AUDITING f-DIFFERENTIAL PRIVACY IN ONE RUN

Anonymous authors
Paper under double-blind review

ABSTRACT

Empirical auditing has emerged as a means of catching some of the flaws in the implementation of privacy-preserving algorithms. Existing auditing mechanisms, however, are either computationally inefficient – requiring multiple runs of the machine learning algorithms —- or suboptimal in calculating an empirical privacy. In this work, we present a tight and efficient auditing procedure and analysis that can effectively assess the privacy of mechanisms. Our approach is efficient; similar to the recent work of Steinke, Nasr, and Jagielski (2023), our auditing procedure leverages the randomness of examples in the input dataset and requires only a single (training) run of the target mechanism. And it is more accurate; we provide a novel analysis that enables us to achieve tight empirical privacy estimates by using the hypothesized f-DP curve of the mechanism, which provides a more accurate measure of privacy than the traditional ϵ, δ differential privacy parameters. We use our auditing procure and analysis to obtain empirical privacy, demonstrating that our auditing procedure delivers tighter privacy estimates.

1 INTRODUCTION

Differentially private machine learning (Chaudhuri et al., 2011; Abadi et al., 2016) has emerged as a principled solution to learning models from private data while still preserving privacy. Differential privacy (Dwork, 2006) is a cryptographically motivated definition, which requires an algorithm to possess certain properties: specifically, a randomized mechanism is differentially private if it guarantees that the participation of any single person in the dataset does not impact the probability of any outcome by much.

Enforcing this guarantee requires the algorithm to be carefully designed and rigorously analyzed. The process of designing and analyzing such algorithms is prone to errors and imperfections as has been noted in the literature Tramer et al. (2022). A result of this is that differentially private mechanisms may not perform as intended, either offering less privacy than expected due to flaws in mathematical analysis or implementation, or potentially providing stronger privacy guarantees that are not evident through a loose analysis.

Empirical privacy auditing (Ding et al., 2018; Nasr et al., 2023; Jagielski et al., 2020) has emerged as a critical tool to bridge this gap. By experimentally assessing the privacy of mechanisms, empirical auditing allows for the verification of privacy parameters. Specifically, an audit procedure is a randomized algorithm that takes an implementation of a mechanism M, runs it in a black-box manner, and attempts to test a privacy hypothesis (such as, a differential privacy parameter). The procedure outputs 0 if there is sufficient evidence that the mechanism does not satisfy the hypothesized guarantees and 1 otherwise. The audit mechanism must possess two essential properties: 1) it must have a *provably* small false-negative rate, ensuring that it would not erroneously reject a truly differentially private mechanism, with high probability; 2) it needs to *empirically* exhibit a "reasonable" false positive rate, meaning that when applied to a non-differentially private mechanism, it would frequently reject the privacy hypothesis. The theoretical proof of the false positive rate is essentially equivalent to privacy accounting (Abadi et al., 2016; Dong et al., 2019; Mironov, 2017), which is generally thought to be impossible in a black-box manner (Zhu et al., 2022).

The prior literature on empirical audits of privacy consists of two lines of work, each with its own set of limitations. The first line of work (Ding et al., 2018; Jagielski et al., 2020; Tramer et al., 2022; Nasr et al., 2023) runs a differentially private algorithm multiple times to determine if the privacy guarantees are violated. This is highly computationally inefficient for most private machine learning use-cases, where running the algorithm a single time involves training a large model.

Recent work (Steinke et al., 2023) remove this limitation by proposing an elegant auditing method that runs a differentially private training algorithm a single time. In particular, they rely on the randomness of training data to obtain bounds on the false negative rates of the audit procedure. A key limitation of the approach in Steinke et al. (2023) is that their audit procedure is sub-optimal in the sense that there is a relatively large gap between the true privacy parameters of mainstream privacy-preserving algorithms (e.g., Gaussian mechanism) and those reported by their auditing algorithm.

In this work, we propose a novel auditing procedure that is computationally efficient and accurate. Our method requires only a single run of the privacy mechanism [1] and leverages the f-DP curve (Dong et al., 2019), which allows for a more fine-grained accounting of privacy than the traditional reliance on ϵ, δ parameters. By doing so, we provide a tighter empirical assessment of privacy.

We experiment with our approach on both simple Gaussian mechanisms as well as a model trained on real data witth DP-SGD. Our experiments show that our auditing procedure can significantly outperform that of Steinke et al. (2023) (see Figure 1). This implies that better analysis may enable relatively tight auditing of differentially privacy guarantees in a computationally efficient manner in the context of large model training.

Technical overview: We briefly summarize the key technical components of our work and compare it with that of Steinke et al. (2023). Their auditing procedure employed a game similar to a membership inference process: the auditor selects a set of canaries and, for each canary, decides whether to inject it into the training set with independent probability 0.5. Once model training is completed, the auditor performs a membership inference attack to determine whether each canary was included. The number of correct guesses made by the adversary in this setting forms a random variable. The key technical contribution of Steinke et al. was to establish a tail bound on this random variable for mechanisms satisfying (ϵ)-DP. Specifically, they demonstrated that the tail of this random variable is bounded by that of a binomial distribution, $\mathbf{binomial}(n, p)$, where n is the number of canaries and $p = \frac{e^\epsilon}{e^\epsilon+1}$. To extend this analysis to approximate DP mechanisms, they further showed that the probability of the adversary's success exceeding this tail bound is at most $O(n \cdot \delta)$.

Steinke et al. highlighted a limitation in their approach in auditing specific mechanisms, such as the Gaussian mechanism. They correctly argue that simplifying the mechanism's behavior to just two parameters, (ϵ, δ) , results in sub-optimal auditing of specific mechanisms. In other words, the effectiveness of membership inference attacks against the Gaussian mechanism differs significantly from predictions based solely on the (ϵ, δ) parameters. To overcome this limitation, we propose auditing the entire privacy curve of a mechanism, rather than focusing solely on (ϵ, δ). Our solution involves three key technical steps:

1. We derive an upper bound on the adversary's success in correctly guessing a specific canary for mechanisms satisfying f-DP. This bound is an improved version of the result by Hayes et al. (2023) for bounding training data reconstruction in DP mechanisms. However, this is insufficient, as the adversary's guesses could be dependent, potentially leading to correlated successes (e.g., correctly or incorrectly guessing all samples).

2. To address the issue of dependency, we refine our analysis by defining p_i as the probability of the adversary making exactly i correct guesses. We derive a recursive relation that bounds p_i based on $p_1, \ldots, p_{i-1}$. This recursive bound is the main technical novelty of our work. To derive this bound, we consider two conditions: the adversary correctly guesses the first canary or not. In the first case, we use our analysis from Step 1 to bound the probability of making $i - 1$ correct guesses given that the first guess was correct. For the incorrect guess case, we perform a combinatorial analysis to eliminate the condition. This analysis uses the fact that shuffling of the canaries does not change the probabilities of making i correct guesses. We note that it is crucial not to use the analysis of Step 1 for both cases. This is because the analysis of Step 1 cannot be tight for both cases at the same time. Finally,

[1] In the context of privacy-preserving training of machine learning models, the privacy mechanism refers to the training algorithm. Therefore, when we mention a single run, we are specifically referring to a single execution of the training algorithm, not the inference algorithm.

leveraging the convexity of trade-off functions and applying Jensen's inequality, we derive our final recursive relation. To the best of our knowledge, This combination of trade-off function with shuffling is a new technique and could have broader applications.

3. Finally, we design an algorithm that takes advantage of the recursive relation to numerically calculate an upper bound on the tail of the distribution. The algorithm is designed carefully so that we do not need to invoke the result of step 2 for very small events.

We also generalize our analysis to a broader notion of canary injection and membership inference. Specifically, we utilize a reconstruction game where the auditor can choose among k options for each canary point, introducing greater entropy for each choice. This generalization allows for auditing mechanisms with fewer canaries.

In the rest of the paper, we first introduce the notions of f-DP and explain what auditing based on f-DP entails. We then present our two auditing procedures, which are based on membership inference and reconstruction attacks (Section 2). In Section 3, we provide a tight analysis of our audit's accuracy based on f-DP curves. Finally, in Section 4, we describe the experimental setup used to compare the bounds.

2 AUDITING f- DIFFERENTIAL PRIVACY

Auditing privacy involves testing a "privacy hypothesis" about an algorithm M. Different mathematical forms can be used for a "privacy hypothesis," but they all share the common characteristic of being about an algorithm/mechanism M. For example, one possible hypothesis is that applying SGD with specific hyperparameters satisfies some notion of privacy. With this in mind, the privacy hypothesis are often mathematical constraints on the sensitivity of the algorithm's output to small changes in its input. The most well-known definition among these is (approximate) differential privacy.

Definition 1. *A mechanism M is (ϵ, δ)-DP if for all neighboring datasets $\mathcal{S}, \mathcal{S}'$ with $|\mathcal{S}\Delta\mathcal{S}'| = 1$ and all measurable sets T, we have $\Pr[M(\mathcal{S}) \in T] \le e^{\epsilon} \Pr[M(\mathcal{S}') \in T] + \delta$.*

In essence, differential privacy ensures that the output distribution of the algorithm does not heavily depend on a single data point. Based on this definition, one can hypothesize that a particular algorithm satisfies differential privacy with certain ϵ and δ parameters. Consequently, auditing differential privacy involves designing a test for this hypothesis. We will later explore the desired properties of such an auditing procedure. However, at present, we recall a stronger notion of privacy known as f-differential privacy.

Notation For a function $f\colon X \to \mathbb{R}$ we use $\bar{f}$ to denote the function $\bar{f}(x) = 1 - f(x)$.

Definition 2. *A mechanism $\mathcal{M}$ is f-DP if for all neighboring datasets $\mathcal{S}, \mathcal{S}'$ and all $|\mathcal{S}\Delta\mathcal{S}'| = 1$ measurable sets T we have*

$$\Pr[M(\mathcal{S}) \in T] \le \bar{f}\big(\Pr[M(\mathcal{S}')] \in T]\big).$$

Note that this definition generalizes the notion of approximate differential privacy by allowing a more complex relation between the probability distributions of $M(S)$ and $M(S')$. The following proposition shows how one can express approximate DP as an instantiation of f-DP.

Proposition 3. *A mechanism is (ϵ, δ)-DP if it is f-DP with respect to $\bar{f}(x) = e^{\epsilon} \cdot x + \delta$.*

Although the function f could be an arbitrary function, without loss of generality, we only consider a specific class of functions in this notion.

Remark 4. *Whenever we say that a mechanism satisfies f-DP, we implicitly imply that f is a valid trade-off function . That is, f is defined on domain $[0, 1]$ and has a range of $[0, 1]$. Moreover, f is a decreasing and convex with $f(x) \le 1 - x$ for all $x \in [0, 1]$. We emphasize that this is without loss of generality. That is, if a mechanism is f-DP for a an arbitrary function $f : [0, 1] \to [0, 1]$, then it is also f'-DP for valid trade-off function f' with $f'(x) \le f(x)$ for all $x \in [0, 1]$ (See Proposition 2.2 in Dong et al. (2019)).*

Definition 5 (Order of f-DP curves)**.** *For two trade-off functions f_1 and f_2, we say f_1 is more private than f_2 and denote it by $f_1 \ge f_2$ iff $f_1(x) \ge f_2(x)$ for all $x \in [0, 1]$. Also, for a family of trade-off* *functions F, we use $maximal(F)$ to denote the set of maximal elements w.r.t to the privacy relation. Note that F could be a partial ordered set, and the set of maximal points could have more than a single element.*

Now that we have defined our privacy hypothesis, we can turn our attention to auditing these notions.

