DP-Sniper: Black-Box Discovery of Differential Privacy Violations using Classifiers

Benjamin Bichsel, Samuel Steffen, Ilija Bogunovic, Martin Vechev
ETH Zurich, Switzerland
{benjamin.bichsel, samuel.steffen, ilija.bogunovic, martin.vechev}@inf.ethz.ch

***Abstract*—We present DP-Sniper, a practical black-box method that automatically finds violations of differential privacy.**

DP-Sniper is based on two key ideas: (i) training a classifier to predict if an observed output was likely generated from one of two possible inputs, and (ii) transforming this classifier into an approximately optimal attack on differential privacy.

Our experimental evaluation demonstrates that DP-Sniper obtains up to 12.4 times stronger guarantees than state-of-the-art, while being 15.5 times faster. Further, we show that DP-Sniper is effective in exploiting floating-point vulnerabilities of naively implemented algorithms: it detects that a supposedly 0.1-differentially private implementation of the Laplace mechanism actually does not satisfy even 0.25-differential privacy.

***Index Terms*—differential privacy; differential distinguishability; inference attacks; machine learning; classifiers**

I. Introduction

Differential privacy [1] is considered the gold standard for quantifying the level of privacy guaranteed by an algorithm. Traditionally, it assesses randomized algorithms $M\colon \mathbb{A} \to \mathbb{B}$ that operate on databases $a \in \mathbb{A}$ and produce output $M(a) \in \mathbb{B}$. Intuitively, M is differentially private if changing the data of a single user in a does not significantly affect the distribution of $M(a)$. In particular, differential privacy precludes attackers from deciding if an output $M(A)$ was generated using $A = a$ or $A = a'$, if a and a' differ in the data of a single user.

Formally and more generally, M is ϵ-differentially private (ϵ-DP) if for every pair of neighboring inputs $(a, a') \in \mathcal{N}$ and for every attack $\mathcal{S} \in \mathcal{P}(\mathbb{B})$,

$$\ln(\Pr[M(a) \in \mathcal{S}]) - \ln(\Pr[M(a') \in \mathcal{S}]) \leq \epsilon, \qquad (1)$$

where $\mathcal{P}(\mathbb{B})$ denotes the power set of $\mathbb{B}$, and the neighborhood $\mathcal{N}$ captures the effect of changing the data of a single user. We note that the probability in Eq. (1) is over M, which may return a different output value for the same input in different runs. Perhaps unintuitively, smaller ϵ provides stronger privacy guarantees: $\epsilon = 0$ ensures perfect privacy and $\epsilon = \infty$ offers no privacy, while achieving $\epsilon < 0$ is impossible.

Guaranteeing privacy in practice raises the following key question: what level of differential privacy does an algorithm satisfy? Accurately answering this question is critical, since overestimating privacy guarantees leads to privacy leaks, while underestimation motivates unnecessary changes to the algorithm such as adding more noise and thereby decreasing the output's utility. As a consequence, automatically verifying that a given algorithm is ϵ-DP is an important area of research [2]–[10]. However, automated verification is generally incomplete and may fail to prove that an algorithm is ϵ-DP, even if that is the case.

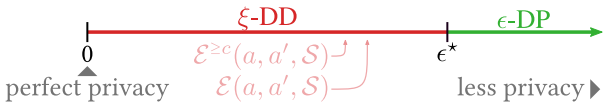

Fig. 1. Differential distinguishability (DD) and differential privacy (DP).

Differential Distinguishability. A complementary line of work [11]–[16] is concerned with showing that a given algorithm *cannot* be ϵ-DP by establishing *differential distinguishability*. [1] Formally, a randomized algorithm $M\colon \mathbb{A} \to \mathbb{B}$ is *ξ-differentially distinguishable* (ξ-DD) if there exists a witness $(a, a', \mathcal{S})$ with $(a, a') \in \mathcal{N}$ and $\mathcal{S} \in \mathcal{P}(\mathbb{B})$, for which

$$\ln(\Pr[M(a) \in \mathcal{S}]) - \ln(\Pr[M(a') \in \mathcal{S}]) \geq \xi. \qquad (2)$$

Relationship to ϵ-DP. Fig. 1 illustrates the relationship between ξ-DD and ϵ-DP: If M is ξ-DD, it cannot be ϵ-DP for any $\epsilon < \xi$. Equivalently, if M is both ξ-DD and ϵ-DP, then $\xi \leq \epsilon$. Therefore, differential distinguishability yields a lower bound on differential privacy, implying in particular that larger differential distinguishability is strictly more informative. Fig. 1 also illustrates $\mathcal{E}$ (discussed next) and $\mathcal{E}^{\geq c}$ (see §II).

Applications. Automatically detecting differential distinguishability is useful in a wide range of scenarios. First, it allows developers to evaluate whether algorithms or proposed implementations are differentially private. Second, a witness demonstrating differential distinguishability may indicate *why* a given algorithm is not differentially private. Third, even for algorithms which are well-known to be differentially private, investigating differential distinguishability is valuable, as a failure to demonstrate strong differential distinguishability may suggest that such algorithms are more private than previously known.

Search Problem. Demonstrating strong differential distinguishability requires finding a witness $(a, a', \mathcal{S})$ with maximal *power*, defined by [2]

$$\mathcal{E}(a, a', \mathcal{S}) := \ln(\Pr[M(a) \in \mathcal{S}]) - \ln(\Pr[M(a') \in \mathcal{S}]). \quad (3)$$

[1] While this goal has been formally described before e.g. in [13], we are the first to introduce the term differential distinguishability.

[2] Note that this terminology is used differently than in the context of statistical hypothesis testing.

As indicated in Fig. 1, witness $(a, a', \mathcal{S})$ demonstrates ξ-DD for $\xi = \mathcal{E}(a, a', \mathcal{S})$, and differential privacy cannot hold for $\epsilon < \mathcal{E}(a, a', \mathcal{S})$. Further, Fig. 1 shows that the most powerful witness has power $\epsilon^\star$, which is simultaneously (i) the largest possible parameter ξ for which M is ξ-DD and (ii) the smallest possible parameter ϵ for which M is ϵ-DP.

Key Challenge: Searching for Attacks. Searching for powerful witnesses amounts to searching for (i) inputs $(a, a') \in \mathcal{N}$ and (ii) attacks $\mathcal{S} \in \mathcal{P}(\mathbb{B})$. Existing work has shown that for (i), simple sampling, heuristic, or exhaustive approaches are often sufficient (see §VII-A).

However, searching for attacks $\mathcal{S}$ is inherently challenging, as the number of possible attacks is $2^{|\mathbb{B}|}$ (in contrast, the input search space has size $|\mathcal{N}| \leq |\mathbb{A}|^2$), meaning that exhaustive search is intractable even for small $\mathbb{B}$, and clearly infeasible for continuous $\mathbb{B}$. Due to this difficulty, existing work does not sufficiently address the search for attacks, as it uses simple heuristics that do not generalize well [12] or exhaustive approaches that do not scale [17]. As a consequence, these approaches perform poorly on state-of-the-art differentially private algorithms such as RAPPOR [18].

This Work. Our work addresses this problem and aims to find the most powerful attack[3] for given inputs $(a, a') \in \mathcal{N}$:

$$\underset{\mathcal{S} \in \mathcal{P}(\mathbb{B})}{\arg\max}\ \mathcal{E}(a, a', \mathcal{S}). \tag{4}$$

Unfortunately, solving Eq. (4) exactly is impossible (cp. [19, Thm. 10]) because computing $\mathcal{E}(a, a', \mathcal{S})$ requires determining probabilities $\Pr[M(a) \in \mathcal{S}]$ and $\Pr[M(a') \in \mathcal{S}]$ that can typically only be estimated, given black-box access to M. As these probabilities can become arbitrarily small, the resulting estimate of $\mathcal{E}(a, a', \mathcal{S})$ can become arbitrarily imprecise, assuming bounded runtime.

Key Insight. To address this problem, we introduce a tractable surrogate problem to Eq. (4) that can be solved without estimating small probabilities, and present an algorithm *DP-Sniper* solving this problem approximately optimally.

DP-Sniper (i) trains a machine learning classifier to determine the probability that $b = M(A)$ was generated using $A = a$ instead of $A = a'$, and (ii) transforms this classifier into an approximately optimal attack $\mathcal{S}$, inspired by the Neyman-Pearson lemma [20, Thm. 3.2.1]. At a high level, we select the attack $\mathcal{S}$ to consist of outputs b which are particularly likely to originate from $A = a$, i.e., for which $\Pr[A = a \mid M(A) = b] \geq t$ for a threshold t. By controlling t, we control the probabilities $\Pr[M(a) \in \mathcal{S}]$ and $\Pr[M(a') \in \mathcal{S}]$, which in turn define $\mathcal{E}(a, a', \mathcal{S})$. We select t as large as possible to maximize $\mathcal{E}(a, a', \mathcal{S})$, but small enough to avoid estimating arbitrarily small probabilities.

Main Contributions. Our main contributions are:

- DP-Sniper, a practical black-box approach to demonstrate differential distinguishability using classifiers (§III).
- A derivation of DP-Sniper's formal guarantees, showing its results are approximately optimal (§IV).
- A thorough evaluation of an end-to-end implementation[4] of DP-Sniper, showing it can demonstrate substantially stronger differential distinguishability by a factor of up to 12.4, while being 15.5 times faster on average, compared to a state-of-the-art baseline (§VI).
- The first automated approach testing if a supposedly differentially private implementation of an algorithm exhibits floating-point vulnerabilities (§VI-D).

[3]Note that we do *not* suggest a novel solution to find powerful inputs, but instead rely on an existing approach [12].

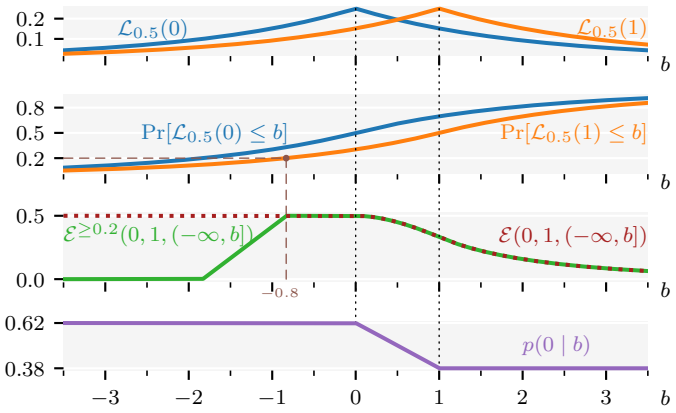

Fig. 2. Properties of Laplace mechanism $\mathcal{L}_{0.5}$ (top to bottom): probability density function, cumulative distribution function, power, and posterior prob.

II. Problem Statement

In this section, we introduce the surrogate problem addressed by this work. In contrast to Eq. (4), this problem can be solved without estimating arbitrarily small probabilities. We start by introducing a running example.

Example 1. The Laplace mechanism $\mathcal{L}_{0.5} \colon \mathbb{R} \to \mathbb{R}$ adds Laplace noise with mean 0 and scale $1/0.5 = 2$ to its input: $\mathcal{L}_{0.5}(a) = a + \text{lap}(0, 2)$ [1, Ex. 1]. Fig. 2 shows the probability density function and cumulative distribution function (top two panels) for $\mathcal{L}_{0.5}(0)$ and $\mathcal{L}_{0.5}(1)$. It is well-known (see [1, Prop. 1]) that $\mathcal{L}_{0.5}$ is 0.5-DP for neighborhood $\mathcal{N} = \{(a, a') \mid |a - a'| \leq 1\}$.

A. Avoiding Small Probabilities

Finding the most powerful attack according to Eq. (4) is typically impossible, as exactly computing the power $\mathcal{E}(a, a', \mathcal{S})$ of a given attack is usually intractable. This is because the probabilities $\Pr[M(a) \in \mathcal{S}]$ and $\Pr[M(a') \in \mathcal{S}]$ determining the power can generally only be estimated. Specifically, provided samples $b_0, \ldots, b_{N-1}$ from $M(a)$, we can estimate $\Pr[M(a) \in \mathcal{S}]$ for a given $\mathcal{S}$ as

$$\hat{P}^N_{M(a) \in \mathcal{S}} := \tfrac{1}{N} \textstyle\sum_{i=0}^{N-1} [b_i \in \mathcal{S}],$$

[4]Publicly available at https://github.com/eth-sri/dp-sniper

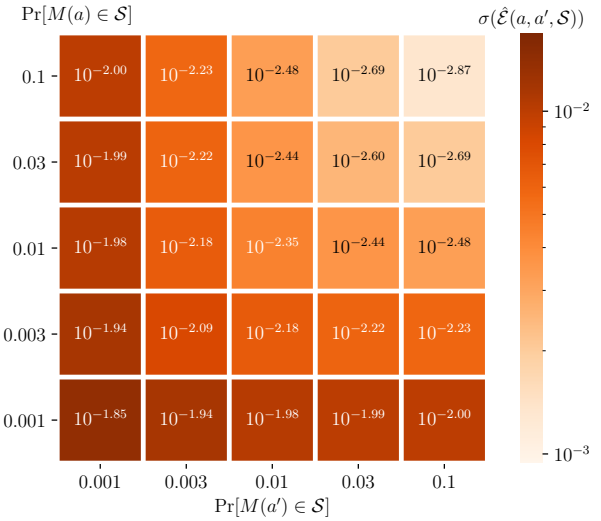

Fig. 3. Empirical standard deviation of $\hat{\mathcal{E}}(a, a', \mathcal{S})$ for arbitrary M, depending on the magnitude of probabilities $\Pr[M(a) \in \mathcal{S}]$ and $\Pr[M(a') \in \mathcal{S}]$, using $n = 10^7$ samples to estimate each probability.

where $[\phi]$ is the Iverson bracket that evaluates to 1 if ϕ is true and 0 otherwise. From $\hat{P}^N_{M(a)\in\mathcal{S}}$ and $\hat{P}^N_{M(a')\in\mathcal{S}}$ (computed analogously), we can then estimate $\mathcal{E}(a, a', \mathcal{S})$ as

$$\hat{\mathcal{E}}(a, a', \mathcal{S}) := \ln\left(\hat{P}^N_{M(a)\in\mathcal{S}}\right) - \ln\left(\hat{P}^N_{M(a')\in\mathcal{S}}\right). \tag{5}$$

Effect of Small Probabilities. As demonstrated in Fig. 3, the standard deviation of $\hat{\mathcal{E}}(a, a', \mathcal{S})$ increases significantly as $\Pr[M(a) \in \mathcal{S}]$ and $\Pr[M(a') \in \mathcal{S}]$ become smaller. For example, if both probabilities are 0.001, the standard deviation of $\hat{\mathcal{E}}(a, a', \mathcal{S})$ when using 10^7 samples is $10^{-1.85} \approx 0.014$, which is substantial if we want to demonstrate 0.1-DD: even if we find $(a, a', \mathcal{S})$ with $\mathcal{E}(a, a', \mathcal{S}) = 0.1$, we can only estimate $\hat{\mathcal{E}}(a, a', \mathcal{S})$ in the range 0.1 ± 0.014.

