essentially mutually orthogonal, and therefore they will interfere minimally with each other's cosines to the model. In this section we give evidence that even in the range of model dimensionalities explored in our experiments, including many canaries in one run does not significantly perturb the estimated epsilon values. Ideally we would train 1000 models each with one canary to collect a set of truly independent statistics. However this is infeasible, particularly if we want to perform the entire process multiple times to obtain confidence intervals. Instead, we reduce the number of canaries per run by a factor of ten and train ten independent models to collect a total of 1000 canary cosine statistics from which to estimate ε. We repeated the experiment 50 times for two different noise multipliers, which still amounts to training a total of 1000 models. (Ten runs, two settings, fifty repetitions.)

The results on the stackoverflow dataset with the same setup as in Section 5 are shown in table 6. We report the 0.1, 0.3, 0.5, 0.7 and 0.9 quantile of the distribution of $\hat{\varepsilon}$ over 50 experiments. For both noise multipliers, the distributions are quite close. Our epsilon estimates do not seem to vary significantly even as the number of canaries per run varies by an order of magnitude.