Definition 6 (Auditing f-DP)**.** *An audit procedure takes the description of a mechanism $\mathcal{M}$, a trade-off function f, and outputs a bit that determines whether the mechanism satisfies f-DP or not. We define the audit procedure as a two-step procedure.*

- *$game$: $M \to O$, In this step, the auditor runs a potentially randomized experiment/game using the description of mechanism $\mathcal{M} \in M$ and obtains some observation $o \in O$.*
- *$evaluate : O \times F \to \{0, 1\}$, In this step, the auditor will output a bit b based on an observation o and a trade-off function f. This audit operation tries to infer whether the observation o is "likely" for a mechanism that satisfies f-DP.*

The audit procedure is ψ-accurate if for all mechanism $\mathcal{M}$ that satisfy f-DP, we have

$$\Pr_{o \leftarrow game(\mathcal{M})}[evaluate(o, f) = 1] \geq \psi.$$

Note that we are defining the accuracy only for positive cases. This is the only guarantee we can get from running attacks. For guarantees in negative cases, we need to perform a proper accounting of the mechanism (Wang et al., 2023).

Auditing f-DP vs DP: f-DP can be viewed as a collection of DP parameters, where instead of considering (ϵ, δ) as fixed scalars, we treat ϵ as a function of δ. For any $\delta \in [0, 1]$, there exists an $\epsilon(\delta)$ such that the mechanism satisfies $(\epsilon(\delta), \delta)$-DP. The f-DP curve effectively represents the entire privacy curve rather than a single (ϵ, δ) pair. Thus, auditing f-DP can be expected to be more effective, as there are more constraints that need to be satisfied. A naive approach for auditing f-DP is to perform an audit for approximate DP at each (ϵ, δ) value along the privacy curve, rejecting if any of the audits fail. However, this leads to suboptimal auditing performance. First, the auditing analysis involves several inequalities that bound the probabilities of various events using differential privacy guarantees. The probability of these events could take any number between $[0, 1]$. Using a single (ϵ, δ) value to bound the probability of all these events cannot be tight because the linear approximation of privacy curve is tight in at most a single point. Hence, the guarantees of (ϵ, δ)-DP cannot be simultaneously tight for all events. However, with f-DP, we can obtain tight bounds on the probabilities of all events simultaneously. Second, For each (ϵ, δ) we have a small possibility of incorrectly rejecting the privacy hypothesis. So if we audit privacy for $(\epsilon(\delta), \delta)$ independently, we will reject any privacy hypothesis with probability 1.0. This challenge can be potentially resolved by using correlated randomness, but that requires a new analysis.

Next, we formally define the notion of empirical privacy (Nasr et al., 2021) based on an auditing procedure. This notion essentially provides the best privacy guarantee that is not violated by auditors' observation from a game setup.

Definition 7 (Empirical Privacy)**.** *Let $(game, evaluate)$ be an audit procedure. We define the empirical privacy random variable for a mechanism $\mathcal{M}$, w.r.t a family F of trade-off functions, to be the output of the following process. We first run the game to obtain observation $o = game(\mathcal{M})$. We then construct*

$$F_o = maximal(\{f \in F; evaluate(o, f) = 1\})$$

where the maximal set is constructed according to Definition 5. Then, the empirical privacy of the mechanism at a particular δ is defined as

$$\epsilon(\delta) = \min_{f \in F_o} \max_{x \in [0,1]} \frac{1 - f(x) - \delta}{x}.$$

Note that the empirical privacy $\epsilon(\delta)$ is a function of the observation o. Since, o itself is a random variable, then $\epsilon(\delta)$ is also a random variable.

How to choose the family of trade-off functions? The family of trade-off functions should be chosen based on the expectations of the true privacy curve. For example, if one expects the privacy curve of a mechanism to be similar to that of a Gaussian mechanism, then they would choose the set of all trade-off functions imposed by a Gaussian mechanism as the family. For example, many believe that in the hidden state model of privacy (Ye & Shokri, 2022), the final model would behave like a Gaussian mechanism with higher noise than what is expected from the accounting in the white-box model (where we assume we release all the intermediate models). Although we may not be able to prove this hypothesis , we can use our framework to calculate the empirical privacy, while assuming that the behavior of the final model would be similar to that of a Gaussian mechanism.

2.1 GUESSING GAMES

Here, we introduce the notion of guessing games which is a generalization of membership inference attacks (Nasr et al., 2023), and closely resembles the reconstruction setting introduced in Hayes et al. (2023).

Definition 8. *Consider a mechanism $M : [k]^m \to \Theta$. In a guessing game we first sample an input dataset $\mathbf{u} \in [k]^m$ from an arbitrary distribution. We run the mechanism to get $\theta \sim M(\mathbf{u})$. Then a guessing adversary $A : \Theta \to ([k] \cup \{\perp\})^m$ tries to guess the input to the mechanism from the output. We define*

- *the number of guesses by $c' = \sum_{i=1}^{m} \mathbf{I}\big(A(\theta)_i \neq \perp\big)$*
- *and the number of correct guesses by $c = \sum_{i=1}^{m} \mathbf{I}\big(A(\theta)_i = \mathbf{u}_i\big)$.*

Then we output (c, c') as the output of the game.

These guessing games are integral to our auditing strategies. We outline two specific ways to instantiate the guessing game. The first procedure is identical to that described in the work of Steinke et al. (2023) and resembles membership inference attacks. The second auditing algorithm is based on the reconstruction approach introduced by Hayes et al. (2023). In Section 3, we present all of our results in the context of the general notion of guessing games, ensuring that our findings extend to both the membership inference and reconstruction settings.

Auditing by membership inference: Algorithm 1 describes a game setup based on membership inference attacks. In this setup, we have a fixed training set $\mathcal{T}$ and a set of canaries $\mathcal{C}$. We first sample a subset $\mathcal{S}$ of the canaries using poisson sampling. Then we run the mechanism $\mathcal{M}$ on $\mathcal{T} \cup \mathcal{S}$ to get a model $\theta \sim \mathcal{M}(\mathcal{T} \cup \mathcal{S})$. Then the adversary A inspects θ and tries to find examples that were present in $\mathcal{S}$. Observe that this procedure is a guessing game with $k = 2$ and $m = |\mathcal{C}|$. This is simply because the adversary is guessing between two choices for each canary, it is either included or not included. Note that this procedure is modular, we can use any $\mathcal{T}$ and $\mathcal{C}$ for the training set and canary set. We can also use any attack algorithm A.

We note that membership inference attacks have received a lot of attention recently (Homer et al., 2008; Shokri et al., 2017; Leino & Fredrikson, 2020; Bertran et al., 2024; Hu et al., 2022; Matthew et al., 2023; Duan et al., 2024; Zarifzadeh et al., 2023). These attack had a key difference from our attack setup and that is the fact that there is only a single example that the adversary is trying to make the inference for. Starting from the work of (Shokri et al., 2017), researchers have tried to improve attacks in various settings (Ye et al., 2022; Zarifzadeh et al., 2023). For example, using calibration techniques has been an effective way to improve membership inference attacks (Watson et al., 2021; Carlini et al., 2022). Researchers have also changed their focus from average case performance of the attack to the tails of the distribution and measured the precision at low recall values (Ye et al., 2022; Nasr et al., 2021).

A substantial body of research has also explored the relationship between membership inference attacks and differential privacy (Sablayrolles et al., 2019; Mahloujifar et al., 2022; Balle et al., 2022; Bhowmick et al., 2018; Stock et al., 2022; Balle et al., 2022; Guo et al., 2022; Kaissis et al., 2023; 2024), using this connection to audit differential privacy (Steinke et al., 2024a; Pillutla et al., 2024; Jagielski et al., 2020; Ding et al., 2018; Bichsel et al., 2018; Nasr et al., 2021; 2023; Steinke et al., 2024b; Tramer et al., 2022; Bichsel et al., 2021; Lu et al., 2022; Andrew et al., 2023; Cebere et al., 2024; Chadha et al., 2024). Some studies have investigated empirical methods to prevent membership inference attacks that do not rely on differential privacy (Hyland & Tople, 2019; Jia et al., 2019; Chen & Pattabiraman, 2023; Li et al., 2024; Tang et al., 2022; Nasr et al., 2018). An intriguing avenue for future research is to use the concept of empirical privacy to compare the performance of these empirical methods with provable methods, such as DP-SGD.

Algorithm 1 Membership inference in one run game

Input: Oracle access to a mechanism $\mathcal{M}(\cdot)$, A training dataset $\mathcal{T}$, An indexed canary set $\mathcal{C} = \{x_i; i \in [m]\}$, An attack algorithm A.

1: Set $m = |\mathcal{C}|$
2: Sample $u = (u_1, \ldots, u_m) \sim \text{Bernoulli}(0.5)^m$, a binary vector where $u_i = 1$ with probability 0.5.
3: Let $\mathcal{S} = \{\mathcal{C}[u_i]; u_i = 1\}_{i \in [m]}$, the subset of selected elements in $\mathcal{C}$.
4: Run mechanism M on $\mathcal{T} \cup \mathcal{S}$ to get output θ.
5: Run membership inference attack A on θ to get set of membership predictions $v = (v_1, \ldots, v_m)$ which is supported on $\{0, 1, \perp\}^m$.
6: Count c, the number of correct guesses where $u_i = v_i$ and c' the total number of guesses where $v_i \neq \perp$.
7: **return** (c, c').

Auditing by reconstruction: We also propose an alternative way to perform auditing by reconstruction attacks. This setup starts with a training set $\mathcal{S}_t$, similar to the membership inference setting. Then, we have a family of m canary sets $\{\mathcal{S}_c^i; i \in [m]\}$ where each $\mathcal{S}_c^i$ contains k distinct examples. Before training, we construct a set $\mathcal{S}_s$ of size m by uniformly sampling an example from each $\mathcal{S}_c^i$. Then, the adversary tries to find out which examples were sampled from each canary set $\mathcal{S}_c^i$ by inspecting the model. We recognize that this might be different from what one may consider a true "reconstruction attack", because the adversary is only performing a selection. However, if you consider the set size to be arbitrary large, and the distribution on the set to be arbitrary, then this will be general enough to cover various notions of reconstruction. We also note that Hayes et al. (2023) use the same setup to measure the performance of the reconstruction attacks.

Algorithm 2 Reconstruction in one run game

Input: Oracle access to a mechanism $\mathcal{M}(\cdot)$, A training dataset $\mathcal{T}$, number of canaries m, number of options for each canary k, a matrix of canaries $\mathcal{C} = \{x_j^i\}_{i \in [m], j \in [k]}$, an attack algorthm A.

1: Let $u = (u_1, \ldots, u_m)$ be a vector uniformly sampled from $[k]^m$.
2: Let $\mathcal{S} = \{x_{u_i}^i\}_{i \in [m]}$.
3: Run mechanism $\mathcal{M}$ on $\mathcal{S} \cup \mathcal{T}$ to get output θ.
4: Run a reconstruction attack A on θ to get a vector $v = (v_1, \ldots, v_m)$ which is a vector in $([k] \cup \{\perp\})^m$.
5: Count c the number of coordinates where $u_i = v_i$ and c' the number of coordinates where $v_i \neq \perp$.
6: **return** (c, c').

3 Implications of f-DP for guessing games

In this section, we explore the implications of f-DP for guessing games. Specifically, we focus on bounding the probability of making more than c correct guesses for adversaries that make at most c' guesses. We begin by stating our main theorem, followed by an explanation of how it can be applied to audit the privacy of a mechanism.