To address this problem, we avoid estimating probabilities below some small constant $c \in (0, 1]$, and instead search for a witness $(a, a', \mathcal{S})$ with maximal *c-power* defined as

$$\begin{aligned}\mathcal{E}^{\geq c}(a, a', \mathcal{S}) := \ &\ln^{\geq c}\left(\Pr[M(a) \in \mathcal{S}]\right) \\ &- \ln^{\geq c}\left(\Pr[M(a') \in \mathcal{S}]\right),\end{aligned} \tag{6}$$

where $\ln^{\geq c}(x) = \ln(\max(c, x))$. If both $\Pr[M(a) \in \mathcal{S}] \leq c$ and $\Pr[M(a') \in \mathcal{S}] \leq c$, then $\mathcal{E}^{\geq c}(a, a', \mathcal{S}) = 0$, reflecting the fact that we cannot derive meaningful information from probabilities below c. Hence, maximizing the c-power automatically avoids estimating small probabilities. In our experiments, we use $c = 0.01$.

Example 2 (c-Power)**.** For $\mathcal{L}_{0.5}$, $a = 0$, and $a' = 1$, the solid green line in Fig. 2 (third from top) shows the c-power $\mathcal{E}^{\geq c}(0, 1, \mathcal{S})$ of a specific family of attacks $\mathcal{S} = (-\infty, b]$.

In contrast to power (dotted red line in Fig. 2), c-power takes into account that attacks with small b are less desirable as quantifying them requires estimating small probabilities.

Importantly, $\mathcal{E}^{\geq c}$ provides DD guarantees analogous to $\mathcal{E}$. In particular, every witness $(a, a', \mathcal{S})$ demonstrates that M is $\mathcal{E}^{\geq c}(a, a', \mathcal{S})$-DD, as shown in Lem. 1.

Lemma 1 (Guarantees for c-power)**.** For any algorithm $M\colon \mathbb{A} \to \mathbb{B}$, and any witness $(a, a', \mathcal{S})$ with $(a, a') \in \mathcal{N}$, M satisfies ξ-DD for $\xi = \mathcal{E}^{\geq c}(a, a', \mathcal{S})$.

We provide a proof of Lem. 1 in App. B. Note that under the reasonable assumption that we cannot accurately estimate probabilities below c, Lem. 1 provides the strongest possible guarantee for a given witness $(a, a', \mathcal{S})$.

Surrogate Problem Statement. In this work, we aim to find an attack that maximizes c-power for given $(a, a') \in \mathcal{N}$:

$$\operatorname*{arg\,max}_{\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]} \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}) \tag{7}$$

In contrast to Eq. (4), Eq. (7) replaces power by c-power, and deterministic attacks $\mathcal{S} \in \mathcal{P}(\mathbb{B})$ by randomized attacks $\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0, 1]$ (discussed shortly).

Example 3. A (deterministic) solution for Eq. (7) on our running example is $\mathcal{S}^\star = (-\infty, -0.8]$. In particular, we have $\mathcal{E}^{\geq c}(0, 1, \mathcal{S}^\star) = 0.5$ (indicated by the vertical dashed line). A more powerful attack does not exist because $\mathcal{L}_{0.5}$ is 0.5-DP.

B. Randomized Attacks

We now discuss randomized attacks and demonstrate that they can be more powerful than deterministic attacks in terms c-power. Importantly, the guarantees of Lem. 1 also apply to randomized attacks (see App. C), meaning that we can safely extend our search space to randomized attacks.

Determinstic vs. Randomized. We first note that the domain of deterministic attacks $\mathcal{P}(\mathbb{B})$ is isomorphic to the set of functions $\mathbb{B} \to \{0, 1\}$, as any attack $\mathcal{S} \in \mathcal{P}(\mathbb{B})$ decides for every $b \in \mathbb{B}$ if $b \in \mathcal{S}$ (encoded as $\mathcal{S}(b) = 1$) or $b \notin \mathcal{S}$ (encoded as $\mathcal{S}(b) = 0$). In contrast, a randomized attack $\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0, 1]$ can also include portions of elements from $\mathbb{B}$ by controlling $\boldsymbol{\mathcal{S}}(b) \in [0, 1]$. For example, $\boldsymbol{\mathcal{S}}(b) = 0.5$ includes b in $\boldsymbol{\mathcal{S}}$ with probability 0.5. In the following, we write $\Pr[b \in \boldsymbol{\mathcal{S}}]$ for $\boldsymbol{\mathcal{S}}(b)$.

Throughout this work, we treat $\boldsymbol{\mathcal{S}}$ as a randomized set whose elements are selected probabilistically. We consistently write $\boldsymbol{\mathcal{S}}$ to indicate randomized attacks from $\mathbb{B} \to [0, 1]$, and $\mathcal{S}$ to indicate deterministic attacks from $\mathcal{P}(\mathbb{B})$.

Limitations of Deterministic Attack. Randomized attacks are at least as powerful as deterministic attacks, because every deterministic attack can be interpreted as a randomized attack. However, as we demonstrate next, randomized attacks can be strictly more powerful than deterministic attacks when avoiding small probability attacks.

Example 4. Consider a threshold version of the Laplace mechanism $\mathcal{L}^{\spadesuit}_{0.5}\colon \mathbb{R} \to \{0, 1\}$ which returns whether the output of $\mathcal{L}_{0.5}$ lies below -1.83, i.e., $\mathcal{L}^{\spadesuit}_{0.5}(a) := [\mathcal{L}_{0.5}(a) \leq -1.83]$. Tab. I shows that for $\mathcal{L}^{\spadesuit}_{0.5}$ and $c = 0.2$, $\mathcal{E}^{\geq c}(0, 1, \mathcal{S}) \approx 0$ for all possible deterministic attacks $\mathcal{S}$.

This is because attacks $\{\}$ and $\mathbb{B}$ always have power zero (regardless of the algorithm M), attack $\{0\}$ has negative

TABLE I
Output distribution and power of all attacks $\mathcal{S}$ for $\mathcal{L}^{\spadesuit}_{0.5}$.

$\mathcal{S}$	$\{\}$	$\{0\}$	$\{1\}$	$\{0,1\}$
$\Pr[\mathcal{L}^{\spadesuit}_{0.5}(0) \in \mathcal{S}]$	0.00	0.80	0.20	1.00
$\Pr[\mathcal{L}^{\spadesuit}_{0.5}(1) \in \mathcal{S}]$	0.00	0.88	0.12	1.00
$\mathcal{E}(0,1,\mathcal{S})$	0.0	-0.1	0.5	0.0
$\mathcal{E}^{\geq 0.2}(0,1,\mathcal{S})$	0.0	-0.1	0.0	0.0

power because $\Pr[\mathcal{L}^{\spadesuit}_{0.5}(0) \in \{0\}] < \Pr[\mathcal{L}^{\spadesuit}_{0.5}(1) \in \{0\}]$, and the only promising attack $\mathcal{S} = \{1\}$ with power 0.5 gives $\Pr[\mathcal{L}^{\spadesuit}_{0.5}(0) \in \{1\}] \approx c$ and $\Pr[\mathcal{L}^{\spadesuit}_{0.5}(1) \in \{1\}] < c$.

The key problem in Ex. 4 is that deterministic attacks cannot partially select outputs. In principle, we would like to use the most powerful attack $\{1\}$. This however forces us to estimate the probability $\Pr[\mathcal{L}^{\spadesuit}_{0.5}(1) \in \{1\}] \approx 0.12$, which is smaller than $c = 0.2$. The only option to increase this probability is to add the output 0 to the attack set. However, this results in the trivial attack $\mathbb{B} = \{0,1\}$ with power 0.

To avoid this problem, we generalize attacks to be randomized. As shown below, this allows for more powerful attacks.

Example 5. For $\mathcal{L}^{\spadesuit}_{0.5}$ and $c = 0.2$, consider $\boldsymbol{\mathcal{S}}$ with $\Pr[0 \in \boldsymbol{\mathcal{S}}] = 0.1$ and $\Pr[1 \in \boldsymbol{\mathcal{S}}] = 1$. Then, $\mathcal{E}^{\geq c}(0,1,\boldsymbol{\mathcal{S}})$ is

$$\ln^{\geq c}\left(\Pr\left[\mathcal{L}^{\spadesuit}_{0.5}(0) \in \boldsymbol{\mathcal{S}}\right]\right) - \ln^{\geq c}\left(\Pr\left[\mathcal{L}^{\spadesuit}_{0.5}(1) \in \boldsymbol{\mathcal{S}}\right]\right)$$
$$= \ln^{\geq c}(0.80 \cdot 0.1 + 0.20) - \ln^{\geq c}(0.88 \cdot 0.1 + 0.12) \approx 0.3.$$

We note that Ex. 5 only demonstrates 0.3-DD, even though $\mathcal{L}^{\spadesuit}_{0.5}$ is actually 0.5-DD, as can be seen in Tab. I. However, proving 0.5-DD would require precisely estimating $\Pr[M(a') \in \{1\}] \approx 0.12$, which lies below $c = 0.2$.

Technicalities. The formal part of this work glosses over some technical details, which we discuss in the following.

First, in Eq. (1) and throughout this work, we follow the convention $\ln(0) - \ln(0) := 0$. Further, a solution to Eq. (4) or Eq. (7) might technically not exist because only the supremum (but not the maximum) exists—a subtlety ignored in this work.

III. Overview

We now introduce our main algorithm DP-Sniper, which finds approximately optimal attacks $\boldsymbol{\mathcal{S}}$ according to Eq. (7). We then introduce DD-Search, which leverages DP-Sniper to find witnesses $(a, a', \boldsymbol{\mathcal{S}})$ with high power.

A. DP-Sniper: Searching Attack

Alg. 1 presents the algorithm DP-Sniper. We discuss the individual steps in detail below. For a discussion of the theory behind DP-Sniper, we refer to §IV.

Threshold Attacks. The high-level goal of DP-Sniper is to construct a *threshold attack* $\boldsymbol{\mathcal{S}}^{t,q} \in \mathbb{B} \to [0,1]$ that contains elements $b \in \mathbb{B}$ which are likely to originate from $M(a)$, as opposed to $M(a')$.

Algorithm 1 DP-Sniper: Searching Attack $\boldsymbol{\mathcal{S}}$.

Hyper-parameters: probability bound $c \in (0,1]$, sample sizes N, N_{train}, family of classifiers $\{p_\theta(a|b)\}_{\theta \in \Theta}$

1: **function** DP-Sniper($M\colon \mathbb{A} \to \mathbb{B}$, a, a') ▷ $(a,a') \in \mathcal{N}$
2: $\quad\theta \leftarrow$ TrainClassifier(M, a, a')
3: $\quad b'_0, \dots, b'_{N-1} \sim M(a')$ ▷ sample N times from $M(a')$
4: $\quad p'_i = p_\theta(a|b'_i)$ for $i \in \{0, \dots, N-1\}$ ▷ score samples
5: $\quad p''_0, \dots, p''_{N-1} \leftarrow$ sort $p'_0, \dots, p'_{N-1}$ in descending order
6: $\quad t^\sharp \leftarrow p''_{\min(\lfloor c \cdot N \rfloor,\ N-1)}$ ▷ at most $c \cdot N$ probabilities above $t^\sharp$
7: $\quad q^\sharp \leftarrow \left(cN - \sum_{i=0}^{N-1}[p'_i > t^\sharp]\right) / \left(\sum_{i=0}^{N-1}[p'_i = t^\sharp]\right)$
8: $\quad \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}_\theta(\cdot) \leftarrow p_\theta(a|\cdot) \succeq (t^\sharp, q^\sharp)$ ▷ construct attack, see Eq. (9)
9: $\quad$ **return** $\boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}_\theta$
10: **function** TrainClassifier($M\colon \mathbb{A} \to \mathbb{B}$, a, a')
11: $\quad b_0, \dots, b_{N_{\text{train}}-1} \sim M(a)$ ▷ sample N_{train} times from $M(a)$
12: $\quad b'_0, \dots, b'_{N_{\text{train}}-1} \sim M(a')$ ▷ sample N_{train} times from $M(a')$
13: $\quad \mathcal{D} \leftarrow \{(a, b_i)\}_{i=0}^{N_{\text{train}}-1} \cup \{(a', b'_i)\}_{i=0}^{N_{\text{train}}-1}$ ▷ prepare training data
14: $\quad \theta \leftarrow$ train $p_\theta(a|b)$ on $\mathcal{D}$ ▷ train classifier
15: $\quad$ **return** θ

In particular, $\boldsymbol{\mathcal{S}}^{t,q}$ compares the posterior probability[5]

$$p(a|b) := \Pr[A = a \mid M(A) = b] \tag{8}$$

to t, assuming that the input A to M is chosen uniformly at random from $\{a, a'\}$, i.e., $\Pr[A = a] = \Pr[A = a'] = \frac{1}{2}$.

Intuitively, t acts as a threshold on $p(a|b)$, ensuring $\boldsymbol{\mathcal{S}}^{t,q}$ contains values b whose posterior probability $p(a|b)$ lies above t. In the case where $p(a|b) = t$, parameter q acts as a tie-break by including a portion q of b. We can interpret $\boldsymbol{\mathcal{S}}^{t,q}$ as a check of the (probabilistic) constraint $p(a|b) \succeq (t,q)$, which is defined to be always satisfied if $p(a|b) > t$, and satisfied with probability q if $p(a|b) = t$. Formally, for $p := p(a|b)$:

$$\Pr\left[b \in \boldsymbol{\mathcal{S}}^{t,q}\right] = \Pr[p \succeq (t,q)] := [p > t] + q \cdot [p = t]\,. \tag{9}$$

As we show later in §IV, only considering threshold attacks is sufficient when maximizing $\mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}})$, which can be derived from the Neyman-Pearson lemma.

Classifier. Line 2 trains a machine learning classifier $p_\theta(a|b)$ parametrized by θ, which approximates the probability $p(a|b)$ that any given output $b \in \mathbb{B}$ was sampled from $M(a)$, as opposed to $M(a')$. We discuss this in more detail in §III-B.

Searching Parameters. Lines 3–7 of Alg. 1 select the parameters $t^\sharp, q^\sharp \in [0,1]$ for the threshold attack constructed in Line 8. The goal is to select $t^\sharp, q^\sharp$ such that $\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right] = c$, which yields an optimal attack when using the perfect classifier $p_\theta(a|b) = p(a|b)$ (see §IV).

First, Line 3 generates N fresh samples b'_i from $M(a')$. Fig. 4 demonstrates the steps of Lines 3–7 on an example, where Line 3 results in samples b'_0 to b'_4 (top panel).

Second, Line 4 scores each sample b'_i by the posterior probability $p'_i = p_\theta(a|b'_i)$, using the classifier parameter θ from

[5]We note that if the output of M is continuos, we must replace the probability in Eq. (8) by a probability density. For simplicity, we only discuss the case where the output of M is discrete.

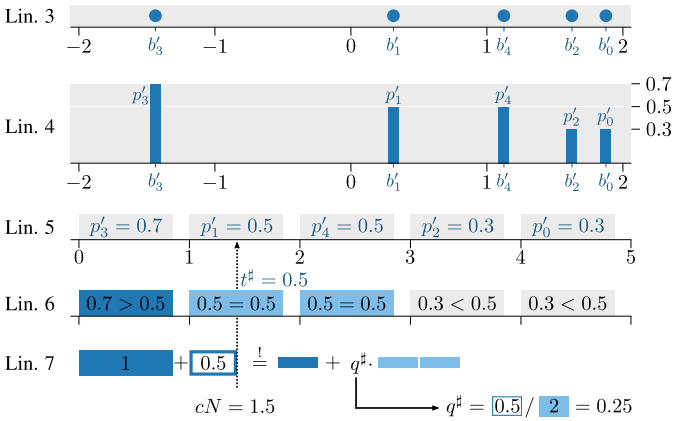

Fig. 4. Example run of Alg. 1 (Lines 3–7) for $c = 0.3$ and $N = 5$.

Line 2. In Fig. 4, this results in p'_0 to p'_4 (second panel).

Then, Lines 5–7 find parameters $t^\sharp, q^\sharp \in [0, 1]$ such that:

$$\sum_{i=0}^{N-1} \left[p'_i > t^\sharp\right] + q^\sharp \sum_{i=0}^{N-1} \left[p'_i = t^\sharp\right] \;=\; cN. \tag{10}$$

Intuitively, Eq. (10) ensures that $t^\sharp, q^\sharp$ (in expectation) cover a fraction c of all N samples as follows: they cover all samples whose probabilities p'_i lie above the threshold $t^\sharp$, and a fraction $q^\sharp$ of samples for which $p'_i = t^\sharp$.

To satisfy Eq. (10), Line 5 sorts the probabilities in descending order. In Fig. 4 (third panel), this results in

$$p''_0 = p'_3, \quad p''_1 = p'_1, \quad p''_2 = p'_4, \quad p''_3 = p'_2, \text{ and } p''_4 = p'_0.$$

Then, Line 6 selects $t^\sharp$ as $p''_{\min(\lfloor c \cdot N \rfloor,\ N-1)}$, which ensures that at most $c \cdot N$ probabilities lie below $t^\sharp$. In Fig. 4, we select $t^\sharp = p''_1 = 0.5$, since $\min(\lfloor c \cdot N \rfloor, N - 1) = 1$. Visually, this corresponds to using the value p'_i at position $cN = 1.5$ (the third panel in Fig. 4 shows the array indices i to the left of each value p'_i). The fourth panel in Fig. 4 illustrates the effect of selecting $t^\sharp = 0.5$: probabilities $p'_i > t^\sharp$ are definitively covered (colored in dark blue), while probabilities $p'_i = t^\sharp$ can be partially covered by controlling $q^\sharp$ (colored in light blue). We note that $\min(\cdot, N - 1)$ is technically necessary to avoid out-of-bounds array accesses to p'_N in case of $c = 1$.

Next, Line 7 selects $q^\sharp$ to satisfy Eq. (10). The last panel in Fig. 4 shows how selecting $q^\sharp = 0.25$ ensures that we cover a fraction of $cN = 1.5$ samples in total. We note that $q^\sharp$ is well-defined and can be interpreted as a probability (see App. D).

Finally, Lines 8–9 construct and return attack $\mathcal{S}_\theta^{t^\sharp, q^\sharp}$, which includes an output b if $p_\theta(a|b) \succeq (t^\sharp, q^\sharp)$ as defined in Eq. (9).

Selecting Sample Size N**.** We provide an empirical guideline for selecting N in order to maximize c-power up to a given error. This guideline is inspired by our theoretical results in Thm. 2 and Thm. 3, see §IV.

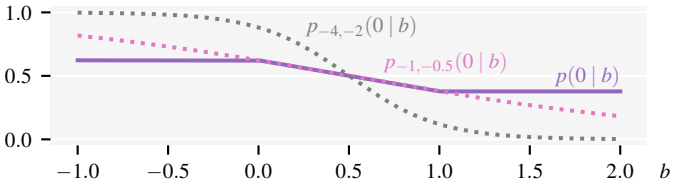

Fig. 5. Learning the posterior probability of $\mathcal{L}_{0.5}$ using logistic regression.

Guideline 1 (Empirical Precision)**.** Given neighboring $(a, a') \in \mathcal{N}$, $c \in (0, 1]$, and a desired precision $\omega > 0$, using

$$N = \max\left(\frac{2(1-c)}{\omega^2 \cdot c}, \frac{8(1-c)}{c}\right)\left(\text{erf}^{-1}(1-2\alpha)\right)^2$$

in Alg. 1 yields $\mathcal{S}_\theta^{t^\sharp, q^\sharp}$ for which, with probability $1 - \alpha$,

$$\mathcal{E}^{\geq c}(a, a', \mathcal{S}_\theta^{t^\sharp, q^\sharp}) \geq \max_{\mathcal{S} \in \mathbb{B} \to [0,1]} \mathcal{E}^{\geq c}(a, a', \mathcal{S}) - \omega,$$

where erf^{-1} is the inverse error function.

In our experiments, we use $c = 0.01$ and target $\omega = 0.005$ with $\alpha = 0.05$, yielding $N = 10.7 \cdot 10^6$ according to Guideline 1.

B. Learning the Posterior Probability

Next, we discuss how DP-Sniper learns the posterior probability $p(a|b)$.

Training. Given a family of classifiers $\{p_\theta(a|b)\}_{\theta \in \Theta}$ parameterized by θ, TrainClassifier selects a parameter θ such that $p_\theta(a|b)$ approximates the posterior probability $p(a|b)$ well.

First, it generates N_{train} samples from $M(a)$ (Line 11) and from $M(a')$ (Line 12), and concatenates them as training data $\mathcal{D}$ (Line 13). Then, Line 14 trains the classifier $p_\theta(a|b)$ distinguishing $M(a)$ from $M(a')$ on $\mathcal{D}$ (i.e., it picks θ) according to a family-specific optimization procedure.

Quality of Approximation. As we show in §IV, DP-Sniper returns an approximately optimal attack under the assumption that it learns the correct posterior probability $p_\theta(a|b) = p(a|b)$. While learning the exact posterior is unrealistic in practice, we can rely on learning a good approximation. Because DP-Sniper can conveniently generate an arbitrary amount of training data (in our experiments, we used $N_{\text{train}} = N$), the key factor determining the quality of the learned posterior is the choice of classifier family.

Families of Classifiers. In our evaluation (§VI), we consider two widely used classifier families. Specifically, we find that even basic logistic regression [21, §4.3.2] and small neural networks [21, §5] yield good results in practice, showing that the choice of classifier family is not a concern. Alternative valid choices include any family of classifiers computing a posterior probability, such as classification trees [21, §14.4] or different neural network architectures.

Logistic Regression. Logistic regression is a commonly used model, appealing due to its simplicity and efficient training [22, §11.2]. Its parameter space is $\Theta = \{\theta \in \mathbb{R}^n, \theta_0 \in \mathbb{R}\}$,

and it classifies $b \in \mathbb{R}^n$ according to $p_{\theta,\theta_0}(a|b) = \sigma\big(b^T\theta + \theta_0\big)$, where $\sigma\colon \mathbb{R} \to [0,1]$ is defined as $\sigma(x) = 1/(1+e^{-x})$.

Fig. 5 visualizes two possible logistic regression models approximating $p(0|b)$ for $\mathcal{L}_{0.5}$. While neither model is a perfect fit, both correctly capture the fact that smaller values of b are more likely to originate from $\mathcal{L}_{0.5}(0)$ than from $\mathcal{L}_{0.5}(1)$. Thus, intuitively, the attacks considered by DP-Sniper with logistic regression are equivalent to attacks of the form $b \leq t$.

In general, logistic regression yields linear decision boundaries, meaning that the attacks considered by DP-Sniper are equivalent to attacks of the form $b^T\theta \succeq (t,q)$. As a consequence, logistic regression is highly effective for many differentially private algorithms, which typically admit such a linear decision boundary. This is the case even for advanced mechanisms such as RAPPOR [18], which relies on hash functions and random bit flips.

To train our logistic regression model, we used regularized stochastic gradient descent optimization over 10 epochs with learning rate 0.3, momentum 0.3 and regularization weight 0.001 with binary cross entropy loss.

Neural Network. To demonstrate that the effectiveness of DP-Sniper is not limited to logistic regression, we also employ a simple neural network model. Neural networks are widely used in practice and have demonstrated to successfully learn complex distributions. Our architecture comprises two hidden layers with sizes 10 and 5 using ReLU activation functions, and a single output neuron using the sigmoid activation function σ. For training with binary cross entropy loss, we use the Adam [23] optimizer over 10 epochs with learning rate 0.1, $\beta_1 = 0.9, \beta_2 = 0.999$ and regularization weight 0.0001.

Feature Transformation. In both models, we apply standard feature transformations to the input. Namely, we (i) encode output values that indicate special outcomes (such as "abort" in SVT variants, see §VI-B) as additional boolean flags, and (ii) normalize all dimensions to ensure they exhibit the same mean and empirical variance.

C. DD-Search: Searching Witness

We now introduce DD-Search (Alg. 2), which combines DP-Sniper with a heuristic for selecting inputs $(a,a') \in \mathcal{N}$ to find a witness $(a,a',\boldsymbol{\mathcal{S}})$ with high power. DD-Search is designed to satisfy the following theorem.

Theorem 1 (DD-Search)**.** With probability $1-\alpha$, DD-Search returns a witness $(a,a',\boldsymbol{\mathcal{S}})$ and a lower bound $\underline{\mathcal{E}}$ for which $\underline{\mathcal{E}} \leq \mathcal{E}(a,a',\boldsymbol{\mathcal{S}})$, showing that M is $\underline{\mathcal{E}}$-DD.

We note that DD-Search is parametrized by α (used in Line 9), N_{check} (used in Line 4), and N_{final} (used in Line 5). In particular, modifying α is possible without modifying the sampling effort (N_{check} or N_{final}). However, selecting smaller α without increasing the sampling effort will lead to loose bounds on $\mathcal{E}$ (i.e., a lower value of $\underline{\mathcal{E}}$ in Thm. 1).

Searching Inputs. DD-Search leverages a heuristic to explore a set of neighboring inputs (a_i,a_i'), see GenerateInputs (Line 2 in Alg. 2). In our evaluation (§VI), we instantiate GenerateInputs by the pattern heuristic of [12] (see Tab. II). [6] For each input pair (a_i,a_i'), Line 3 then finds an approximately optimal attack by calling DP-Sniper.

Algorithm 2 DD-Search: Searching Witness $(a,a',\boldsymbol{\mathcal{S}})$.

Hyper-parameters: confidence $1-\alpha < 1$, sample sizes $0 < N_{\text{check}} \leq N_{\text{final}}$

1: **function** DD-Search($M\colon \mathbb{A} \to \mathbb{B}$)
2: **for** $(a_i,a_i') \in$ GenerateInputs() **do** ▷ select inputs by heuristic
3: $\boldsymbol{\mathcal{S}}_i \leftarrow$ DP-Sniper(M, a_i, a_i') ▷ select attack by Alg. 1
4: $i \leftarrow \arg\max_{i=0}^{N_{\text{input}}-1}$ Estimate$\mathcal{E}$(M, a_i, a_i', $\boldsymbol{\mathcal{S}}_i$, N_{check})
5: **return** $(a_i,a_i',\boldsymbol{\mathcal{S}}_i)$ and LowerBound$\mathcal{E}$(M, a_i, a_i', $\boldsymbol{\mathcal{S}}_i$, N_{final})
6: **function** Estimate$\mathcal{E}$(M, a, a', $\boldsymbol{\mathcal{S}}$, N) ▷ estimate $\mathcal{E}(a,a',\boldsymbol{\mathcal{S}})$
7: **return** $\ln\left(\hat{P}^N_{M(a)\in\boldsymbol{\mathcal{S}}}\right) - \ln\left(\hat{P}^N_{M(a')\in\boldsymbol{\mathcal{S}}}\right)$
8: **function** LowerBound$\mathcal{E}$(M, a, a', $\boldsymbol{\mathcal{S}}$, N) ▷ lower bound $\mathcal{E}(a,a',\boldsymbol{\mathcal{S}})$
9: **return** $\ln\left(\underline{P}^{N,\alpha/2}_{M(a)\in\boldsymbol{\mathcal{S}}}\right) - \ln\left(\bar{P}^{N,\alpha/2}_{M(a')\in\boldsymbol{\mathcal{S}}}\right)$

Estimating Power. Next, Line 4 selects the best witness according to an estimate using N_{check} samples, following the approach from Eq. (5). Note that DD-Search estimates $\mathcal{E}(a,a',\boldsymbol{\mathcal{S}})$ and not $\mathcal{E}^{\geq c}(a,a',\boldsymbol{\mathcal{S}})$. This is not a problem as DP-Sniper never returns an attack $\boldsymbol{\mathcal{S}}$ for which $\Pr[M(a) \in \boldsymbol{\mathcal{S}}]$ or $\Pr[M(a') \in \boldsymbol{\mathcal{S}}]$ is significantly smaller than c.