Theorem 9. *[Bounds for adversary with bounded guesses] Let $M : [k]^m \to \Theta$ be a f-DP mechanism. Let $\mathbf{u}$ be a random variable uniformly distributed on $[k]^m$. Let $A \colon \Theta \to ([k] \cup \{\perp\})^m$ be a guessing* *adversary which always makes at most c' guesses, that is*

$$\forall \theta \in \Theta, \Pr\left[\Big(\sum_{i=1}^{m} I\big(A(\theta)_i \neq \bot\big)\Big) > c'\right] = 0,$$

and let $\mathbf{v} \equiv A(M(\mathbf{u}))$*. Define* $p_i = \Pr\left[\Big(\sum_{j\in[m]} \mathbf{I}(\mathbf{u}_j = \mathbf{v}_j)\Big) = i\right]$*. For all subset of indices* $T \subseteq [c']$*, we have*

$$\sum_{i\in T} \frac{i}{m} p_i \leq \bar{f}\Big(\frac{1}{k-1} \sum_{i\in T} \frac{c'-i+1}{m} p_{i-1}\Big).$$

This Theorem, which we consider to be our main technical contribution, provides a nice invariant that bounds the probability p_i (probability of making exactly i correct guesses) based on the value of other p_js. Imagine P_f to be a set of vectors $p = (p_1, \dots, p_{c'})$ that could be realized for an attack on a f-DP mechanism. Theorem 9 significantly confines this set. However, this still does not resolve the auditing task. We are interested in bounding $\max_{p\in P_f} \sum_{i=c}^{c'} p_i$, the maximum probability that an adversary can make more than c correct guesses for an f-DP mechanism. Next, we show how we can algorithmically leverage the limitations imposed by Theorem 9 and calculate an upper bound on $\max_{p\in P_f} \sum_{i=c}^{c'} p_i$.

3.1 Numerically bounding the tail

In this subsection, we specify our procedure for bounding the tail of the distribution and hence the accuracy of our auditing procedure. Our algorithm needs oracle access to f and $\bar{f}$ and decides an upper bound on the probability of an adversary making c correct guesses in a guessing game with alphabet size k and a mechanism that satisfies f-DP. This algorithm relies on the confinement imposed by Theorem 9. Note that Algorithm 3 is a decision algorithm, it takes a value τ and decide if the probability of making more than c correct guesses is less than or equal to τ. We can turn this algorithm to a estimation algorithm by performing a binary search on the value of τ. However, for our use cases, we are interested in a fixed τ. This is because we (similar to (Steinke et al., 2023)) want to set the accuracy of our audit to be a fixed value such as 0.95.

Algorithm 3 Numerically deciding an upper bound probability of making more than c correct guesses

Input: Oracle access to $\bar{f}$ and $\bar{f}^{-1}$, number of guesses c', number of correct guesses c, number of samples m, alphabet size k, probability threshold τ (default is $\tau = 0.05$).

$\forall 0 \leq i \leq c$ set $h[i] = 0$, and $r[i] = 0$.
set $r[c] = \tau \cdot \frac{c}{m}$.
set $h[c] = \tau \cdot \frac{c'-c}{m}$.
for $i \in [c-1, \dots, 0]$ **do**
  $h[i] = (k-1)\bar{f}^{-1}\big(r[i+1]\big)$
  $r[i] = r[i+1] + \frac{i}{c'-i} \cdot \big(h[i] - h[i+1]\big)$.
end for
if $r[0] + h[0] \geq \frac{c'}{m}$ **then**
  Return True; (Probability of c correct guesses (out of c') is less than τ).
else
  Return False; (Probability of having c correct guesses (out of c') could be more than τ).
end if

Theorem 10. *If Algorithm 3 returns True on inputs $\bar{f}, k, m, c, c'$ and τ, then for any f-DP mechanism $M : [k]^m \to \Theta$, any guessing adversary $A : \Theta \to ([k] \cup \{\bot\})^m$ with at most c' guesses, defining $\mathbf{u}$ to be uniform over $[k]^m$, and setting $\mathbf{v} \equiv A\big(M(\mathbf{u})\big)$, we have* $\Pr[\big(\sum_{i=1}^{m} \mathbf{I}(\mathbf{u}_i = \mathbf{v}_i)\big) \geq c] \leq \tau$.

In a nutshell, this algorithm tries to obtain an upper bound on the sum $p_c + p_{c+1} + \dots, p_{c'}$. We assume this probability is greater than τ, and we obtain lower bound on $p_{c-1} + p_c + \cdots + p_{c'}$ based on this assumption. We keep doing this recursively until we have a lower bound on $p_0 + \cdots + p_{c'}$. If this lower bound is greater than 1, then we have a contradiction and we return true. The detailed proof of this Theorem is involved and requires careful analysis. We defer the full proof of Theorem to appendix.

Auditing f-DP with Algorithm 3: When auditing the f-DP for a mechanism, we assume we have injected m canaries, and ran an adversary that is allowed to make c' guesses and recorded that the adversary have made c correct guesses. In such scenario, we will reject the hypothesized privacy of the mechanism if the probability of this observation is less than a threshold τ, which we by default set to 0.05. To this end, we just call Algorithm 3 with parameters c, c', m, $\tau = 0.05$ and f. Then if the algorithm returns *True*, we will reject the privacy hypothesis and approve it otherwise.

Empirical privacy: Although auditing in essence is a hypothesis testing, previous work has used auditing algorithms to calculate empirical privacy as defined in definition 7. In this work, we follow the same route. For simplicity, we only consider an ordered set of privacy hypotheses $h_1, \ldots, h_w$ as our family of f-DP curves. These sets are ordered in their strength, meaning that any mechanism that satisfies h_i, would also satisfy h_j for all $j < i$. Then, we would report the strongest privacy hypothesis that passes the test as the empirical privacy of the mechanism.

4 EXPERIMENTS

Most of our experiments are conducted in an *idealized setting*, similar to that used in Steinke et al. (2023), unless otherwise stated. In this setting, the attack success rate is automatically calculated to simulate the expected number of correct guesses by an optimal adversary (Details of the idealized setting are provided in Algorithm 4 in Appendix). We then use this expected number as the default value for the number of correct guesses to derive the empirical ϵ. More specifically, as specified in Definition 6, we instantiate our auditing with a game and evaluation setup. We use Algorithm 4 in Appendix as our game setup. This algorithm returns the number of guesses and the number of correct guesses as the observation from the game. Then, we use Algorithm 3 as our evaluation setup to audit an f-DP curve based on the observation from Algorithm 4. Note that in our comparison with the auditing of Steinke et al., we always use the same membership inference game setup ($k = 2$) as defined in their work. This ensures that our comparison is only on the evaluation part of the audit procedure.

In all experiments, we use empirical ϵ as the primary metric for evaluating our bounds. As described in Section 3.1 , we need an ordered set of f-DP curves to obtain empirical privacy. In our experiments, we use f-DP curves for Gaussian mechanisms with varying standard deviations (this forms an ordered set because the f-DP curve of a Gaussian mechanism with a higher standard deviation dominates that of a lower standard deviation). For sub-sampled Gaussian mechanisms, the ordered set consists of f-DP curves for sub-sampled Gaussian mechanisms with the given sub-sampling rate and number of steps and different noise standard deviations.

4.1 COMPARISON WITH STEINKE ET AL. (2023)

In this section, we evaluate our auditing method for membership inference in an idealized setting, using the work of Steinke et al. (2023) as our main baseline. We compare our approach directly to their work, which operates in the same setting as ours.

Simple Gaussian Mechanism: In the first experiment (Figure 1), we audit a simple Gaussian mechanism, varying the standard deviations from $[0.5, 1.0, 2.0, 4.0]$, resulting in different theoretical ϵ values. We vary the number of canaries (m) from 10^2 to 10^7 for auditing, set the bucket size to $k = 2$, and adjust the number of guesses (c') for each number of canaries. For each combination of m, c', and each standard deviation, we calculate the expected number of correct guesses (c) using Algorithm 4 (the idealized setting). We then audit all tuples of (m, c, c') using the f-DP curves of the Gaussian mechanism, selecting the c that achieves the highest empirical ϵ as the reported empirical ϵ for m canaries at a given standard deviation.

We also apply the same setup for the auditing procedure of Steinke et al. (2023), differing only in the way empirical privacy is calculated. Figure 1 demonstrates that our approach outperforms the empirical privacy results from Steinke et al. Interestingly, while the bound in Steinke et al. (2023) degrades as the number of canaries increases, our bounds continue to improve.

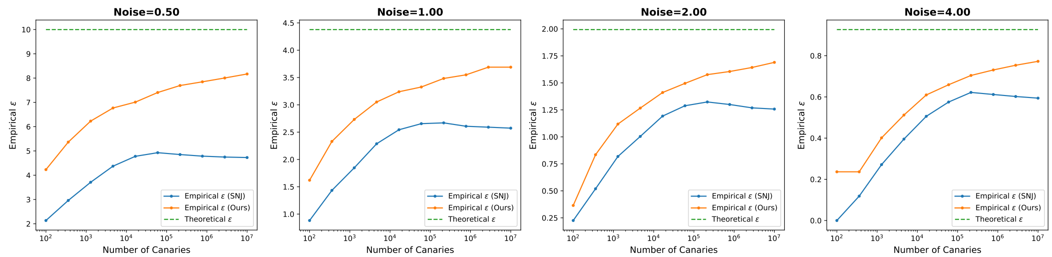

Figure 1: Comparison between our empirical privacy lower bounds and that of Steinke et al. (2023)

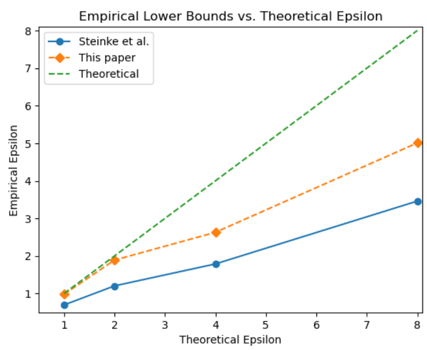

Figure 2: Comparison with auditing procedure of Steinke et al. (2023) on auditing CIFAR-10 in white-box setting using gradient-based membership inference attacks.

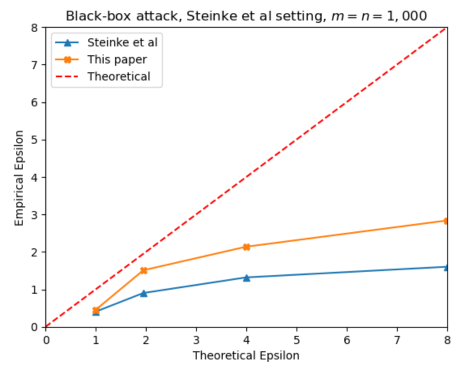

Figure 3: Comparison with auditing procedure of Steinke et al. (2023) on auditing CIFAR-10 in black-box setting.