Finally, Line 5 returns a lower bound on $\mathcal{E}(a,a',\boldsymbol{\mathcal{S}})$ using N_{final} samples. This bound is computed by LowerBound$\mathcal{E}$, which is constructed to satisfy Thm. 1 as follows.

Proof of Thm. 1. LowerBound$\mathcal{E}$ computes a lower bound $\underline{P}^{N,\alpha/2}_{M(a)\in\boldsymbol{\mathcal{S}}}$ on $\Pr[M(a) \in \boldsymbol{\mathcal{S}}]$ and an upper bound $\bar{P}^{N,\alpha/2}_{M(a')\in\boldsymbol{\mathcal{S}}}$ on $\Pr[M(a') \in \boldsymbol{\mathcal{S}}]$, which both hold except with probability $\alpha/2$ each. By to the union bound, the probability that either bound does not hold is at most $\alpha/2 + \alpha/2 = \alpha$. Hence, with probability $1-\alpha$, both bounds hold and Line 9 returns a lower bound on $\mathcal{E}(a,a',\boldsymbol{\mathcal{S}})$. □

To compute bounds $\underline{P}^{N,\alpha/2}_{M(a)\in\boldsymbol{\mathcal{S}}}$ and $\bar{P}^{N,\alpha/2}_{M(a')\in\boldsymbol{\mathcal{S}}}$ in Line 9, our implementation leverages the Clopper-Pearson interval [24].

Hyper-parameters. In our experiments (§VI), we use the hyper-parameters $N_{\text{check}} = N = 10.7\cdot10^6$, and $N_{\text{final}} = 2\cdot10^8$.

Underestimating Differential Distinguishability. As shown in Thm. 1, DD-Search never overestimates differential distinguishability for an analyzed algorithm. In fact, Thm. 1 even holds for arbitrary instantiations of GenerateInputs and DP-Sniper. In particular, it also holds if the classifier learned by DP-Sniper is not accurate.

However, DD-Search may return a witness with suboptimal power and hence underestimate differential distinguishability. Finding a maximally powerful witness requires that (i) GenerateInputs produces inputs admitting attacks with maximal power, and (ii) DP-Sniper learns the correct posterior probability $p_\theta(a|b) = p(a|b)$.

[6] We note that this instantiation of DD-Search does *not* run in time $\Omega(|\mathbb{A}|^2)$, as it only investigates the specific patterns (a_i,a_i') listed in Tab. II.

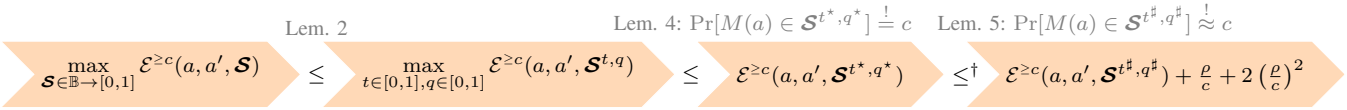

Fig. 6. Steps to derive optimality of DP-Sniper's attack, where the last inequality is subject to the additional conditions from Lem. 5 (indicated by †).

These assumptions are typically not realistic. However, as we demonstrate in our evaluation (§VI), we obtain good results when (i) GenerateInputs uses a heuristic for selecting inputs admitting powerful attacks, and (ii) DP-Sniper learns a good approximation $p_\theta(a|b) \approx p(a|b)$ of the posterior probability.

IV. Formal Guarantees

Next, we discuss the formal guarantees of DP-Sniper. Specifically, we provide a step-by-step derivation of Thm. 2.

Theorem 2 (Approximatly Optimal Attack)**.** For all neighboring $(a, a') \in \mathcal{N}$, the attack $\boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}$ returned by Alg. 1 satisfies

$$\mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}) \geq \max_{\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]} \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}) - \frac{\rho}{c} - 2\left(\frac{\rho}{c}\right)^2$$

with probability at least $1 - \alpha \in [0, 1)$, for $N > 0$, $c \in (0, 1]$, $\rho = \sqrt{\frac{\ln(2/\alpha)}{2N}}$, and assuming $\frac{\rho}{c} \leq \frac{1}{2}$.

Accuracy of Classification. In Thm. 2, $\boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}$ denotes the threshold attack $p(a|\cdot) \succeq (t^\sharp, q^\sharp)$ that uses the perfect classifier $p(a|b) = \Pr[A = a \mid M(A) = b]$.

In practice, $p_\theta(a|b)$ is only an approximation of $p(a|b)$. Still, due to Thm. 1, using DP-Sniper within DD-Search allows us to demonstrate (potentially sub-optimal) differential distinguishability, regardless of the accuracy of $p_\theta(a|b)$.

Proof of Thm. 2. Fig. 6 shows the necessary steps to prove Thm. 2 and serves as an outline for the following sub sections. First, we show that parametrized threshold attacks $\boldsymbol{\mathcal{S}}^{t,q}$ are as powerful as arbitrary randomized attacks $\boldsymbol{\mathcal{S}}$ (Lem. 2 in §IV-A). Next, we demonstrate that ensuring $\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right] = c$ yields optimal parameters $t^\star, q^\star$ (Lem. 4 in §IV-B). Because selecting optimal parameters $t^\star, q^\star$ is intractable in practice, we conclude by selecting approximately optimal parameters $t^\sharp, q^\sharp$ with $\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}] \approx c$ (Lem. 5 in §IV-C). □

Motivation for Guideline 1. Guideline 1 is inspired by Thm. 2 as follows. First, we ignore the higher order term $2\left(\frac{\rho}{c}\right)^2$ from Thm. 2, as it is negligible compared to $\frac{\rho}{c}$ if $\rho \ll c$.

As we will see later, ρ bounds the estimation error of $\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}]$ and $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]$ using the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality [25], which is overly conservative in practice. Hence, for Guideline 1, we further replace ρ by the (smaller) Wald interval:

$$\rho = \sqrt{\frac{2 \cdot c \cdot (1-c)}{N}} \text{erf}^{-1}(1 - 2\alpha).$$

This empirical bound is inspired by the Central Limit Theorem [26, Prop. 2.18] and is quite precise for sufficiently large N, and holds because $\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}]$ and $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]$ are roughly equal to c.

We obtain Guideline 1 by solving for the smallest N that satisfies the resulting constraints.

A. Restriction to Threshold Attacks

We now show that we can restrict our attention to the family of threshold attacks $\{\boldsymbol{\mathcal{S}}^{t,q}\}_{t,q \in [0,1]}$ introduced in §III-A, which significantly reduces the search space of possible attacks while still admitting optimal attacks.

Intuition. Intuitively, in order to maximize $\mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}})$, the probability that $\boldsymbol{\mathcal{S}}$ incudes $b \in \mathbb{B}$ should be high if b is likely to be an output from $M(a)$, i.e., if $p(a|b)$ is high.

Example 6. The purple line in Fig. 2 (bottom panel) shows $p(a|b)$ for $\mathcal{L}_{0.5}$. It is maximal for $b \leq 0$, decreases for b between 0 and 1, and then remains constant.

Fig. 2 also illustrates the strong connection between $p(a|b)$ and $\mathcal{E}^{\geq c}(0, 1, (-\infty, b])$: if $(-\infty, b]$ only contains values b' for which $p(a|b')$ is maximal (which is the case for $b \leq 0$), $\mathcal{E}^{\geq c}(0, 1, (-\infty, b])$ is also maximal, unless b is too small. In contrast, if $(-\infty, b]$ also contains values b' with sub-optimal $p(a|b')$, then $\mathcal{E}^{\geq c}(0, 1, (-\infty, b])$ decreases.

Optimality. Threshold attacks $\boldsymbol{\mathcal{S}}^{t,q}$ formalize the above intuition by (i) always selecting elements b whose posterior probability $p(a|b)$ lies above t, (ii) probabilistically selecting elements whose posterior probability is tied with t, and (iii) never selecting elements with lower posterior probability. Lem. 2 demonstrates that this approach is optimal.

Lemma 2 (Threshold Attacks Optimal)**.** For all $(a, a') \in \mathcal{N}$,

$$\max_{\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]} \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}) \leq \max_{t \in [0,1], q \in [0,1]} \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t,q}).$$

In order to prove Lem. 2, we prove the stronger Lem. 3.

Lemma 3. For all $(a, a') \in \mathcal{N}$ and any $\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0, 1]$, there exists a threshold attack $\boldsymbol{\mathcal{S}}^{t,q}$ with

$$\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}\right] \geq \Pr[M(a) \in \boldsymbol{\mathcal{S}}] \text{ and} \tag{11}$$

$$\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\right] = \Pr[M(a') \in \boldsymbol{\mathcal{S}}]. \tag{12}$$

Proof of Lem. 3. We first note that there exist t, q which satisfy Eq. (12), as shown by Lem. 12 in App. G. We illustrate this visually on our running example $\mathcal{L}_{0.5}$ in Fig. 8. In particular, starting from $t \approx 0.62$ and $q = 0$ yields $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}] = 0$. From there, we can gradually increase the probability to obtain any value p', by either (i) increasing q or (ii) decreasing t and resetting q to 0, as shown in Fig. 8.

Second, Lem. 13 in App. G uses the Neyman-Pearson lemma (Lem. 9 in App. F) to show that satisfying Eq. (12) implies that we also satisfy Eq. (11). We note that the Neyman-

$$\underbrace{\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]}_{\text{(a)}} = \underbrace{\Pr_{P'}[p(a|M(a')) \succeq (t,q)]}_{\text{(b)}} \approx \underbrace{\frac{1}{N}\sum_{i=0}^{N-1} \Pr_{p'_i \sim P'}[p'_i \succeq (t,q)]}_{\text{(c)}} \xrightarrow[t^\sharp, q^\sharp]{\text{pick}} \underbrace{\frac{1}{N}\sum_{i=0}^{N-1} \Pr[p'_i \succeq (t^\sharp, q^\sharp)]}_{\text{(d)}} = c$$

Fig. 7. Selecting approximately optimal $t^\sharp, q^\sharp$.

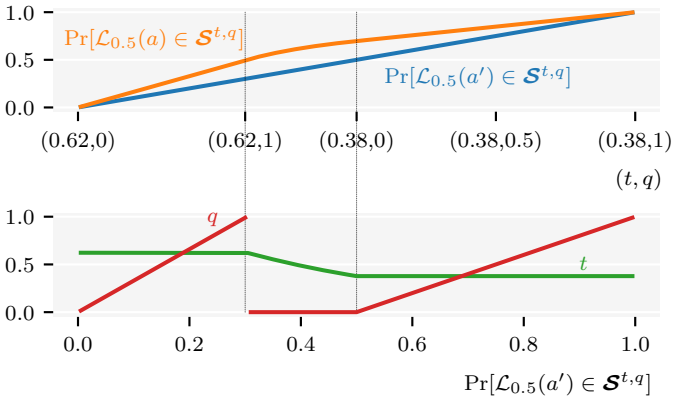

Fig. 8. Relationship of $\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}]$, $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]$, and (t,q) for $\mathcal{L}_{0.5}$. The top panel shows $\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}]$ and $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]$ for different choices of (t,q), while the bottom panel shows the appropriate (t,q) required to obtain any given value of $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]$.

Pearson lemma is often used in the context of differential privacy, e.g., in [27, §IV-A], [28, §9.1], and [29]. □

Proof of Lem. 2. Lem. 3 is sufficient to prove Lem. 2, since it shows there exists t, q for which:

$$\begin{aligned}
&\mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}) \\
&= \ln^{\geq c}\big(\underbrace{\Pr[M(a) \in \boldsymbol{\mathcal{S}}]}\big) - \ln^{\geq c}\big(\underbrace{\Pr[M(a') \in \boldsymbol{\mathcal{S}}]}\big) \\
&\qquad \leq \Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}] \text{ by Eq. (11)} \quad = \Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}] \text{ by Eq. (12)} \\
&\leq \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t,q}).
\end{aligned}$$

□

B. Optimal Attack

As can be seen in Fig. 8, every choice of t, q yields a different trade-off between $\Pr[M(a) \in \boldsymbol{\mathcal{S}}]$ (which should be maximized) and $\Pr[M(a') \in \boldsymbol{\mathcal{S}}]$ (which should be minimized). Among all possible values for t, q, Lem. 4 shows how to find the optimal choice of $t^\star, q^\star$.

Lemma 4 (Optimal Attack)**.** For $c \in (0, 1]$ and all neighboring inputs $(a, a') \in \mathcal{N}$, there exist $(t^\star, q^\star)$ that satisfy

$$\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right] = c. \tag{13}$$

For any such $t^\star, q^\star$, we have

$$\max_{t\in[0,1], q\in[0,1]} \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t,q}) \leq \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t^\star,q^\star}). \tag{14}$$

Proof. Parameters $t^\star, q^\star$ that satisfy Eq. (13) exist by a reasoning analogous to our proof of Lem. 3.

Next, we provide the intuition for proving Eq. (14), formalized in App. H. We proceed by case distinction. If $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}] < c$, then replacing $\boldsymbol{\mathcal{S}}^{t,q}$ by $\boldsymbol{\mathcal{S}}^{t^\star,q^\star}$ in

$$\ln^{\geq c}\left(\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}\right]\right) - \ln^{\geq c}\left(\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\right]\right)$$

increases the minuend (as $\boldsymbol{\mathcal{S}}^{t^\star,q^\star}$ contains more elements than $\boldsymbol{\mathcal{S}}^{t,q}$) but not the subtrahend (as $\ln^{\geq c}$ is constant for inputs below c). Otherwise, if $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}] > c$, then replacing $\boldsymbol{\mathcal{S}}^{t,q}$ by $\boldsymbol{\mathcal{S}}^{t^\star,q^\star}$ increases the minuend more than the subtrahend, as it removes the "worst" elements from $\boldsymbol{\mathcal{S}}^{t,q}$. □

C. Approximately Optimal Attack

Based on Lem. 4, we would like DP-Sniper to find $t^\star, q^\star$ that satisfy Eq. (13):

$$\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right] = c.$$

Unfortunately, finding $(t^\star, q^\star)$ which satisfy this equation exactly is not possible in practice, as $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]$ can generally only be estimated. Therefore, a more tractable goal is selecting $(t^\sharp, q^\sharp)$ that approximately satisfy Eq. (13).