Experiments on CIFAR-10: We also run experiments on CIFAR-10 on a modified version of the WRN16-4 (Zagoruyko & Komodakis, 2016) architecture, which substitutes batch normalization with group normalization. We follow the setting proposed by Sander et al. (2023), which use custom augmentation multiplicity (i.e., random crop around the center with 20 pixels padding with reflect, random horizontal flip and jitter) and apply an exponential moving average of the model weights with a decay parameter of 0.9999. We run white-box membership inference attacks by following the strongest attack used in the work of Steinke et al. (2023), where the auditor injects multiple canaries in the training set with crafted gradients. More precisely, each canary gradient is set to zero except at a single random index ("Dirac canary" Nasr et al. (2023)). Note that in the white-box attack, the auditor has access to all intermediate iterations of DP-SGD. The attack scores are computed as the dot product between the gradient update during consecutive model iterates and the aggregated gradients from dp-sgd. As done in the work of Steinke et al. (2023), we audit CIFAR-10 model with $m = 5,000$ canaries and all training points from CIFAR-10 $n = 50,000$ for the attack. We set the batch size to $4,096$, use augumented multiplicity of $K = 16$ and train for $2,500$ DP-SGD steps. For $\varepsilon = 8.0, \delta = 10^{-5}$, we achieved 77% accuracy when auditing, compared to 80% without injected canaries. Figure 2 shows the comparison between the auditing scheme by Steinke et al. (2023) with ours for different values of theoretical ε. We are able to achieve tighter empirical lower bounds. We also report the performance of the black-box attack, where the auditor does not control the training pipeline and can only compute memberships scores (losses) from the final model. Figure 3 shows how we are able to achieve tighter lower bounds compared to Steinke et al. (2023) where we set $m = 1,000$ and all training samples are used for auditing ($m = n$). This corresponds to the stronger setup for the black-box auditor in Steinke et al. (2023).

Finally, we report the results of auditing the robust membership inference attack Zarifzadeh et al. (2023) (RMIA), which to the best of our knowledge represents the State-of-The-Art (SoTA) black-box membership inference attack on CIFAR-10 from the literature. We reproduce the results in Zarifzadeh et al. (2023) with a non-private WideResNet model (with depth 28 and width 2) for 100 training epochs on half of the dataset chosen at random resulting on a test accuracy of 92.2%. We run the low-cost black-box membership inference attack using 2 reference models in the offline setting Zarifzadeh et al. (2023). We audit with $m = 5,000$ canaries and report in Figure 4 the comparison between our scheme and Steinke et al. (2023) with different abstention values. Our auditing method clearly outperforms Steinke et al. for all bounded guesses settings, with higher empirical epsilon for larger abstentions values (i.e., smaller number of guesses).

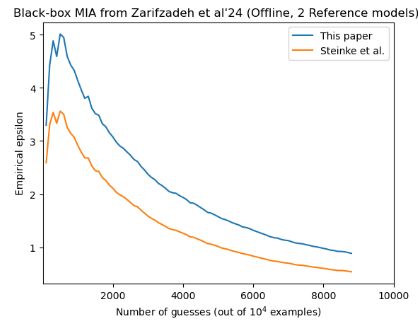

Figure 4: Comparison with auditing procedure of Steinke et al. (2023) on non-private model trained on CIFAR-10 against black-box RMIA method Zarifzadeh et al. (2023).

Why is our bound better better than Steinke et al. (2023)? The bounds in Steinke et al. audit approximate DP. That is, they take DP parameters (ϵ, δ) and prove an upper bound on the probability of any adversary obtaining c' correct guesses out of c total guesses, given m canaries available. For the case of $\delta = 0$, their bound is tight. For the case of $\delta > 0$, however, they need to define a set of undesirable events and bound their collective probability. This incurs an additional $O(m \cdot \delta)$ in the probability. The reason why their bounds start to degrade when we increase m is this very fact. The $m \cdot \delta$ term starts to dominate and causes the empirical epsilon estimation to become worse. The reason we do not observe this behavior is that we do not use (ϵ, δ) to approximate the privacy curve, we use the exact curve as is. As we know, the linear approximation of privacy curve is optimal only in a single point for mechanisms that we are interested in (e.g. the Gaussian mechanism). Namely, there is only a single probability $p' \in [0, 1]$ where we have

$$p = \Pr[M(D) \in E] \quad \text{and} \quad e^{\epsilon} \cdot p + \delta = \Pr[M(D') \in E].$$

Our bound is designed to avoid this issue. We derive a bound that uses the exact f-DP curve, which ensures that for all probabilities $p \in [0, 1]$ the upper bound on the blow-up of events of size p is tight. Moreover, the way we invoke our Theorem 9 in our numerical estimation 3 is designed to apply the bound on events that can be simultaneously tight. This way, our bound does not have the problem of getting worse as the number of samples increases.

Note that this does not mean that there is no way to improve our bound. We still see some gap between the empirical epsilon and the true epsilon. The reason for this, we believe, is in the way numerical tail bound in Algorithm 10 is designed. In this algorithm, we make some relaxations that can be a source of sub-optimality. Specifically, our analysis benefits from the fact that the expectation of correct guesses, conditioned on the correct guesses being greater than c divided by the expectation incorrect guesses conditioned on the same event is greater than c/c'. This step is not tight as we cannot have a mechanism where the adversary makes exactly c correct guesses with probability greater than 0, while making more than c correct guesses with probability exactly 0. For a more interested reader, Equations 6 and 7 in the proof of Theorem 10 is a source of sub-optimality that future work can resolve.

5 Conclusions and Limitations

We introduce a new approach for auditing the privacy of algorithms in a single run using f-DP curves. This method enables more accurate approximations of the true privacy guarantees, addressing the risk of a "false sense of privacy" that may arise from previous approximation techniques. By leveraging the entire f-DP curve, rather than relying solely on point estimates, our approach provides a more nuanced understanding of privacy trade-offs. This allows practitioners to make more informed decisions regarding privacy-utility trade-offs in real-world applications. However, our approach does not provide a strict upper bound on privacy guarantees but instead offers an estimate of the privacy parameters that can be expected in practical scenarios. We also recognize that, despite the improvements over prior work, we still observe a gap between the empirical and theoretical privacy reported in the "one run" setting. Future work could focus on closing this gap to further enhance the reliability of empirical privacy estimations.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Galen Andrew, Peter Kairouz, Sewoong Oh, Alina Oprea, H Brendan McMahan, and Vinith Suriyakumar. One-shot empirical privacy estimation for federated learning. *arXiv preprint arXiv:2302.03098*, 2023.

Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1138–1156. IEEE, 2022.

Martin Bertran, Shuai Tang, Aaron Roth, Michael Kearns, Jamie H Morgenstern, and Steven Z Wu. Scalable membership inference attacks via quantile regression. *Advances in Neural Information Processing Systems*, 36, 2024.

Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

Benjamin Bichsel, Timon Gehr, Dana Drachsler-Cohen, Petar Tsankov, and Martin Vechev. Dp-finder: Finding differential privacy violations by sampling and optimization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 508–524, 2018.

Benjamin Bichsel, Samuel Steffen, Ilija Bogunovic, and Martin Vechev. Dp-sniper: Black-box discovery of differential privacy violations using classifiers. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 391–409. IEEE, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914. IEEE, 2022.

Tudor Cebere, Aurélien Bellet, and Nicolas Papernot. Tighter privacy auditing of dp-sgd in the hidden state threat model. *arXiv preprint arXiv:2405.14457*, 2024.

Karan Chadha, Matthew Jagielski, Nicolas Papernot, Christopher Choquette-Choo, and Milad Nasr. Auditing private prediction. *arXiv preprint arXiv:2402.09403*, 2024.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

Zitao Chen and Karthik Pattabiraman. Overconfidence is a dangerous thing: Mitigating membership inference attacks by enforcing less confident prediction. *arXiv preprint arXiv:2307.01610*, 2023.

Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 475–489, 2018.

Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841*, 2024.

Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pp. 1–12. Springer, 2006.

Chuan Guo, Brian Karrer, Kamalika Chaudhuri, and Laurens van der Maaten. Bounding training data reconstruction in private (deep) learning. In *International Conference on Machine Learning*, pp. 8056–8071. PMLR, 2022.

Jamie Hayes, Saeed Mahloujifar, and Borja Balle. Bounding training data reconstruction in dp-sgd. *arXiv preprint arXiv:2302.07225*, 2023.

Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS genetics*, 4(8):e1000167, 2008.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s): 1–37, 2022.

Stephanie L Hyland and Shruti Tople. On the intrinsic privacy of stochastic gradient descent. *Preprint at https://arxiv. org/pdf/1912.02919. pdf*, 2019.

Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33: 22205–22216, 2020.

Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 259–274, 2019.

Georgios Kaissis, Jamie Hayes, Alexander Ziller, and Daniel Rueckert. Bounding data reconstruction attacks with the hypothesis testing interpretation of differential privacy. *arXiv preprint arXiv:2307.03928*, 2023.

Georgios Kaissis, Alexander Ziller, Stefan Kolek, Anneliese Riess, and Daniel Rueckert. Optimal privacy guarantees for a relaxed threat model: Addressing sub-optimal adversaries in differentially private machine learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In *29th USENIX security symposium (USENIX Security 20)*, pp. 1605–1622, 2020.

Jiacheng Li, Ninghui Li, and Bruno Ribeiro. {MIST}: Defending against membership inference attacks through {Membership-Invariant} subspace training. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 2387–2404, 2024.

Fred Lu, Joseph Munoz, Maya Fuchs, Tyler LeBlond, Elliott Zaresky-Williams, Edward Raff, Francis Ferraro, and Brian Testa. A general framework for auditing differentially private machine learning. *Advances in Neural Information Processing Systems*, 35:4165–4176, 2022.

Saeed Mahloujifar, Alexandre Sablayrolles, Graham Cormode, and Somesh Jha. Optimal membership inference bounds for adaptive composition of sampled gaussian mechanisms. *arXiv preprint arXiv:2204.06106*, 2022.

Jagielski Matthew, Nasr Milad, Choquette-Choo Christopher, Lee Katherine, and Carlini Nicholas. Students parrot their teachers: Membership inference on model distillation. *arXiv preprint arXiv: 2303.03446*, 2023.

Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pp. 263–275. IEEE, 2017.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pp. 634–646, 2018.

Milad Nasr, Shuang Songi, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*, pp. 866–882. IEEE, 2021.

Milad Nasr, Jamie Hayes, Thomas Steinke, Borja Balle, Florian Tramèr, Matthew Jagielski, Nicholas Carlini, and Andreas Terzis. Tight auditing of differentially private machine learning. *arXiv preprint arXiv:2302.07956*, 2023.

Krishna Pillutla, Galen Andrew, Peter Kairouz, H Brendan McMahan, Alina Oprea, and Sewoong Oh. Unleashing the power of randomization in auditing differentially private ml. *Advances in Neural Information Processing Systems*, 2024.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pp. 5558–5567. PMLR, 2019.

Tom Sander, Pierre Stock, and Alexandre Sablayrolles. Tan without a burn: Scaling laws of dp-sgd. In *International Conference on Machine Learning*. PMLR, 2023.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

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

Thomas Steinke, Milad Nasr, Arun Ganesh, Borja Balle, Christopher A Choquette-Choo, Matthew Jagielski, Jamie Hayes, Abhradeep Guha Thakurta, Adam Smith, and Andreas Terzis. The last iterate advantage: Empirical auditing and principled heuristic analysis of differentially private sgd. *arXiv preprint arXiv:2410.06186*, 2024a.

Thomas Steinke, Milad Nasr, and Matthew Jagielski. Privacy auditing with one (1) training run. *Advances in Neural Information Processing Systems*, 36, 2024b.

Pierre Stock, Igor Shilov, Ilya Mironov, and Alexandre Sablayrolles. Defending against reconstruction attacks with r\'enyi differential privacy. *arXiv preprint arXiv:2202.07623*, 2022.

Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. Mitigating membership inference attacks by {Self-Distillation} through a novel ensemble architecture. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 1433–1450, 2022.