Fig. 7 illustrates how DP-Sniper achieves this goal. First, we rewrite $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]$ according to Eq. (9). Second, we interpret $p(a|M(a'))$ as a random variable P' and estimate $\Pr[P' \succeq (t,q)]$ using samples $p'_0, \ldots, p'_{N-1}$ from P'. Finally, we pick $t^\sharp, q^\sharp$ for which our estimate yields exactly c. Note that this is achieved by DP-Sniper in Lines 6–7, as Eq. (10) is equivalent to $\frac{1}{N}\sum_{i=0}^{N-1} \Pr[p'_i \succeq (t^\sharp, q^\sharp)] = c$.

Lem. 5 formalizes the obtained optimality guarantees.

Lemma 5 (Approximate Optimal Attack)**.** With probability at least $1 - \alpha$,

$$\mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}) \geq \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t^\star,q^\star}) - \tfrac{\rho}{c} - 2\left(\tfrac{\rho}{c}\right)^2,$$

for $\rho = \sqrt{\frac{\ln(2/\alpha)}{2n}}$ and assuming $\frac{\rho}{c} \leq \frac{1}{2}$.

Proof Sketch. Our proof in App. J relies on the DKW inequality (App. I), which indicates that with probability $1 - \alpha$, the estimate in Fig. 7c is simultaneously accurate (up to a precision of ρ) for all possible choices of t, q. We can then prove Lem. 5 by applying multiple insights about the relationship of $\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}]$ and $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]$. □

We note that the regret $\frac{\rho}{c} + 2\left(\frac{\rho}{c}\right)^2$ in Lem. 5 is almost optimal. Specifically, if our probability estimates achieve an accuracy of $\pm\rho$, we should expect a regret of roughly $\frac{1}{2}\frac{\rho}{c}$, as we show in App. E.

From this, we conclude that we cannot significantly improve Lem. 5, Thm. 2, or Guideline 1.

TABLE II
Input patterns used in StatDP, from [12, Tab. 1].

Category	a	a'
One Above	$[1,1,1,1,1]$	$[2,1,1,1,1]$
One Below	$[1,1,1,1,1]$	$[0,1,1,1,1]$
One Above Rest Below	$[1,1,1,1,1]$	$[2,0,0,0,0]$
One Below Rest Above	$[1,1,1,1,1]$	$[0,2,2,2,2]$
Half Half	$[1,1,1,1,1]$	$[0,0,0,2,2]$
All Above & All Below	$[1,1,1,1,1]$	$[2,2,2,2,2]$
X Shape	$[1,1,0,0,0]$	$[0,0,1,1,1]$

V. Comparison Baseline

To put our results in perspective, we decided to use StatDP [12] as a baseline. It reports state-of-the-art results and, unlike many other tools detecting differential distinguishability, is not limited by fundamental assumptions (see §VII).

Unfortunately, StatDP (§V-A) does not directly yield a suitable baseline: it does not address the exact same problem statement as DD-Search (§V-B) and its implementation exhibits multiple issues (§V-C). In the following, we discuss both of these challenges and how we addressed them by extending StatDP to a system we refer to as StatDP-Fixed.

A. StatDP

At a high level, StatDP iterates over a search space of predetermined witnesses $(a, a', \mathcal{S})$, and applies a statistical test to determine whether they demonstrate ξ_0-DD for a given ξ_0.

When searching for input pairs (a, a'), StatDP considers the simple patterns shown in Tab. II. Its attack search space is more advanced: an attack $\mathcal{S}$ applies a post-processing function $f\colon \mathbb{B} \to \mathbb{R}^n$ to the output $M(A)$ and checks if the result $f(M(A))$ lies inside an n-dimensional hypercube. Formally, the attacks considered in StatDP can be written as:

$$\mathcal{S} = \{b \,|\, f(b) \in [l_1, u_1] \times \cdots \times [l_n, u_n]\}\,, \tag{15}$$

for $l_i, u_i \in \mathbb{R} \cup \{-\infty\}$ and post-processing $f \in \mathcal{F}$. StatDP focuses on mechanisms whose output b is a list. Amongst others, the collection $\mathcal{F}$ of possible post-processings includes the length of b, numerical statistics about b such as the mean and maximum, and the Hamming distance between b and a reference output $\tilde{M}(a)$ computed by evaluating M on a without using any noise ($\mathcal{F}$ implicitly depends on a). See [12, §4.3] for details about the collection $\mathcal{F}$.

B. Different Problem Statement

The problem statements addressed by our work and StatDP are different: StatDP tests for ξ_0-DD with a fixed value of ξ_0, while DD-Search searches for a witness with maximal power. As shown in Fig. 9, StatDP guarantees that when it finds a witness $(a, a', \mathcal{S})$, this witness demonstrates ξ_0-DD (testing). When StatDP does not find such a witness, it reports a failure. In contrast, DD-Search always returns both ξ and a witness $(a, a', \boldsymbol{\mathcal{S}})$ demonstrating ξ-DD (maximizing power).

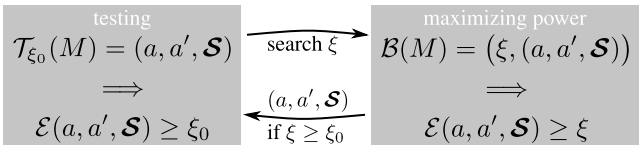

Fig. 9. Comparison of testing to maximizing power.

Conversion. As shown in Fig. 9 (middle), there exists a conversion from any testing algorithm $\mathcal{T}_{\xi_0}$ (left) to an algorithm $\mathcal{B}$ maximizing power (right) by searching for the largest ξ for which the testing algorithm finds a witness. This approach is in line with the original evaluation of StatDP [12, Fig. 2].

In StatDP-Fixed, we use a binary search to find the largest ξ for which StatDP's statistical test succeeds with high confidence (p-value ≤ 0.05), up to a precision of 0.001 for ξ. To determine an upper bound on ξ to be used by the binary search, StatDP-Fixed uses exponential search.

We note that it is also possible to convert any algorithm maximizing power into a testing algorithm $\mathcal{T}_{\xi_0}$ by returning $(a, a', \boldsymbol{\mathcal{S}})$ if $\xi \geq \xi_0$, and reporting a failure otherwise. Thus, DD-Search can directly solve the problem statement of StatDP.

C. Issues with StatDP Implementation

Unfortunately, the StatDP implementation [30] exhibits multiple issues discussed next. In particular, it (i) does not follow the description in [12], (ii) is unsound in some cases, and (iii) produces unstable results for some mechanisms.

Hard-Coded Post-Processing. The StatDP implementation does not search for a post-processing $f \in \mathcal{F}$ as described in [12, §4.3], but instead hard-codes an appropriate post-processing f for each algorithm evaluated in its publication [12, §5]. A request for clarification on this discrepancy between publication and implementation did not receive a response up to this day. [7]

As a consequence, the existing implementation of StatDP cannot be used to reproduce the results reported in its publication. For example, StatDP crashes when provided a correct implementation of SVT3 (see §VI-B, this is iSVT4 in [12]) that does not include a post-processing, as SVT3 produces outputs of varying length. If we pad the output of SVT3 to its maximum output length 10, StatDP also crashes—it requires more than 500 GB of memory, because its search space is exponentially large in the output length.

In order to enable the usage of StatDP described in its publication, StatDP-Fixed includes a search over the possible post-processings described in [12, §4.3]. As a consequence, the runtimes we report in this work are incomparable to the runtimes reported in [12, Tab. 3], which assumed a fixed post-processing. We stress that this is not a shortcoming of this work, but an artifact of the incorrect evaluation from [12].

Unsoundness. Moreover, the results reported by StatDP are sometimes unsound, meaning that StatDP may claim that a witness $(a, a', \mathcal{S})$ demonstrates ξ_0-DD with high confidence,

[7] https://github.com/cmla-psu/statdp/issues/4

even if this is not the case. Therefore, StatDP-Fixed includes a confirmation phase that validates the results reported by StatDP. Specifically, we employ the same lower bound estimation as DD-Search (Line 5 in Alg. 2).

Fig. 10 (§VI-C) demonstrates the unsoundness of StatDP, where the unfilled bars indicate the power of the witnesses according to StatDP, and the filled bars indicate the true power according to StatDP-Fixed. For example, StatDP claims 0.2-DD for SVT1 with p-value ≤ 0.05, even though SVT1 is known to be 0.1-DP [31].

These findings indicate that the statistical test employed by StatDP is not always sound. Possible reasons include an incorrect design or implementation of the test, or numerical issues. This issue typically occurs when StatDP returns $(a, a', \mathcal{S})$ for which $\Pr[M(a) \in \mathcal{S}]$ and $\Pr[M(a') \in \mathcal{S}]$ are small—a situation we explicitly avoid in our problem statement (§II).

Instability. Finally, even after addressing the above issues, StatDP-Fixed is highly unstable for some algorithms, reporting vastly different results in independent runs.

In our evaluation (§VI-C), we therefore perform two independent runs of StatDP-Fixed on all algorithms. For instance, while the first run of StatDP-Fixed demonstrates 0.471-DD for OneTimeRAPPOR, the second run only finds 0.060-DD. Similar instabilities were observed for RAPPOR, SVT3 and SVT34Parallel (see Fig. 10). We conjecture these to be artifacts of the power overestimation by StatDP.

VI. Evaluation

In the following, we evaluate DD-Search by comparing it to StatDP-Fixed (see §V), using the default parameters suggested by StatDP. As DD-Search mainly relies on DP-Sniper, this approach also evaluates DP-Sniper implicitly. We ran all experiments on a machine with 500 GB RAM and 128 cores at 1.2 GHz. We note that our implementation never required more than 16.2 GB of memory.

A. Implementation

We implemented DP-Sniper and DD-Search in Python, using PyTorch[8] for instantiating and training classifiers.

Input Search. For GenerateInputs in Line 2 of Alg. 2, we use the same patterns as StatDP (see Tab. II). Unlike StatDP, we also include flipped inputs in our search: If (a, a') is a pattern included in StatDP, we include both (a, a') and (a', a). This is not required in StatDP, as StatDP implicitly assumes a symmetric neighborhood.

Vectorization and Parallelization. Our implementations of the evaluated algorithms rely on NumPy[9] for vectorization. Further, our implementation of DD-Search exploits the fact that the loop in Line 2 of Alg. 2 can be trivially parallelized.

[8]https://github.com/pytorch/pytorch

[9]https://github.com/numpy/numpy

TABLE III
Evaluated algorithms with their neighborhoods and DP guarantees.

Algorithm	$\mathcal{N}$	ϵ
NoisyHist1 [12, Alg. 9]	$\|\cdot\|_1$	0.1
NoisyHist2 [12, Alg. 10]	$\|\cdot\|_1$	10
ReportNoisyMax1 [12, Alg. 5]	$\|\cdot\|_\infty$	0.1
ReportNoisyMax2 [12, Alg. 6]	$\|\cdot\|_\infty$	0.1
ReportNoisyMax3 [12, Alg. 7]	$\|\cdot\|_\infty$	∞
ReportNoisyMax4 [12, Alg. 8]	$\|\cdot\|_\infty$	∞
SVT1 [31, Alg. 1]	$\|\cdot\|_\infty$	0.1
SVT3 [31, Alg. 3]	$\|\cdot\|_\infty$	∞
SVT4 [31, Alg. 4]	$\|\cdot\|_\infty$	0.18
SVT5 [31, Alg. 5]	$\|\cdot\|_\infty$	∞
SVT6 [31, Alg. 6]	$\|\cdot\|_\infty$	∞
LaplaceMechanism $\mathcal{L}_{0.1}$ Ex. 1	$\|\cdot\|_1$	0.1
LaplaceParallel $a \mapsto \big(\mathcal{L}_{0.005}(a), \dots, \mathcal{L}_{0.005}(a)\big)$	$\|\cdot\|_1$	0.1
NumericalSVT [10, Fig. 10]	$\|\cdot\|_\infty$	0.1
OneTimeRAPPOR [18, Steps 1-2]	$\|\cdot\|_1$	0.8
PrefixSum [10, App. C.3]	$\|\cdot\|_\infty$	0.1
RAPPOR [18, Steps 1-3]	$\|\cdot\|_1$	0.4
SVT2 [31, Alg. 2]	$\|\cdot\|_\infty$	0.1
SVT34Parallel $a \mapsto \big(\text{SVT3}(a), \text{SVT4}(a)\big)$	$\|\cdot\|_\infty$	∞
TruncatedGeometric [32, Ex. 2.2], impl. [33, Alg. 4.8]	$\|\cdot\|_1$	0.12

B. Evaluated Algorithms

We evaluated the tools on all algorithms in Tab. III, which includes widely used differentially private algorithms as well as variations of them. The second column indicates the used neighborhood, where $\|\cdot\|_p$ denotes the p-norm neighborhood $\mathcal{N} = \{(a, a') \mid \|a - a'\|_p \leq 1\}$.

Only the algorithms up to and including SVT6 were used in the original evaluation of StatDP [12]. This will be relevant later, where we demonstrate that StatDP-Fixed does not generalize well to new algorithms.

NoisyHist1–2 create a noisy version of a given histogram, using parameters $\epsilon_0 = 0.1$ (the targeted DP guarantee) and input length 5 as in [12]. ReportNoisyMax1–4 are variants determining the maximum of an array, where we have instantiated the parameters in [12] by $\epsilon_0 = 0.1$ (targeted DP) and input length 5. Algorithms SVT1–6 are variants of an algorithm for detecting which elements of an array lie above a given threshold T. In all variants, we instantiated the parameters by $\epsilon = 0.1$ (targeted DP), $\Delta = 1$ (maximum distance of inputs), $c = 1$ (maximum number of responses), input length 10, and threshold $T = 1$ (except $T = 0.5$ for SVT1) following the notation in [31]. We have selected these parameters to remain consistent with the evaluation of StatDP [12].

LaplaceParallel applies the Laplace mechanism to the same output 20 times, testing whether the evaluated tools can handle parallel composition—a key operation in differential privacy. NumericalSVT is an interesting variant of SVT which outputs numerical query answers when the query answer is large. OneTimeRAPPOR collects statistics over categorical data, and was instantiated with $k = 20$, $h = 4$, and $f = 0.95$. This algorithm is particularly interesting as it involves complex operations such as evaluating hash functions, which cannot be handled by non-black-box tools such as DiPC [14]. PrefixSum computes all prefix sums of a given array, where we used arrays of length 10, and parameter $\epsilon = 0.1$. RAPPOR is a variant of

OneTimeRAPPOR that provides stronger privacy guarantees, and we instantiated it with $k = 20$, $h = 4$, $f = 0.75$, $q = 0.55$, and $p = 0.45$. SVT34Parallel is the parallel composition of SVT3 and SVT4. Finally, TruncatedGeometric obfuscates its input by adding noise sampled from a two-sided geometric distribution.