Florian Tramer, Andreas Terzis, Thomas Steinke, Shuang Song, Matthew Jagielski, and Nicholas Carlini. Debugging differential privacy: A case study for privacy auditing. *arXiv preprint arXiv:2202.12219*, 2022.

Jiachen T Wang, Saeed Mahloujifar, Tong Wu, Ruoxi Jia, and Prateek Mittal. A randomized approach for tight privacy accounting. *arXiv preprint arXiv:2304.07927*, 2023.

Lauren Watson, Chuan Guo, Graham Cormode, and Alex Sablayrolles. On the importance of difficulty calibration in membership inference attacks. *arXiv preprint arXiv:2111.08440*, 2021.

Jiayuan Ye and Reza Shokri. Differentially private learning needs hidden state (or much faster convergence). *Advances in Neural Information Processing Systems*, 35:703–715, 2022.

Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3093–3106, 2022.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Sajjad Zarifzadeh, Philippe Cheng-Jie Marc Liu, and Reza Shokri. Low-cost high-power membership inference by boosting relativity. 2023.

Yuqing Zhu, Jinshuo Dong, and Yu-Xiang Wang. Optimal accounting of differential privacy via characteristic function. In *International Conference on Artificial Intelligence and Statistics*, pp. 4782–4817. PMLR, 2022.

A Proofs

A.1 Proof outline for Theorem 9

In this subsection, we outline the main ingredients we need to prove our Theorem 9. We also provide the full proof for a simplified version of Theorem 9 using these ingredients. First, we have a Lemma that bounds the probability of any event conditioned on correctly guessing a single canary.

Lemma 11. *Let $M : [k]^m \to \Theta$ be a mechanism that satisfies f-DP. Also let $A\colon \Theta \to ([k] \cup \{\perp\})^m$ be a guessing attack. Let $\mathbf{u}$ be a random variable uniformly distributed over $[k]^m$ and let $\mathbf{v} \equiv A\big(M(\mathbf{u})\big)$. Then for any subset $E \subseteq \Theta$ we have*

$$f''_k\Big(\Pr\big[M(\mathbf{u}) \in E\big]\Big) \le \Pr\big[M(\mathbf{u}) \in E \text{ and } u_1 = v_1\big] \le f'_k\Big(\Pr\big[M(\mathbf{u}) \in E\big]\Big)$$

where

$$f'_k(x) = \sup\{\alpha; \alpha + f(\frac{x-\alpha}{k-1}) \le 1\} \text{ and } f''_k(x) = \inf\{\alpha; (k-1)f(\alpha) + x - \alpha) \le 1\}.$$

This Lemma which is a generalization and an improvement over the main Theorem of (Hayes et al., 2023), shows that the probability of an event cannot change too much if we condition on the success of adversary on one of the canaries. Note that this Lemma immediately implies a bound on the expected number of correct guesses by any guessing adversary (by just using linearity of expectation). However, here we are not interested in expectations. Rather, we need to derive tail bounds. The proof of Theorem 9 relies on some key properties of the f' and f'' functions defined in the statement of Lemma 11. These properties are specified in the following Proposition and proved in the Appendix.

Proposition 12. *The functions f'_k as defined in Lemma 11 is increasing and concave. The function f''_k as defined in Lemma 11 is increasing and convex.*

Now, we are ready to outline the proof of a simplified variant of our Theorem 9 for adversaries that make a guess on all canaries. This makes the proof much simpler and enables us to focus more on the key steps in the proof.

Theorem 13 (Special case of 9)**.** *Let $M : [k]^m \to \Theta$ be a f-DP mechanism. Let $\mathbf{u}$ be a random variable uniformly distributed on $[k]^m$. Let $A\colon \Theta \to [k]^m$ be a guessing adversary and let $\mathbf{v} \equiv A(M(\mathbf{u}))$. Define $p_i = \Pr\Big[\big(\sum_{j\in[m]} \mathbf{I}\big(\mathbf{u}_j = \mathbf{v}_j\big)\big) = i\Big]$. For all subset of indices $T \subseteq [m]$, we have*

$$\sum_{i\in T} \frac{i}{m} p_i \le \bar{f}\big(\frac{1}{k-1} \sum_{i\in T} \frac{m-i+1}{m} p_{i-1}\big)$$

Proof. Let us define a random variable $\mathbf{t} = (\mathbf{t}_1, \ldots, \mathbf{t}_m)$ which is defined as $\mathbf{t}_i = \mathbf{I}(\mathbf{u}_i = \mathbf{v_i})$ We have

$$p_c = \Pr[\sum_{i=1}^{m} \mathbf{t}_i = c] = \Pr[\sum_{i=2}^{m} \mathbf{t}_i = c-1 \text{ and } \mathbf{t}_1 = 1] + \Pr[\sum_{i=2}^{m} \mathbf{t}_i = c \text{ and } \mathbf{t}_1 = 0]$$

Now by Lemma 11 we have $\Pr[\sum_{i=2}^{m} \mathbf{t}_i = c-1 \text{ and } \mathbf{t}_1 = 1] \le f'_k(\sum_{i=2}^{m} \mathbf{t}_i = c-1)$. This is a nice invariant that we can use but $\sum_{i=2}^{m} \mathbf{t}_i = c-1$ could be really small depending on how large m is. To strengthen the bound we sum all p_c's for $c \in T$, and then apply the lemma on the aggregate. That is

$$\begin{aligned}\sum_{j\in T} p_j = \sum_{j\in T} \Pr[\sum_{i=1}^{m} \mathbf{t}_i = j] &= \sum_{j\in T} \Pr[\sum_{i=2}^{m} \mathbf{t}_i = j \text{ and } \mathbf{t}_1 = 0] + \sum_{j\in T} \Pr[\sum_{i=2}^{m} \mathbf{t}_i = j-1 \text{ and } \mathbf{t}_1 = 1] \\ &= \Pr[\sum_{i=2}^{m} \mathbf{t}_i \in T \text{ and } \mathbf{t}_1 = 0] + \Pr[1 + \sum_{i=2}^{m} \mathbf{t}_i \in T \text{ and } \mathbf{t}_1 = 1]\end{aligned}$$

Now we only use the inequality from Lemma 11 for the second quantity above. Using the inequality for both probabilities is not ideal because they cannot be tight at the same time. So we have,

$$\sum_{j\in T} p_j \leq \Pr[\sum_{i=2}^{m} \in T \text{ and } \mathbf{t}_1 = 0] + f'_k(\Pr[1 + \sum_{i=2}^{m} \mathbf{t}_i \in T]).$$

Now we use a trick to make this cleaner. We use the fact that this inequality is invariant to the order of indices. So we can permute $\mathbf{t_i}$'s and the inequality still holds. We have,

$$\begin{aligned}\sum_{j\in T} p_j &\leq \mathop{\mathrm{E}}_{\pi\sim\Pi[m]}[\Pr[\sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T \text{ and } \mathbf{t}_{\pi(1)} = 0]] + \mathop{\mathrm{E}}_{\pi\sim\Pi[m]}[f'_k(\Pr[1 + \sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T])] \\ &\leq \mathop{\mathrm{E}}_{\pi\sim\Pi[m]}[\Pr[\sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T \text{ and } \mathbf{t}_{\pi(1)} = 0]] + f'_k(\mathop{\mathrm{E}}_{\pi\sim\Pi[m]}[\Pr[1 + \sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T]]).\end{aligned}$$

Now we perform a double counting argument. Note that when we permute the order $\sum_{i=2}^{m} \mathbf{t}_{\pi(i)} = j$ and $\mathbf{t}_{\pi(1)} = 0$ counts each instance $t_1, \ldots, t_m$ with exactly j non-zero locations, for exactly $(m-j) \times (m-1)!$ times. Therefore, we have

$$\mathop{\mathrm{E}}_{\pi\sim\Pi[m]}[\Pr[\sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T \text{ and } \mathbf{t}_{\pi(1)} = 0]] = \sum_{j\in T} \frac{m-j}{m} p_j.$$

With a similar argument we have,

$$\mathop{\mathrm{E}}_{\pi\sim\Pi[m]}[\Pr[1 + \sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T]] = \sum_{j\in T} \frac{m-j+1}{m} p_{j-1} + \frac{j}{m} p_j.$$

Then, we have

$$\sum_{j\in T} p_j \leq \sum_{j\in T} \frac{m-j}{m} p_j + f'_k(\sum_{j\in T} \frac{j}{m} p_j + \frac{m-j+1}{m} p_{j-1}).$$

And this implies

$$\sum_{j\in T} \frac{j}{m} p_j \leq f'_k(\sum_{j\in T} \frac{j}{m} p_j + \frac{m-j+1}{m} p_{j-1}).$$

And this, by definition of f'_k implies

$$\sum_{j\in T} \frac{j}{m} p_j \leq \bar{f}(\frac{1}{k-1} \sum_{j\in T} \frac{m-j+1}{m} p_{j-1}).$$

□

A.2 Full Proofs

Proof of Lemma 11. Let $p = \Pr[M(\mathbf{u}) \in E \text{ and } u_1 = v_1]$ and $q = \Pr[M(\mathbf{u}) \in E]$. We have

$$
\begin{aligned}
p &= \sum_{i\in[k]} \Pr[M(\mathbf{u}) \in E \text{ and } u_1 = v_1 = i] \\
&= \frac{1}{k}\sum_{i\in[k]} \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = i] \\
&= \frac{1}{k}\sum_{i\in[k]} \frac{1}{k-1}\Big(\sum_{j\in[k]\setminus\{i\}} \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = i]\Big) \\
\text{(By definition of } f\text{-DP)} \quad &\le \frac{1}{k}\sum_{i\in[k]} \frac{1}{k-1}\Big(\sum_{j\in[k]\setminus\{i\}} 1 - f\big(\Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = j]\big)\Big) \\
\text{(By convexity of } f\text{)} \quad &\le 1 - f\left(\frac{1}{k}\sum_{i\in[k]} \frac{1}{k-1}\Big(\sum_{j\in[k]\setminus\{i\}} \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = j]\Big)\right) \\
&= 1 - f\left(\frac{1}{k-1}\sum_{i\in[k]}\Big(\sum_{j\in[k]\setminus\{i\}} \frac{1}{k}\Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = j]\Big)\right) \\
&= 1 - f\left(\frac{1}{k-1}\sum_{i\in[k]}\Big(\sum_{j\in[k]\setminus\{i\}} \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \text{ and } u_1 = j]\Big)\right) \\
&= 1 - f\Big(\frac{1}{k-1}\Pr[M(\mathbf{u}) \in E \text{ and } u_1 \neq v_1]\Big) \\
&= 1 - f\Big(\frac{q-p}{k-1}\Big).
\end{aligned}
$$

Similarly we have,

$$
\begin{aligned}
p &= \sum_{i\in[k]} \Pr[M(\mathbf{u}) \in E \text{ and } u_1 = v_1 = i] \\
&= \frac{1}{k}\sum_{i\in[k]} \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = i] \\
&= \frac{1}{k}\sum_{i\in[k]} \frac{1}{k-1}\Big(\sum_{j\in[k]\setminus\{i\}} \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = i]\Big) \\
\text{(By definition of } f\text{-DP)} \quad &\ge \frac{1}{k}\sum_{i\in[k]} \frac{1}{k-1}\Big(\sum_{j\in[k]\setminus\{i\}} f^{-1}\big(1 - \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = j]\big)\Big) \\
\text{(By convexity of } f\text{)} \quad &\ge f^{-1}\left(\frac{1}{k}\sum_{i\in[k]} \frac{1}{k-1}\Big(\sum_{j\in[k]\setminus\{i\}} 1 - \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = j]\Big)\right) \\
&= f^{-1}\left(\frac{1}{k-1}\sum_{i\in[k]}\Big(\sum_{j\in[k]\setminus\{i\}} \frac{1}{k}(1 - \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \mid u_1 = j])\Big)\right) \\
&= f^{-1}\left(\frac{1}{k-1}\sum_{i\in[k]}\Big(\sum_{j\in[k]\setminus\{i\}} \Pr[M(\mathbf{u}) \in E \text{ and } v_1 = i \text{ and } u_1 = j]\Big)\right) \\
&= f^{-1}\Big(\frac{1}{k-1}(1 - \Pr[M(\mathbf{u}) \in E \text{ and } u_1 \neq v_1])\Big) \\
&= f^{-1}\Big(\frac{1-q+p}{k-1}\Big).
\end{aligned}
$$

This implies that,

$$f(p) \cdot (k-1) + q - p \leq 1$$

□

Proof of Proposition 12. The function is increasing simply because f is decreasing. We now prove concavity. Let $\alpha_1 = f_k(x_1)$ and $\alpha_2 = f_k(x_2)$. By definition of f_k we have

$$\alpha_1 + f(\frac{x_1 - \alpha_1}{k-1}) \leq 1$$

and

$$\alpha_2 + f(\frac{x_2 - \alpha_2}{k-1}) \leq 1.$$

Averaging these two we get,

$$\frac{\alpha_1 + \alpha_2}{2} + \frac{f(\frac{x_1 - \alpha_1}{k-1}) + f(\frac{x_2 - \alpha_2}{k-1})}{2} \leq 1$$

By convexity of f we have

$$\frac{\alpha_1 + \alpha_2}{2} + f(\frac{\frac{x_1+x_2}{2} - \frac{\alpha_1+\alpha_2}{2}}{k-1}) \leq 1$$

Therefore, by definition of f'_k, we have $f'_k(\frac{x_1+x_2}{2}) \geq \frac{\alpha_1+\alpha_2}{2}$. Similarly, f''_k in increasing just because f is decreasing. And assuming $\alpha_1 = f_k(x_1)$ and $\alpha_2 = f_k(x_2)$ we have

$$f''_k(\frac{x_1 + x_2}{2}) \leq \frac{\alpha_1 + \alpha_2}{2}$$

which implies f''_k is convex. □

Proof of Theorem 9. Instead of working with an adversary with c' guesses, we assume we have an adversary that makes a guess on all m inputs, however, it also submits a vector $\mathbf{q} \in \{0,1\}^m$, with exactly c' 1s and $m - c'$ 0s. So the output of this adversary is a vector $\mathbf{v} \in [k]^m$ and a vector $\mathbf{q} \in \{0,1\}^m$. Then, only correct guesses that are in locations that $\mathbf{q}$ is non-zero is counted. That is, if we define a random variable $\mathbf{t} = (\mathbf{t}_1, \ldots, \mathbf{t}_m)$ as $\mathbf{t}_i = \mathbf{I}(\mathbf{u}_i = \mathbf{v_i})$ then we have

$$\begin{aligned} p_c &= \Pr[\sum_{i=1}^{m} \mathbf{t}_i \cdot \mathbf{q}_i = c] \\ &= \Pr[\sum_{i=2}^{m} \mathbf{t}_i = c - 1 \text{ and } \mathbf{t}_1 = 1 \text{ and } \mathbf{q}_1 = 1] + \Pr[\sum_{i=2}^{m} \mathbf{t}_i = c \text{ and } \mathbf{t}_1 \cdot \mathbf{q}_1 = 0] \end{aligned}$$

Now by Lemma 11 we have

$$\Pr[\sum_{i=2}^{m} \mathbf{t}_i = c - 1 \text{ and } \mathbf{t}_1 = 1 \text{ and } \mathbf{q}_1 = 1] \leq f'_k(\sum_{i=2}^{m} \mathbf{t}_i = c - 1 \text{ and } \mathbf{q}_1 = 1).$$

This is a nice invariant that we can use but $\sum_{i=2}^{m} \mathbf{t}_i = c - 1$ could be really small depending on how large m is. To strengthen the bound we sum all p_c's for $c \in T$, and then apply the lemma on the aggregate. That is

$$\begin{aligned} \sum_{j \in T} p_j &= \sum_{j \in T} \Pr[\sum_{i=1}^{m} \mathbf{t}_i = j] \\ &= \sum_{j \in T} \Pr[\sum_{i=2}^{m} \mathbf{t}_i = j \text{ and } \mathbf{t}_1 \cdot \mathbf{q}_1 = 0] + \sum_{j \in T} \Pr[\sum_{i=2}^{m} \mathbf{t}_i = j - 1 \text{ and } \mathbf{t}_1 = 1 \text{ and } \mathbf{q}_1 = 1] \\ &= \Pr[\sum_{i=2}^{m} \mathbf{t}_i \in T \text{ and } \mathbf{t}_1 \cdot \mathbf{q}_1 = 0] + \Pr[1 + \sum_{i=2}^{m} \mathbf{t}_i \in T \text{ and } \mathbf{t}_1 = 1 \text{ and } \mathbf{q}_1 = 1] \end{aligned}$$

Now we only use the inequality from Lemma 11 for the second quantity above. Using the inequality for both probabilities is not ideal because they cannot be tight at the same time. So we have,

$$\sum_{j \in T} p_j \leq \Pr[\sum_{i=2}^{m} \in T \text{ and } \mathbf{t}_1 \cdot \mathbf{q}_1 = 0] + f'_k(\Pr[1 + \sum_{i=2}^{m} \mathbf{t}_i \in T \text{ and } \mathbf{q}_1 = 1]).$$

Now we use a trick to make this cleaner. We use the fact that this inequality is invariant to the order of indices. So we can permute $\mathbf{t_i}$'s and the inequality still holds. We have,

$$\begin{aligned}\sum_{j \in T} p_j &\leq \mathop{\mathrm{E}}_{\pi \sim \Pi[m]}[\Pr[\sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T \text{ and } \mathbf{t}_{\pi(1)} \cdot \mathbf{q}_{\pi(1)} = 0]] + \mathop{\mathrm{E}}_{\pi \sim \Pi[m]}[f'_k(\Pr[1 + \sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T])] \\ &\leq \mathop{\mathrm{E}}_{\pi \sim \Pi[m]}[\Pr[\sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T \text{ and } \mathbf{t}_{\pi(1)} = 0]] + f'_k(\mathop{\mathrm{E}}_{\pi \sim \Pi[m]}[\Pr[1 + \sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \in T \text{ and } \mathbf{q}_{\pi(1)} = 1]]).\end{aligned}$$

Now we perform a double counting argument. Note that when we permute the order $\sum_{i=2}^{m} \mathbf{t}_{\pi(i)} = j$ and $\mathbf{t}_{\pi(1)} = 0$ counts each instance $t_1, \ldots, t_m$ with exactly j non-zero locations, for exactly $(m-j) \times (m-1)!$ times. Therefore, we have

$$\mathop{\mathrm{E}}_{\pi \sim \Pi[m]}[\Pr[\sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \cdot \mathbf{q}_{\pi(i)} \in T \text{ and } \mathbf{t}_{\pi(1)} \cdot \mathbf{q}_{\pi(i)} = 0]] = \sum_{j \in T} \frac{m-j}{m} p_j.$$

With a similar argument we have,

$$\mathop{\mathrm{E}}_{\pi \sim \Pi[m]}[\Pr[1 + \sum_{i=2}^{m} \mathbf{t}_{\pi(i)} \cdot \mathbf{q}_{\pi(i)} \in T \text{ and } \mathbf{q}_{\pi(1)} = 1]] = \sum_{j \in T} \frac{c' - j + 1}{m} p_{j-1} + \frac{j}{m} p_j.$$

Then, we have

$$\begin{aligned}\sum_{j \in T} p_j &\leq \sum_{j \in T} \frac{m-j}{m} p_j + f'_k(\sum_{j \in T} \frac{j}{m} p_j + \frac{c' - j + 1}{m} p_{j-1}) \\ &= \sum_{j \in T} \frac{m-j}{m} p_j + f'_k(\sum_{j \in T} \frac{j}{m} p_j + \frac{c' - j + 1}{m} p_{j-1}).\end{aligned}$$

And this implies

$$\sum_{j \in T} \frac{j}{m} p_j \leq f'_k(\sum_{j \in T} \frac{j}{m} p_j + \frac{c' - j + 1}{m} p_{j-1}).$$

And this, by definition of f'_k implies

$$\sum_{j \in T} \frac{j}{m} p_j \leq \bar{f}(\frac{1}{k-1} \sum_{j \in T} \frac{c' - j + 1}{m} p_{j-1}).$$

□

Proof of Theorem 10. To prove Theorem 10, we first state and prove a lemma which is consequence of Theorem 9.

Lemma 14. *For all $c \leq c' \in [m]$ let us define*

$$\alpha_c = \sum_{i=c}^{c'} \frac{i}{m} p_i \quad \textit{and} \quad \beta_c = \sum_{i=c}^{c'} \frac{c' - i}{m} p_i$$

We also define a family of functions $r = \{r_{i,j} : [0,1] \times [0,1] \to [0,1]\}_{i \leq j \in [m]}$ and $h = \{h_{i,j} : [0,1] \to [0,1]\}$ that are defined recursively as follows.

$\forall i \in [m] : r_{i,i}(\alpha, \beta) = \alpha$ and $h_{i,i}(\alpha, \beta) = \beta$ and for all $i < j$ we have

$$h_{i,j}(\alpha, \beta) = (k-1)\bar{f}^{-1}\Big(r_{i+1,j}(\alpha, \beta)\Big)$$

$$r_{i,j}(\alpha, \beta) = r_{i+1,j}(\alpha, \beta) + \frac{i}{c' - i}(h_{i,j}(\alpha, \beta) - h_{i+1,j}(\alpha, \beta))$$

Then for all $i \le j$ we have

$$\alpha_i \ge r_{i,j}(\alpha_j, \beta_j) \quad \textit{and} \quad \beta_i \ge h_{i,j}(\alpha_j, \beta_j)$$

Moreover, for $i < j$, $r_{i,j}$ and $h_{i,j}$ are increasing with respect to their first argument and decreasing with respect to their second argument.