Guaranteed Level of Differential Privacy. The last column in Tab. III shows the level of differential privacy guaranteed by each algorithm according to existing literature (as cited in Tab. III). We note that for some incorrect algorithms (e.g., SVT5), the guarantees are weaker than originally intended (i.e., the actually guaranteed ϵ is higher than the targeted ϵ).

Unfortunately, it is typically unclear if these guarantees are tight, or if an algorithm satisfies ϵ-DP for smaller ϵ than indicated in Tab. III. Our results (discussed in §VI-C) indicate the guarantees are often tight, but not always. For example, NoisyHist1 is known to be 0.1-DP, and we demonstrate 0.098-DD. In contrast, RAPPOR is 0.4-DP, but we only demonstrate 0.301-DD. Such a gap may either indicate that DD-Search does not find good witnesses, or that RAPPOR satisfies stronger differential privacy guarantees than previously known. For example, the differential distinguishability of ReportNoisyMax3 is only ∞ for inputs of arbitrary size. For inputs of size 5 (as investigated in §VI-C), a more precise analysis shows that ReportNoisyMax3 satisfies 0.25-DP, meaning that our demonstration of 0.249-DD is actually very tight.

We note that the values of ϵ in Tab. III assume idealized implementations—as we demonstrate in §VI-D, floating point vulnerabilities may invalidate these theoretical guarantees.

C. Results

Fig. 10 shows the lower bounds on the power of the witnesses returned by DD-Search with logistic regression and neural network classifiers, and StatDP-Fixed. Because the results of StatDP-Fixed are unstable (see §V-C), we show the results of two independent runs. The solid bars in Fig. 10 display the lower bounds computed according to Line 8 in Alg. 2. StatDP-Fixed throws an exception for NumericalSVT, which is due to an exception raised in StatDP.

Detected Distinguishability. On algorithms that were already used to evaluate StatDP in [12] (top plot in Fig. 10), DD-Search and StatDP-Fixed perform similarly in most cases. In some cases, DD-Search can demonstrate ξ-DD for substantially higher ξ, specifically for ReportNoisyMax3–4 and SVT1. For two cases with extremely high differential distinguishability, StatDP-Fixed performs better. However, as both tools demonstrate these algorithms are clearly not ϵ-DP for the target parameter $\epsilon = 0.1$, the exact level of differential distinguishability is less relevant in these cases. For SVT6, StatDP-Fixed finds a witness whose power is roughly 20% higher than the one returned by DD-Search. DD-Search cannot find this witness as it induces probabilities below $c = 0.01$. We note that decreasing c and increasing N would likely yield an equally powerful witness, but at the cost of increased runtime.

The fundamental improvement of DD-Search over StatDP-Fixed becomes apparent when evaluating both tools on new benchmark algorithms (bottom plot in Fig. 10). On these, DD-Search performs consistently and substantially better, by a factor of up to 7.9 (compared to 1^{st} run) resp. 12.4 (2^{nd} run). This suggests that the search space of StatDP-Fixed overfits to the examples StatDP was originally evaluated on.

Both classifier families for DD-Search yield similar powers. The more complex neural network model only significantly outperforms logistic regression for PrefixSum. We conclude that the choice of classifier family is not essential.

Runtime. Fig. 11 compares the runtimes of DD-Search and StatDP-Fixed. It demonstrates that DD-Search with logistic regression is almost always significantly faster than StatDP-Fixed, on average by a factor of 15.5. Due to higher training costs, using neural networks in DD-Search is more expensive. Still, compared to to StatDP-Fixed, this combination admits an average speedup of 3.4. We note that the reported runtimes are generous towards StatDP-Fixed, as they do not include computing a lower bound on the power of its returned witness.

The runtime of StatDP-Fixed varies significantly between algorithms due to the different number of binary search evaluations, valid input patterns, and applicable post-processings. In contrast, the runtime of DD-Search is more predictable.

As discussed in §V-B, StatDP-Fixed applies multiple ξ-DD tests in a binary search. For some algorithms, running a single such test is much faster than running DD-Search (e.g., 5.5 times faster for LaplaceMechanism), while it is much slower for others (e.g., 4.5 times slower for LaplaceParallel).

D. Floating Point Vulnerabilities

In order to highlight the versatility of our approach, we also used it to demonstrate differential distinguishability of seemingly differentially private algorithms by exploiting vulnerabilities that arise from floating-point arithmetic.

Bit Pattern Features. Past work [34] discovered floating-point vulnerabilities in common implementations of the Laplace mechanism. However, exploiting these required a tedious analysis of the effect of floating-point imprecisions on the algorithm's output—we refer to [34] for a thorough and insightful discussion. In contrast, with DP-Sniper, we can simply conjecture that the bit patterns of the output may reveal information about the input, and use an appropriate family of classifiers that exploits this information.

To this end, we extended the neural network classifier family described in §III-B by a pre-processing step extracting the 64 bit IEEE 754 floating point representation of the mechanism output. In particular, we use an input layer of size 64 (one neuron per bit), followed by two hidden layers with sizes 10 and 5 using ReLU activation functions. As an example, for the mechanism output -1.5 we would feed its bit-representation $10111111111110\cdots0$ to the first layer.

The Laplace Mechanism is Differentially Distinguishable. We ran DD-Search with this classifier family to investigate our implementation of $\mathcal{L}_{0.1}$, which uses NumPy's implementation of Laplace noise. To generate input pairs, we again used the StatDP patterns as described in §VI-A. In this case, the patterns yield four input paris: $(1, 0), (0, 1), (1, 2), (2, 1)$.

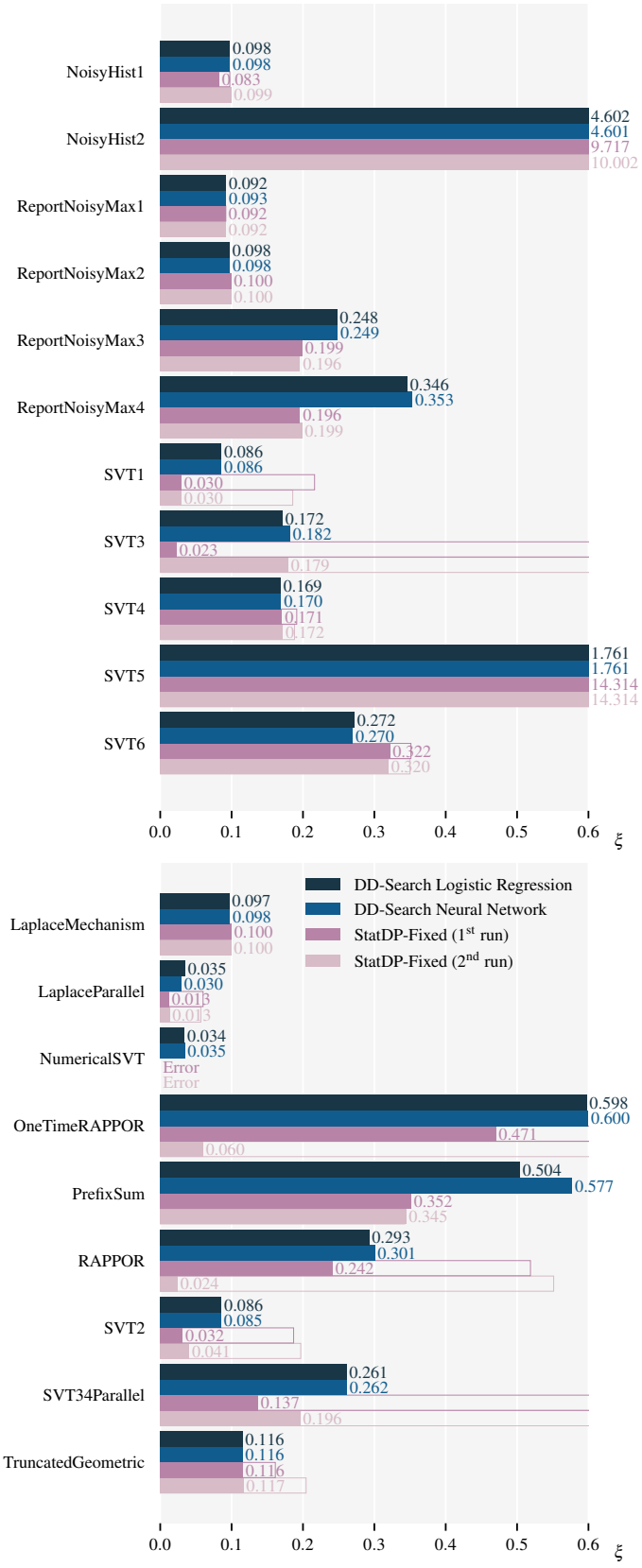

Fig. 10. Value ξ of ξ-DD demonstrated by DD-Search (this work) and StatDP-Fixed, where higher values are better.

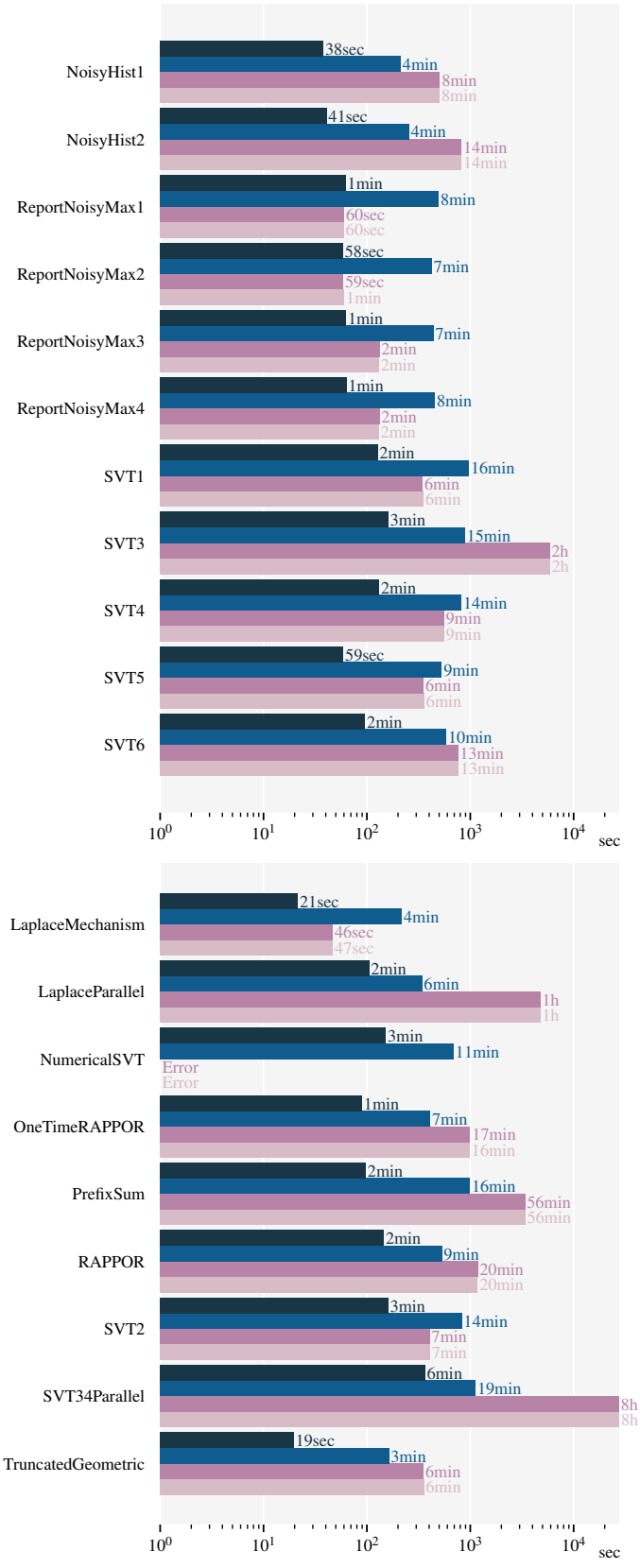

Fig. 11. Runtime of DD-Search (this work) and StatDP-Fixed. Note the logarithmic scale.

TABLE IV
Comparison of tools that demonstrate differential distinguishability.

Tool	Category	Input search	Attack search	Assumptions	Code
$\mathcal{T}_{\text{priv}}$ [11]	testing	distribution	exhaustive	oracle for $\Pr[M(a)=b]$	-
StatDP [12]	$\text{testing}_{\wp}$	patterns	patterns	semi-black-box access to M	[30]
DP-Finder [13]	$\text{maximizing}_{\wp}$	gradient descent	patterns	limited class of programs	[35]
DiPC [14]	testing	exhaustive	exhaustive	limited class of programs	[17]
DP-Stochastic-Tester [15]	$\text{maximizing}_{\wp}$	Halton sequence	small intervals	$\mathbb{B} \subseteq \mathbb{R}$	[36]
CheckDP [37]	testing	symbolic solver	symbolic solver	limited class of programs	[38]
DD-Search [this work]	$\text{maximizing}_{\wp}$	patterns	classifier	black-box access to M	Footnote 4

Our results indicate that this implementation is 0.25-DD and hence *clearly violates its theoretical privacy level of 0.1-DP*. Running this experiment took 36 min. Note that finding this vulnerability is completely automatic.

We stress that existing tools cannot detect floating-point vulnerabilities, either because their search space is not expressive enough (e.g., StatDP-Fixed), or because they assume idealized computation on real numbers (e.g., DiPC [14]). For the same reason, DP verifiers also miss such vulnerabilities.

Evaluating a Fixed Version. The *snapping mechanism* [34, §5.2] has been proposed as a defense against floating-point vulnerabilities in the Laplace mechanism. We implemented this defense to obtain a fixed version of the mechanism with target $\epsilon = 0.1$ and ran the above experiments again. Indeed, DD-Search is only able to demonstrate 0.098-DD in this case, indicating that the snapping mechanism defeats our attack.

VII. Related Work

Next, we discuss existing approaches detecting differential distinguishability (§VII-A) and inference attacks determining the input of an algorithm from its output (§VII-B).