Proof of Lemma 14. We prove this by induction on $j - i$. For $j - i = 0$, the statement is trivially correct. We have

$$h_{i,j}(\alpha_j, \beta_j) = (k-1)\bar{f}^{-1}(r_{i+1,j}(\alpha_j, \beta_j)).$$

By induction hypothesis, we have $r_{i+1,j}(\alpha_j, \beta_j) \le \alpha_{i+1}$. Therefore we have

$$h_{i,j}(\alpha_j, \beta_j) \le (k-1)\bar{f}^{-1}(\alpha_{i+1}). \tag{1}$$

Now by invoking Theorem 9, we have

$$\alpha_{i+1} \le \bar{f}(\frac{\beta_i}{k-1}).$$

Now since $\bar{f}$ is increasing, this implies

$$(k-1)\bar{f}^{-1}(\alpha_{i+1}) \le \beta_i \tag{2}$$

Now putting, inequalities 1 and 2 together we have $h_{i,j}(\alpha_j, \beta_j) \le \beta_i$. This proves the first part of the induction hypothesis for the function h. Also note that $h_{i,j}$ is increasing in its first component and decreasing in the second component by invoking induction hypothesis and the fact that $\bar{f}^{-1}$ is increasing. Now we focus on function $r_{i,j}$. First note that there is an alternative form for $r_{i,j}$ by opening up the recursive relation. Let $\gamma_z = \frac{z}{c'-z} - \frac{z-1}{c'-z+1}$. We have ,

$$\begin{aligned}
r_{i,j}(\alpha, \beta) &= r_{j,j}(\alpha, \beta) + \frac{i}{c' - i}h_{i,j}(\alpha, \beta) - \frac{j-1}{c' - j + 1}h_{j,j}(\alpha, \beta) + \sum_{z=i+1}^{j-1} \gamma_z h_{z,j}(\alpha, \beta) \\
&= r_{j,j}(\alpha, \beta) + \frac{i}{c' - i}h_{i,j}(\alpha, \beta) - \frac{j}{c' - j}h_{j,j}(\alpha, \beta) + \sum_{z=i+1}^{j} \gamma_z h_{z,j}(\alpha, \beta) \\
&= \alpha - \frac{j}{c' - j}\beta + \frac{i}{c' - i}h_{i,j}(\alpha, \beta) + \sum_{z=i+1}^{j} \gamma_z h_{z,j}(\alpha, \beta).
\end{aligned} \tag{3}$$

Now we show that for all i we have

$$\alpha_i = \frac{i}{c' - i}\beta_i + \sum_{z=i+1}^{m} \gamma_z \beta_z. \tag{4}$$

This is because we have

$$\alpha_i - \frac{i}{c' - i}\beta_i = \sum_{z=i+1}^{c'} (\frac{z}{m} - \frac{i(c' - z)}{(c' - i)m})p_z.$$

On the other hand we have

$$\begin{aligned}
\sum_{z=i+1}^{m} \gamma_z \beta_z &= \sum_{z=i+1}^{m} (\sum_{z'=i+1}^{z} \gamma_{z'})\frac{c' - z}{m}p_z \\
&= \sum_{z=i+1}^{m} (\frac{z}{c' - z} - \frac{i}{c' - i})\frac{c' - z}{m}p_z \\
&= \sum_{z=i+1}^{m} (\frac{z}{m} - \frac{i(c' - z)}{(c' - i)m})p_z
\end{aligned}$$

and this shows that Equation 4 is correct. Therefore for all $i < j$ we have

$$\alpha_i - \alpha_j = \frac{i}{c' - i}\beta_i - \frac{j}{c' - j}\beta_j + \sum_{z=i+1}^{j} \gamma_z \beta_z$$

Now, using the induction hypothesis for h we have,

$$\alpha_i \geq \alpha_j + \frac{i}{c' - i} h_{i,j}(\alpha_j, \beta_j) - \frac{j}{c' - j}\beta_j + \sum_{z=i+1}^{j} \gamma_z h_{z,j}(\alpha_j, \beta_j). \tag{5}$$

Now verify that the right hand side of Equation 5 is equal to $r_{i,j}(\alpha_j, \beta_j)$ by the formulation of Equation 3

Also, using the induction hypothesis, we can observe that the right hand side of 3 is increasing in α_j and decreasing in β_j because all terms there are increasing in α_j and decreasing in β_j. $\square$

This lemma enables us to prove that algorithm 3 is deciding a valid upper bound on the probability correctly guessing c examples out of c' guesses. To prove this, assume that the probability of such event is equal to τ', Note that this means $\alpha_c + \beta_c = \frac{c'}{m}\tau'$. Also note that

$$\frac{\alpha_c}{\beta_c} \geq \frac{c}{c' - c} \tag{6}$$

therefore, we have

$$\alpha_c \geq \frac{c}{m}\tau' \tag{7}$$

and $\beta_c \leq \frac{c'-c}{m}\tau'$. Therefore, using Lemma 11 we have $\alpha_0 \geq r_{0,c}(\frac{c}{m}\tau', \frac{c'-c}{m}\tau')$ and $\beta_0 \geq h_{0,c}(\frac{c}{m}\tau', \frac{c'-c}{m}\tau')$.

Now we prove a lemma about the function $s_{i,j}(\tau) = h_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) + r_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$.

Lemma 15. *the function $s_{i,j}(\tau) = h_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) + r_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$ is increasing in τ for $i < j \leq c$.*

Proof. To prove this, we show that for all $i < j \leq c$ both $r_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$ and $h_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$ are increasing in τ. We prove this by induction on $j - i$. For $j - i = 1$, we have

$$h_{i,i+1}(\frac{c}{m}\tau, \frac{c' - c}{m}\tau) = (k-1)\bar{f}^{-1}(\frac{c}{m}\tau).$$

We know that $\bar{f}^{-1}$ is increasing, therefore $h_{i,i+1}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$ is increasing in τ as well. For $r_{i,i+1}$ we have

$$r_{i,i+1}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) = \frac{c}{m}\tau + \frac{i}{c'-i}(h_{i,i+1}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) - \frac{c'-c}{m}\tau)$$

So we have

$$\begin{aligned} r_{i,i+1}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) &= \frac{c(c'-i) - i(c'-c)}{m(c'-i)}\tau + \frac{i}{c'-i}h_{i,i+1}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) \\ &= \frac{(c-i)c'}{m(c'-i)}\tau + \frac{i}{c'-i}h_{i,i+1}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau). \end{aligned}$$

We already proved that $h_{i,i+1}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$ is increasing in τ. We also have $\frac{(c-i)c'}{m(c'-i)} > 0$, since $i < c$. Therefore

$$r_{i,i+1}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$$

is increasing in τ. So the base of induction is proved. Now we focus on $j - i > 1$. For $h_{i,j}$ we have

$$h_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) = (k-1)\bar{f}^{-1}(r_{i+1,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau).$$

By the induction hypothesis, we know that $r_{i+1,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$ is increasing in τ, and we know that $\bar{f}^{-1}$ is increasing, therefore, $h_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau)$ is increasing in τ.

For $r_{i,j}$, note that we rewrite it as follows

$$r_{i,j}(\alpha, \beta) = \alpha - \frac{j}{c'-j}\beta + \sum_{z=i}^{j-1} \lambda_z \cdot h_{z,j}(\alpha, \beta)$$

where $\lambda_z = (\frac{z+1}{c'-z-1} - \frac{z}{c'-z}) \geq 0$. Therefore, we have

$$\begin{aligned} r_{i,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) &= \tau(\frac{c}{m} - \frac{(c'-c)j}{m(c'-j)}) + \sum_{z=i}^{j-1} \lambda_z \cdot h_{z,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau) \\ &= \tau\frac{c'(c-j)}{m(c'-j)} + \sum_{z=i}^{j-1} \lambda_z \cdot h_{z,j}(\frac{c}{m}\tau, \frac{c'-c}{m}\tau). \end{aligned}$$

Now we can verify that all terms in this equation are increasing in τ, following the induction hypothesis and the fact that $\lambda_z > 0$ and also $j \leq c$. □

Now using this Lemma, we finish the proof. Note that we have $\alpha_0 + \beta_0 = \frac{c'}{m}$.

So assuming that $\tau' \geq \tau$, then we have

$$\frac{c'}{m} = \alpha_0 + \beta_0 \geq s_{0,c}(\tau') \geq s_{0,c}(\tau).$$

The last step of algorithm checks if $s_{0,c} \geq \frac{c'}{m}$ and it concludes that $\tau' \leq \tau$ if that's the case, because $s_{0,c}$ is increasing in τ. This means that the probability of having more than c guesses cannot be more than τ. □

B Ablation Experiments

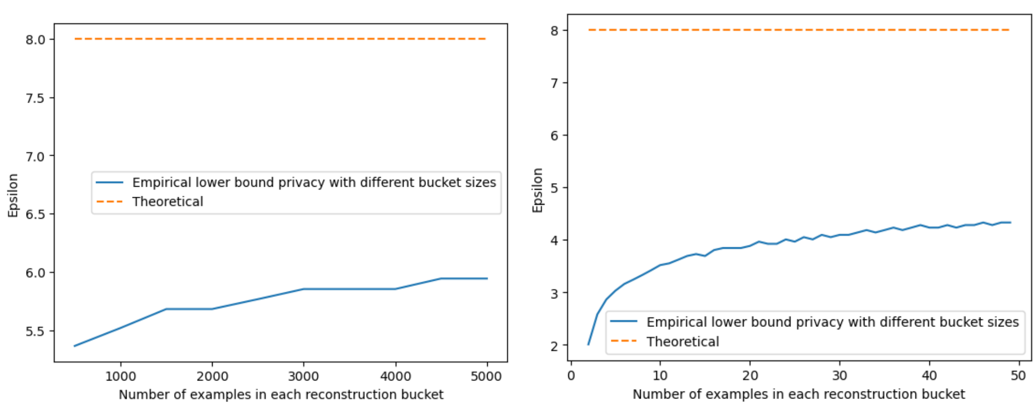

Figure 5: Effect of bucket size on the empirical lower bounds for reconstruction attack (Gaussian mechanism with standard deviation 0.6). Left: 10,000 canaries with bucket size up-to 5000. Right: 100 canaries with bucket-size up-to 50.

Reconstruction attacks: To show the effect of the bucket size (k) on the auditing performance, in Figure 5, we change the number of examples in the two different setups. In first setup we use 10,000 canaries and change the bucket size from 50 to 5000. In the other setup we only use 100 canaries and change the bucket-size from 3 to 50. Note that in these experiments, we do not use abstention and only consider adversaries that guess all examples.

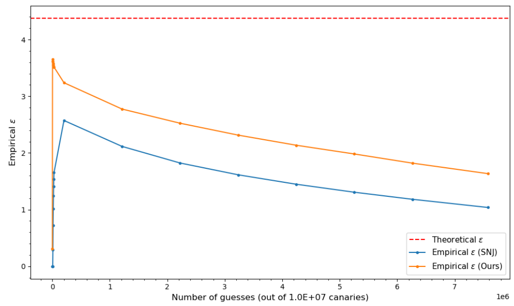

Figure 6: Effect of number of guesses (Gaussian mechanism with standard deviation 1.0)

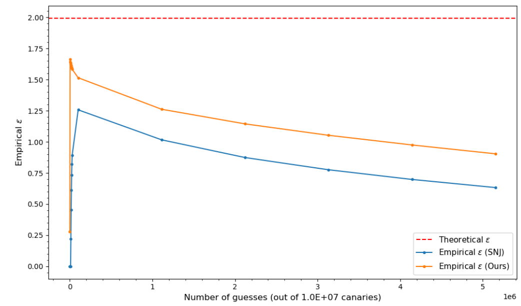

Figure 7: Effect of number of guesses (Gaussian mechanism with standard deviation 2.0)

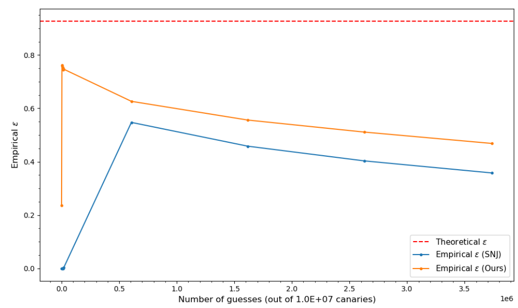

Figure 8: Effect of number of guesses (Gaussian mechanism with standard deviation 4.0)

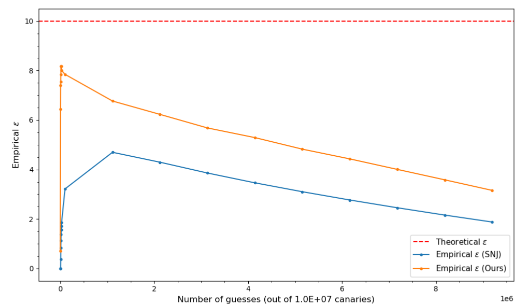

Figure 9: Effect of number of guesses (Gaussian mechanism with standard deviation 0.5)

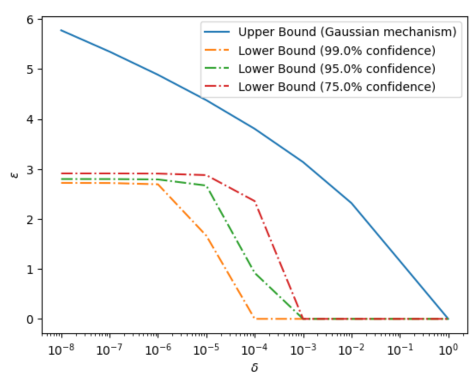

Figure 10: Idealized setting for different values of δ and confidence levels for bounds of Steinke et al. (2023).

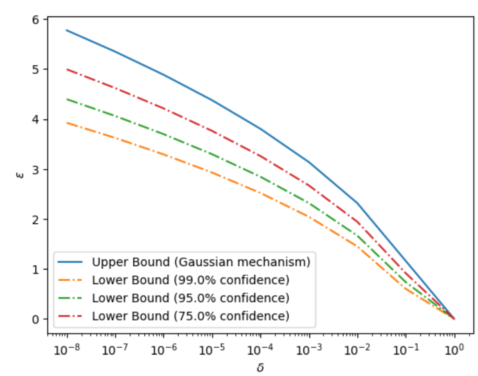

Figure 11: Idealized setting for different values of δ and confidence levels for our bounds.

Effect of number of guesses In Figures 6–9, we compare the theoretical upper bound, our lower bound and the bound of Steinke et al. lower bound with varying number of guesses. In total, we have $m = 10^7$ canaries. The number of correct guesses is determined by using Algorithm 4 (the idealized setting). Then we use our and Steinke et al. (2023)'s auditing with the resulting numbers and report the empirical ϵ. As we can see, both our and Steinke et al's auditing procedure achieve the best auditing performance for small number of guesses. This shows the importance of abstention in auditing.

A curious reader might wonder why the number of guesses has such a big impact on empirical privacy. Essentially, our analysis involves estimating how many correct guesses an adversary can make when given a certain number of attempts. We focus on specific percentiles of these distributions. The accuracy of our empirical privacy estimates can vary significantly based on how much the number of correct guesses fluctuates, which is influenced by how many guesses we allow the adversary to make. To explain further, consider a random variable representing the ratio of correct guesses (c) to total guesses (c'). If we reduce the number of guesses the variance of this ratio tends to decrease because the ratio approaches 1 (the adversary can make more correct guesses when we decrease c'). Conversely, if we increase the number of guesses, the variance can also decrease because having more guesses generally leads to a more stable average, owing to the law of large numbers. This balance makes the number of guesses a crucial factor to optimize for the best estimation of empirical privacy.

Varying δ and confidence levels: We also examine the effect of δ on the obtained empirical ϵ. We fix the number of canaries to 10^5 and the number of guesses to $1,500$ and the number of correct guesses are set to $1,429$, suggested by the idealized setting. We use a Gaussian mechanism with standard deviation 1.0, we vary the value of δ and the confidence level to observe how they affect the results. Figures 10 and 11 shows the bound of Steinke et al. (2023) and our bound, respectively. Note that our lower bounds represent the true behavior of δ independent of the confidence level, in contrast to the bound of Steinke et al. (2023).

C EXPERIMENTAL DETAILS

Idealized setting: In the idealized setting, we work with a toy version of the mechanism to calculate the *expected* number of correct guesses for the ideal adversary. For Gaussian mechanism, the ideal setting for an adversary is when we have a Gaussian mechanism that is used to calculate the sum of vectors. In this setting, each canary represents a unit vector that is orthogonal to all other canary vectors. Then, given the noisy sum, the adversary will calculate the likelihood of the canary being used in the sum, and then decides on the guesses based on these likelihoods. For the setting that the adversary has more than 2 guesses ($k > 2$), we use a slightly different idealized setting. In all settings, we run the attack 100 times and average the result to get the expected number of correct guesses. Algorithm 4 shows how we calculate the number of correct guesses in the idealized setting.

Algorithm 4 Simulate the Number of Correct Guesses

```
import numpy as np
from scipy.special import softmax
from numpy.random import normal, binomial
def idealized_setting(target_noise, n_guesses, n_canaries, k):
    n_correct_vec = []
    if k==2:
        for _ in range(100):
            s_vector = binomial(1, 0.5, size=n_canaries) * 2 - 1
            noise = normal(0, 2*target_noise, n_canaries)

            noisy_s = s_vector + noise

            sorted_noisy_s = np.sort(noisy_s)


            threshold_c = sorted_noisy_s[-int(n_guesses)//2-1]
            n_correct = np.ceil(n_guesses*(s_vector[noisy_s >
             ↪  threshold_c] == 1).mean())

            n_correct_vec.append(n_correct)
    else:
        for _ in range(100):
            s_recon_vec = np.random.randint(0, k, n_canaries)

            s_vec_recn_ohe = np.eye(k)[s_recon_vec]
            s_recon_noisy_vec_ohe = s_vec_recn_ohe + normal(0,
             ↪  np.sqrt(2)*target_noise, s_vec_recn_ohe.shape)

            idx_max = np.argmax(s_recon_noisy_vec_ohe, axis=1)

            buckets =
             ↪  softmax(s_recon_noisy_vec_ohe/(2*target_noise**2),
             ↪  axis=1)[np.arange(s_recon_noisy_vec_ohe.shape[0]),
             ↪  idx_max]
            sorted_buckets = np.sort(buckets)
            bucket_c_thr = sorted_buckets[-int(n_guesses)]

            n_correct_rec = np.ceil(
                n_guesses*(s_recon_vec[buckets > bucket_c_thr] ==
                 ↪  s_recon_noisy_vec_ohe[buckets >
                 ↪  bucket_c_thr].argmax(1)).mean()
            )
            n_correct_vec.append(n_correct_rec)

    return int(np.array(n_correct_vec).mean(0))
```

AUDITING CODE

Here we include the code to compute empirical epsilon.

```
from scipy.stats import norm
import numpy as np

# Calculate h and r recursively (no abstentions)
def rh(inverse_blow_up_function, alpha, beta, j, m, k=2):
    # Initialize lists to store h and r values
    h = [0 for _ in range(j + 1)]
    r = [0 for _ in range(j + 1)]
    # Set initial values for h and r
    h[j] = beta
    r[j] = alpha
    # Iterate from j-1 to 0
    for i in range(j - 1, -1, -1):
        # Calculate h[i] using the maximum of h[i+1] and a scaled inverse blow-up function
        h[i] = max(h[i + 1], (k - 1) * inverse_blow_up_function(r[i + 1]))
        # Update r[i] based on the difference between h[i] and h[i+1]
        r[i] = r[i + 1] + (i / (m - i)) * (h[i] - h[i + 1])
    # Return the lists of h and r values
    return (r, h)

# Audit function without abstention
def audit_rh(inverse_blow_up_function, m, c, threshold=0.05, k=2):
    # Calculate alpha and beta values
    alpha = threshold * c / m
    beta = threshold * (m - c) / m
    # Call the rh function to get the lists of h and r values
    r, h = rh(inverse_blow_up_function, alpha, beta, c, m, k)
    # Check if the differential privacy condition is satisfied
    if r[0] + h[0] > 1.0:
        return False
    else:
        return True

# Calculate h and r recursively (with abstentions)
def rh_with_cap(inverse_blow_up_function, alpha, beta, j, m,c_cap, k=2):
    h=[0 for i in range(j+1)]
    r=[0 for i in range(j+1)]
    h[j]= beta
    r[j]= alpha
    for i in range(j-1,-1,-1):

        h[i]=max(h[i+1],(k-1)*inverse_blow_up_function(r[i+1]))
        r[i]= r[i+1] + (i/(c_cap-i))*(h[i] - h[i+1])

    return (r,h)

# Audit function with abstentions
def audit_rh_with_cap(inverse_blow_up_function, m, c,c_cap, threshold=0.05, k=2):
    threshold=threshold*c_cap/m
```

```
        alpha=(threshold*c/c_cap)
        beta=threshold*(c_cap-c)/c_cap
        r,h=rh_with_cap(inverse_blow_up_function, alpha, beta, c,
        ↪  m, c_cap, k)

        if r[0]+h[0]>c_cap/m:
            return False
        else:
            return True

    # Calculate the blow-up function for Gaussian noise
    def gaussianDP_blow_up_function(noise):
        def blow_up_function(x):
            # Calculate the threshold value
            threshold = norm.ppf(x)
            # Calculate the blown-up threshold value
            blown_up_threshold = threshold + 1 / noise
            # Return the CDF of the blown-up threshold value
            return norm.cdf(blown_up_threshold)
        return blow_up_function

    # Calculate the inverse blow-up function for Gaussian noise
    def gaussianDP_blow_up_inverse(noise):
        def blow_up_inverse_function(x):
            # Calculate the threshold value
            threshold = norm.ppf(x)
            # Calculate the blown-up threshold value
            blown_up_threshold = threshold - 1 / noise
            # Return the CDF of the blown-up threshold value
            return norm.cdf(blown_up_threshold)
        return blow_up_inverse_function

    # Define a function to calculate delta for Gaussian noise
    def calculate_delta_gaussian(noise, epsilon):
        # Calculate delta using the formula
        delta = norm.cdf(-epsilon * noise + 1 / (2 * noise)) -
        ↪  np.exp(epsilon) * norm.cdf(-epsilon * noise - 1 / (2 *
        ↪  noise))
        return delta

    # Define a function to calculate epsilon for Gaussian noise
    def calculate_epsilon_gaussian(noise, delta):
        # Set initial bounds for epsilon
        epsilon_upper = 100
        epsilon_lower = 0
        # Perform binary search to find epsilon
        while epsilon_upper - epsilon_lower > 0.001:
            epsilon_middle = (epsilon_upper + epsilon_lower) / 2
            if calculate_delta_gaussian(noise, epsilon_middle) >
            ↪  delta:
                epsilon_lower = epsilon_middle
            else:
                epsilon_upper = epsilon_middle
        # Return the upper bound of epsilon
        return epsilon_upper

    # Get the empirical epsilon value
    def get_gaussian_emp_eps_ours(candidate_noises,
    ↪  inverse_blow_up_functions, m, c, threshold, delta, k=2):
```

```
        # Initialize the empirical privacy index
        empirical_privacy_index = 0
        # Iterate through candidate noises until the privacy condition fails
        while audit_rh(inverse_blow_up_functions[empirical_privacy_index], m, c, threshold=0.05, k=k):
            empirical_privacy_index += 1
        # Get the empirical noise and calculate the empirical epsilon
        empirical_noise = candidate_noises[empirical_privacy_index]
        empirical_eps = calculate_epsilon_gaussian(empirical_noise, delta=delta)
        # Return the empirical epsilon
        return empirical_eps

    # Set target noise and generate candidate noises
    target_noise = 0.6

    candidate_noises=[target_noise+ i*0.01 for i in range(1000)]
    inverse_blow_up_functions=[gaussianDP_blow_up_inverse(noise) for noise in candidate_noises]
```