A. Goal: Detecting Differential Distinguishability

Tab. IV compares different tools detecting differential distinguishability. It does not discuss verification of differential privacy [2]–[10], as this is not the focus of our work. Further, it omits ADP-Estimator [16], which only targets (ϵ, δ)-DP. Finally, we note that DPCheck [39] is not a testing approach in the sense of Fig. 9 (left), as it may incorrectly report DD for algorithms not meeting specific technical conditions.

Tools. $\mathcal{T}_{\text{priv}}$ [11] assumes oracle access to $\Pr[M(a)=b]$, which is typically not available. Specifically, it exhaustively iterates over all outputs $b \in \mathbb{B}$, samples inputs $a \in \mathbb{A}$, and evaluates $\Pr[M(a)=b]$ for every choice of a and b. It then applies a Lipschitz test to determine if varying a can lead to substantial changes in $\Pr[M(a)=b]$.

We have already discussed StatDP [12] in detail in §V. We note that StatDP requires semi-black-box access to M, as one of its post-processings requires running M without any noise.

DP-Finder samples sets $\mathcal{S}$, and then maximizes $\mathcal{E}(a, a', \mathcal{S})$ using gradient descent on (a, a'). This requires making $\mathcal{E}(\cdot, \cdot, \mathcal{S})$ differentiable, which is only possible for a limited class of programs M. For example, DP-Finder does not support arbitrary loops or hash functions (used in RAPPOR).

DiPC [14] exhaustively iterates over inputs $a \in \mathbb{A}$ and outputs $b \in \mathbb{B}$, and uses a symbolic solver to decide if $\Pr[M(a)=b] > e^{\epsilon} \Pr[M(a')=b]$. Due to the restrictions of solvers, DiPC cannot handle arbitrary loops or hash functions.

DP-Stochastic-Tester [15] samples inputs (a, a') and uses confidence intervals to determine if for specific intervals $[l, u] \subseteq \mathbb{R}$, $\Pr[M(a) \in [l,u]] > e^{\epsilon} \Pr[M(a') \in [l,u]]$, restricting its applicability to $\mathbb{B} \subseteq \mathbb{R}$.

CheckDP [37] synthesizes *alignment proofs* parameterized by θ, which, when valid, demonstrate ϵ-DP. To this end, it uses a symbolic solver to find a candidate parameter θ valid for a set of (differently synthesized) inputs $I_1, \ldots, I_n$, and then tries to confirm that θ is valid for *all* inputs. If the solver cannot find any valid θ for a given I_n, it transforms I_n to a candidate witness and uses a probabilistic solver to confirm that this witness indeed demonstrates ϵ-DD. As alignment proofs are incomplete, this confirmation may fail in principle. Like DiPC, CheckDP is subject to the limitations of (symbolic) solvers. For instance, it cannot handle hash functions or square roots.

In contrast to these tools, DD-Search (based on DP-Sniper) only requires black-box access to M.

Categories. The second column of Tab. IV categorizes the tools into approaches for testing and maximizing power (cp. Fig. 9). As explained in §V, there exists a conversion between these two types. Tab. IV uses subscript $\wp$ to indicate that the guarantees from Fig. 9 hold with high probability.

Searching Attacks. As discussed above, some existing works exhaustively check possible mechanism outputs [11, 14]. This is problematic, as it (i) precludes continuous output spaces (unless the output is projected to a discrete space, which may lose information), (ii) does not scale to large output spaces, and (iii) requires estimating low probabilities (because the probability of obtaining a specific output is small). We note that estimating low probabilities is not required in DiPC [14], as it computes probabilities symbolically.

Other works test for simple, hand-crafted output patterns [12, 13]. Unfortunately, such patterns do not generalize well to previously unseen algorithms (see §VI).

In contrast to these tools, DD-Search (based on DP-Sniper) introduces a general method to leverage machine learning classifiers for detecting differential distinguishability.

B. Technique: (Membership) Inference Attacks

In the following, we discuss attacks that infer if a given output $b = M(A)$ was generated from $A = a$ or $A = a'$.

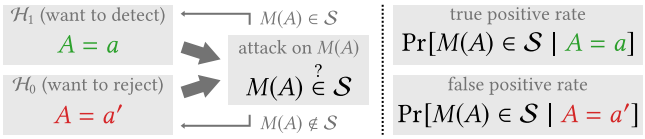

Fig. 12. Statistical hypothesis testing to determine input.

DP-Sniper relies on such attacks for the specific purpose of demonstrating differential distinguishability. An important special case of such attacks are membership inference attacks [27, 40]–[44], also known as tracing attacks, where $a = \mathcal{D} \cup \{d\}$ is a database including the information d of an individual and $a' = \mathcal{D}$ is the same database but excluding d.

Statistical Hypothesis Testing. Fig. 12 (left) phrases the goal of inference attacks as a statistical hypothesis test detecting $A = a$. For consistency with the notation of differential privacy, we describe the attacker's decision rule as a set $\mathcal{S}$, concluding $A = a$ if $M(A) \in \mathcal{S}$. Then, the goal of inference attacks is to simultaneously maximize their true positive rate and minimize their false positive rate, see Fig. 12 (right).

Differential Privacy Prevents Inference Attacks. Differential privacy limits the true positive rate of an inference attack in terms of its false positive rate—an immediate consequence of the definition of differential privacy (cp. Eq. (1)):

$$\Pr[M(a) \in \mathcal{S}] \leq \exp(\epsilon) \cdot \Pr[M(a') \in \mathcal{S}].$$

Based on this fundamental insight, various works point out that differential privacy thwarts inference attacks [43, Thm. 1], [27, §IV-A], [28, Thm. 2.4], [45, Prop. 4], [46, Thm. 2.1]. An immediate but rarely discussed consequence is that conversely, any successful inference attack disproves differential privacy [47, Lem. 2.5].

Complementary to these works, DP-Sniper turns any inference attack into a demonstration of differential distinguishability. To this end, it solves the key challenge of calibrating the true positive rate and false positive rate of this attack to obtain maximal differential distinguishability, while avoiding estimating small probabilities (recall §II-A).

Metrics for Inference Attacks. Existing inference attacks measure their success in terms of various metrics, including true positive rate (same as power or precision) [27, 41, 44], recall [41], false positive rate (same as error) [27, 44], accuracy [41, 42, 44], area under curve [40], or custom notions such as privacy loss [40], attacker gain [42], or indifferentiability [48]. Unfortunately, these metrics are typically hard to interpret in terms of privacy impact and often cannot quantify an attack as a single number. For example, achieving high true positive rate is uninformative, unless the false positive rate is small.

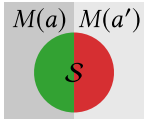

Fig. 13. Reference for different metrics.

In contrast, our work can be viewed as quantifying existing attacks in terms of their impact on differential privacy—a natural and simple metric for attacks.

VIII. Conclusion

We presented DP-Sniper, a practical approach which leverages classifiers to demonstrate that a given algorithm is differentially distinguishable. DP-Sniper finds approximately optimal attacks in terms of c-power, and takes into account that we cannot accurately estimate small probabilities.

Compared to a strengthened baseline, we can prove substantially stronger differential distinguishability.

We expect that future work can generalize our approach to (ϵ, δ)-DP by generalizing the notion of c-power.

Acknowledgment

We thank our shepherd and the anonymous reviewers for the constructive feedback. The research leading to these results was partially supported by an ERC Starting Grant 680358. Ilija Bogunovic is supported by the ETH Zürich Postdoctoral Fellowship 19-2 FEL-47.

References

[1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Theory of Cryptography*, 2006, vol. 3876. [Online]. Available: http://link.springer.com/10.1007/11681878_14

[2] J. Reed and B. C. Pierce, "Distance Makes the Types Grow Stronger: A Calculus for Differential Privacy," in *ICFP'10*, 2010. [Online]. Available: https://doi.org/10.1145/1932681.1863568

[3] G. Barthe, B. Köpf, F. Olmedo, and S. Zanella-Béguelin, "Probabilistic Relational Reasoning for Differential Privacy," in *TOPLAS'13*, 2013. [Online]. Available: https://doi.org/10.1145/2492061

[4] G. Barthe, B. Danezis, George Grégoire, C. Kunz, and S. Zanella-Béguelin, "Verified Computational Differential Privacy with Applications to Smart Metering," in *CSF'13*, 2013.

[5] G. Barthe, M. Gaboardi, E. J. Gallego Arias, J. Hsu, A. Roth, and P.-Y. Strub, "Higher-Order Approximate Relational Refinement Types for Mechanism Design and Differential Privacy," 2015. [Online]. Available: https://doi.org/10.1145/2775051.2677000

[6] G. Barthe, M. Gaboardi, B. Grégoire, J. Hsu, and P.-Y. Strub, "Proving Differential Privacy via Probabilistic Couplings," in *LICS'16*, 2016. [Online]. Available: https://doi.org/10.1145/2933575.2934554

[7] G. Barthe, N. Fong, M. Gaboardi, B. Grégoire, J. Hsu, and P.-Y. Strub, "Advanced Probabilistic Couplings for Differential Privacy," in *CCS'16*, 2016. [Online]. Available: https://doi.org/10.1145/2976749.2978391

[8] A. Albarghouthi and J. Hsu, "Synthesizing Coupling Proofs of Differential Privacy," *POPL'17*, vol. 2, no. POPL, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3177123.3158146

[9] D. Zhang and D. Kifer, "LightDP: Towards Automating Differential Privacy Proofs," in *POPL'17*, 2017. [Online]. Available: https://doi.org/10.1145/3009837.3009884

[10] Y. Wang, Z. Ding, G. Wang, D. Kifer, and D. Zhang, "Proving Differential Privacy with Shadow Execution," in *PLDI'19*, 2019. [Online]. Available: https://doi.org/10.1145/3314221.3314619

[11] K. Dixit, M. Jha, S. Raskhodnikova, and A. Thakurta, "Testing the Lipschitz Property over Product Distributions with Applications to Data Privacy," in *Theory of Cryptography*, 2013.

[12] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer, "Detecting Violations of Differential Privacy," in *CCS'18*, 2018. [Online]. Available: http://dl.acm.org/doi/10.1145/3243734.3243818

[13] B. Bichsel, T. Gehr, D. Drachsler-Cohen, P. Tsankov, and M. Vechev, "DP-Finder: Finding Differential Privacy Violations by Sampling and Optimization," in *CCS'18*, 2018. [Online]. Available: http://dl.acm.org/doi/10.1145/3243734.3243863

[14] G. Barthe, R. Chadha, V. Jagannath, A. P. Sistla, and M. Viswanathan, "Automated Methods for Checking Differential Privacy," 2019. [Online]. Available: http://arxiv.org/abs/1910.04137

[15] R. J. Wilson, C. Y. Zhang, W. Lam, D. Desfontaines, D. Simmons-Marengo, and B. Gipson, "Differentially Private SQL with Bounded User Contribution," 2019. [Online]. Available: http://arxiv.org/abs/1909.01917

[16] X. Liu and S. Oh, "Minimax Rates of Estimating Approximate Differential Privacy," 2019. [Online]. Available: http://arxiv.org/abs/1905.10335

[17] G. Barthe, R. Chadha, V. Jagannath, A. P. Sistla, and M. Viswanathan, "DiPC," accessed: 2020-04-08. [Online]. Available: https://anonymous.4open.science/r/febcbe47-1c53-41db-be91-ea98b4cf18c1/

[18] U. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response," in *CCS'14*, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2660267.2660348

[19] A. Gilbert and A. McMillan, "Property Testing for Differential Privacy," 2019. [Online]. Available: http://arxiv.org/abs/1806.06427

[20] E. L. Lehmann and J. P. Romano, *Testing statistical hypotheses*, 3rd ed., 2010, oCLC: 837651836.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*, 2006.

[22] C. R. Shalizi, *Advanced Data Analysis from an Elementary Point of View*, 2019. [Online]. Available: https://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/ADAfaEPoV.pdf

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[24] C. J. Clopper and E. S. Pearson, "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial," *Biometrika*, vol. 26, no. 4, 1934. [Online]. Available: https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/26.4.404

[25] P. Massart, "The Tight Constant in the Dvoretzky-Kiefer-Wolfowitz Inequality," *The Annals of Probability*, vol. 18, no. 3, 1990. [Online]. Available: http://projecteuclid.org/euclid.aop/1176990746

[26] A. W. van der Vaart, *Asymptotic Statistics*, 1998. [Online]. Available: http://ebooks.cambridge.org/ref/id/CBO9780511802256

[27] S. K. Murakonda, R. Shokri, and G. Theodorakopoulos, "Ultimate Power of Inference Attacks: Privacy Risks of Learning High-Dimensional Graphical Models," 2019. [Online]. Available: http://arxiv.org/abs/1905.12774

[28] L. Wasserman and S. Zhou, "A Statistical Framework for Differential Privacy," 2009. [Online]. Available: http://arxiv.org/abs/0811.2501

[29] C. Liu, X. He, T. Chanyaswad, S. Wang, and P. Mittal, "Investigating Statistical Privacy Frameworks from the Perspective of Hypothesis Testing," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, 2019. [Online]. Available: https://content.sciendo.com/view/journals/popets/2019/3/article-p233.xml

[30] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer, "StatDP," accessed: 2020-05-14. [Online]. Available: https://github.com/cmla-psu/statdp

[31] M. Lyu, D. Su, and N. Li, "Understanding the sparse vector technique for differential privacy," *Proceedings of the VLDB Endowment*, vol. 10, no. 6, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3055330.3055331

[32] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally Utility-Maximizing Privacy Mechanisms," in *STOC'09*, 2009. [Online]. Available: https://doi.org/10.1145/1536414.1536464

[33] V. Balcer and S. Vadhan, "Differential Privacy on Finite Computers," 2019. [Online]. Available: http://arxiv.org/abs/1709.05396

[34] I. Mironov, "On Significance of the Least Significant Bits for Differential Privacy," in *CCS'12*, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2382196.2382264

[35] B. Bichsel, T. Gehr, D. Drachsler-Cohen, P. Tsankov, and M. Vechev, "DP-Finder," accessed: 2020-04-08. [Online]. Available: https://github.com/eth-sri/dp-finder

[36] R. J. Wilson, C. Y. Zhang, W. Lam, D. Desfontaines, D. Simmons-Marengo, and B. Gipson, "DP-Stochastic-Tester," accessed: 2020-04-08. [Online]. Available: https://github.com/google/differential-privacy

[37] Y. Wang, Z. Ding, D. Kifer, and D. Zhang, "CheckDP: An Automated and Integrated Approach for Proving Differential Privacy or Finding Precise Counterexamples," in *CCS'20*, 2020. [Online]. Available: https://doi.org/10.1145/3372297.3417282

[38] Wang, Yuxin and Ding, Zeyu and Kifer, Daniel and Zhang, Danfeng, "CheckDP," accessed: 2020-11-12. [Online]. Available: https://github.com/cmla-psu/checkdp

[39] H. Zhang, E. Roth, A. Haeberlen, B. C. Pierce, and A. Roth, "Testing Differential Privacy with Dual Interpreters," *OOPSLA'20*, 2020. [Online]. Available: https://doi.org/10.1145/3428233

[40] A. Pyrgelis, C. Troncoso, and E. De Cristofaro, "Knock Knock, Who's There? Membership Inference on Aggregate Location Data," 2017. [Online]. Available: http://arxiv.org/abs/1708.06145

[41] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks against Machine Learning Models," 2017. [Online]. Available: http://arxiv.org/abs/1610.05820

[42] M. Nasr, R. Shokri, and A. Houmansadr, "Machine Learning with Membership Privacy using Adversarial Regularization," 2018. [Online]. Available: http://arxiv.org/abs/1807.05852

[43] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting," 2018. [Online]. Available: http://arxiv.org/abs/1709.01604

[44] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8835245/

[45] R. Hall, A. Rinaldo, and L. Wasserman, "Differential Privacy for Functions and Functional Data," *The Journal of Machine Learning Research*, vol. 14, no. 1, 2013.

[46] P. Kairouz, S. Oh, and P. Viswanath, "The Composition Theorem for Differential Privacy," *arXiv:1311.0776 [cs, math]*, 2015, arXiv: 1311.0776. [Online]. Available: http://arxiv.org/abs/1311.0776

[47] M. Bafna and J. Ullman, "The Price of Selection in Differential Privacy," *arXiv:1702.02970 [cs]*, 2017, arXiv: 1702.02970. [Online]. Available: http://arxiv.org/abs/1702.02970

[48] A. Kassem, G. Ács, C. Castelluccia, and C. Palamidessi, "Differential Inference Testing: A Practical Approach to Evaluate Sanitizations of Datasets," in *2019 IEEE Security and Privacy Workshops (SPW)*, 2019.

[49] J. Neyman and E. S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, 1933. [Online]. Available: https://www.jstor.org/stable/91247

[50] G. R. Shorack and J. A. Wellner, *Empirical Processes with Applications to Statistics*, 2009, no. 59, oCLC: ocn421147764.

TABLE V
Notational conventions used in this work.

Object	Notation
Differential Privacy	
Randomized algorithm	$M\colon \mathbb{A} \to \mathbb{B}$
Neighborhood	$\mathcal{N} \subseteq \mathbb{A} \times \mathbb{A}$
Input value	$a, a' \in \mathbb{A}$
Output value	$b \in \mathbb{B}$
Attack	$\mathcal{S} \in \mathcal{P}(\mathbb{B})$
Level of differential privacy	$\epsilon \in \mathbb{R} \cup \{\infty\}$
Level of differential distinguishability	$\xi \in \mathbb{R} \cup \{\infty\}$
Power of a witness $(a, a', \mathcal{S})$	$\mathcal{E}(a, a', \mathcal{S})$
Random variables	(capital latin letters)
Random input	$A \in \Delta(\mathbb{A})$
Random output	$B \in \Delta(\mathbb{B})$
Randomized attack	$\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]$
Generic random variable	X, X_i
Probability	
Set of distributions over $\mathbb{B}$	$\Delta(\mathbb{B})$
Probability that ϕ holds for randomly chosen $X_1, \ldots, X_n$	$\Pr[\phi(X_1, \ldots, X_n)]$
Standard deviation of random variable X	$\sigma(X)$
Confidence	$1 - \alpha$
Probability distributions	
Laplace distribution (location μ, scale b)	$\mathrm{lap}(\mu, b)$
Estimates	
Number of samples	N
Upper bound on error of estimate	ρ
Machine Learning	
Family of classifiers	$\{p_\theta(a\|b) \mid \theta \in \Theta\}$
Classifier	$p_\theta(a\|b)$ or simply θ

APPENDIX

A. Notational Conventions

Tab. V shows notational conventions used in this work.

B. Avoiding Small Probabilities

Lemma 1 (Guarantees for c-power)**.** For any algorithm $M\colon \mathbb{A} \to \mathbb{B}$, and any witness $(a, a', \mathcal{S})$ with $(a, a') \in \mathcal{N}$, M satisfies ξ-DD for $\xi = \mathcal{E}^{\geq c}(a, a', \mathcal{S})$.

Proof. If $\mathcal{E}^{\geq c}(a, a', \mathcal{S}) \leq 0$, Lem. 1 holds trivially. Otherwise, it holds because $\mathcal{E}^{\geq c}(a, a', \mathcal{S}) \leq \mathcal{E}(a, a', \mathcal{S})$, see Lem. 6. □

Lemma 6 (Lower Bound)**.** $\mathcal{E}^{\geq c}(a, a', \mathcal{S}) \leq \mathcal{E}(a, a', \mathcal{S})$ if $\mathcal{E}^{\geq c}(a, a', \mathcal{S}) > 0$.

Proof. We prove the following statement, which is equivalent to Lem. 1:

$$\frac{\max(c, \Pr[M(a) \in \mathcal{S}])}{\max(c, \Pr[M(a') \in \mathcal{S}])} \leq \frac{\Pr[M(a) \in \mathcal{S}]}{\Pr[M(a') \in \mathcal{S}]} \text{ if} \tag{16}$$

$$\max(c, \Pr[M(a) \in \mathcal{S}]) > \max(c, \Pr[M(a') \in \mathcal{S}]). \tag{17}$$

Case I. If $\Pr[M(a') \in \mathcal{S}] \geq c$, then $\Pr[M(a) \in \mathcal{S}] \geq c$ due to Eq. (17). Hence, Eq. (16) holds with equality.

Case II. If $\Pr[M(a') \in \mathcal{S}] \leq c$ and $\Pr[M(a) \in \mathcal{S}] \geq c$, Eq. (16) holds.

Case III. If $\Pr[M(a') \in \mathcal{S}] \leq c$ and $\Pr[M(a) \in \mathcal{S}] \leq c$, then we contradict $\mathcal{E}^{\geq c}(a, a', \mathcal{S}) > 0$. □

TABLE VI
Distributions used for proving Thm. 3 (minimum regret).

b	b_1	b_2	b_3	$\sum$
$p_1(b)$	$c + \rho$	c	$1 - 2c - \rho$	1
$p_2(b)$	c	$c + \rho$	$1 - 2c - \rho$	1
$p'(b)$	c	c	$1 - 2c$	1

C. Randomized Differential Privacy

Lem. 7 shows that randomized differential privacy is equivalent to the standard notion of differential privacy. As an immediate consequence, Lemmas 1 and 6 also hold for randomized attacks.

Lemma 7 (Randomized ϵ-DP)**.** For all $M\colon \mathbb{A} \to \mathbb{B}$, $\mathcal{N} \subseteq \mathbb{A} \times \mathbb{A}$, and $\epsilon \in \mathbb{R}$,

$$\forall (a, a') \in \mathcal{N}, \mathcal{S} \in \mathcal{P}(\mathbb{B}). \frac{\Pr[M(a) \in \mathcal{S}]}{\Pr[M(a') \in \mathcal{S}]} \leq \exp(\epsilon)$$

$$\iff$$

$$\forall (a, a') \in \mathcal{N}, \boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]. \frac{\Pr[M(a) \in \boldsymbol{\mathcal{S}}]}{\Pr[M(a') \in \boldsymbol{\mathcal{S}}]} \leq \exp(\epsilon)$$

Proof. " $\impliedby$ ": This direction is straight-forward, because $\Pr[M(a) \in \mathcal{S}] = \Pr[M(a) \in \boldsymbol{\mathcal{S}}]$ for a distribution $\boldsymbol{\mathcal{S}}$ with $\Pr[\boldsymbol{\mathcal{S}} = \mathcal{S}] = 1$, and likewise for a'.

" $\implies$ ": Let $(a, a') \in \mathcal{N}$ and $\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]$ be arbitrary. Then, we have:

$$\begin{aligned}\Pr[M(a) \in \boldsymbol{\mathcal{S}}] &= \sum_{\mathcal{S} \in \mathcal{P}(\mathbb{B})} \Pr[\boldsymbol{\mathcal{S}} = \mathcal{S}] \Pr[M(a) \in \mathcal{S}] \\ &\leq \sum_{\mathcal{S} \in \mathcal{P}(\mathbb{B})} \Pr[\boldsymbol{\mathcal{S}} = \mathcal{S}] \exp(\epsilon) \Pr[M(a') \in \mathcal{S}] \\ &= \exp(\epsilon) \Pr[M(a') \in \boldsymbol{\mathcal{S}}]\end{aligned}$$

□

D. Probability $q^\sharp$

Lemma 8 (Probability $q^\sharp$)**.** Line 7 in Alg. 1 yields $q^\sharp \in [0,1]$.

Proof. First, note that $q^\sharp$ is well-defined: Line 7 never divides by zero, as $t^\sharp = p'_i$ for some i.

Second, $q^\sharp \geq 0$ as $\sum_{i=1}^{N} \left[p'_i > t^\sharp\right] \leq cN$, because we index the sorted array p' at position $i \leq cN$.

Third, to show that $q^\sharp \leq 1$, it suffices to observe that $cN - \sum_{i=1}^{N} \left[p'_i > t^\sharp\right] \geq \sum_{i=1}^{N} \left[p'_i = t^\sharp\right]$ if and only if $cN \geq \sum_{i=1}^{N} \left[p'_i \geq t^\sharp\right]$. □

E. Minimum Regret

In this section, we prove Thm. 3:

Theorem 3 (Minimum Regret)**.** There exist inputs $(a, a') \in \mathcal{N}$ and algorithms $M_1, M_2\colon \mathbb{A} \to \mathbb{B}$ whose probability distributions differ by at most ρ, i.e., for all $x \in \mathbb{A}$ and $\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]$,

$$|\Pr[M_1(x) \in \boldsymbol{\mathcal{S}}] - \Pr[M_2(x) \in \boldsymbol{\mathcal{S}}]| \leq \rho,$$

such that for any attack $\boldsymbol{\mathcal{S}}^\sharp$, we have

$$\mathcal{E}^{\geq c}_{M_1}(a,a',\boldsymbol{\mathcal{S}}^\sharp) \leq \max_{\boldsymbol{\mathcal{S}}\in\mathbb{B}\to[0,1]} \mathcal{E}^{\geq c}_{M_1}(a,a',\boldsymbol{\mathcal{S}}) - \tfrac{1}{2}\tfrac{\rho}{c-\rho} \quad \text{or}$$
$$\mathcal{E}^{\geq c}_{M_2}(a,a',\boldsymbol{\mathcal{S}}^\sharp) \leq \max_{\boldsymbol{\mathcal{S}}\in\mathbb{B}\to[0,1]} \mathcal{E}^{\geq c}_{M_2}(a,a',\boldsymbol{\mathcal{S}}) - \tfrac{1}{2}\tfrac{\rho}{c-\rho},$$

where $\mathcal{E}^{\geq c}_{M_1}$ and $\mathcal{E}^{\geq c}_{M_2}$ denote the c-power of M_1 and M_2, respectively.

Proof. Our proof uses the probability distributions in Tab. VI as follows: $\Pr[M_1(a)=b]=p_1(b)$, $\Pr[M_2(a)=b]=p_2(b)$, and $\Pr[M_1(a')=b]=\Pr[M_2(a')=b]=p'(b)$.

For both M_1 and M_2, the optimal attack yields

$$\max_{\boldsymbol{\mathcal{S}}} \mathcal{E}^{\geq c}_{M_1}(a,a',\boldsymbol{\mathcal{S}}) = \max_{\boldsymbol{\mathcal{S}}} \mathcal{E}^{\geq c}_{M_2}(a,a',\boldsymbol{\mathcal{S}}) = \ln(c+\rho)-\ln(c).$$

Thus, it suffices to show that any given attack $\boldsymbol{\mathcal{S}}^\sharp$ cannot perform equally well on both M_1 and M_2. We can describe any such attack by $\Pr\left[b_i \in \boldsymbol{\mathcal{S}}^\sharp\right] = \lambda_i$ for $i \in \{1,2,3\}$. Without loss of generality, we can assume $\lambda_3 = 0$, as decreasing λ_3 increases both $\mathcal{E}^{\geq c}_{M_1}$ and $\mathcal{E}^{\geq c}_{M_2}$. With analogous reasoning, we can assume that $\lambda_1+\lambda_2 \geq 1$. Then, $\mathcal{E}^{\geq c}_{M_1}(a,a',\boldsymbol{\mathcal{S}}^\sharp)$ yields

$$\begin{aligned}
&\ln^{\geq c}(\lambda_1(c+\rho)+\lambda_2 c) - \ln^{\geq c}(\lambda_1 c+\lambda_2 c)\\
=&\ln(\lambda_1(c+\rho)+\lambda_2 c) \quad - \ln(\lambda_1 c+\lambda_2 c)\\
=&\ln\left(\tfrac{\lambda_1(c+\rho)+\lambda_2 c}{\lambda_1+\lambda_2}\right) \quad - \ln\left(\tfrac{\lambda_1 c+\lambda_2 c}{\lambda_1+\lambda_2}\right)\\
=&\ln\left(c+\tfrac{\lambda_1}{\lambda_1+\lambda_2}\rho\right) \quad - \ln(c)
\end{aligned}$$

Thus, we provide a lower bound for the regret as

$$\begin{aligned}
&\max_{\boldsymbol{\mathcal{S}}} \mathcal{E}^{\geq c}_{M_1}(a,a',\boldsymbol{\mathcal{S}}) - \mathcal{E}^{\geq c}_{M_1}(a,a',\boldsymbol{\mathcal{S}}^\sharp)\\
=&\ln(c+\rho) - \ln\left(c+\tfrac{\lambda_1}{\lambda_1+\lambda_2}\rho\right)\\
\geq&\frac{c+\rho-c-\frac{\lambda_1}{\lambda_1+\lambda_2}\rho}{c+\rho} = \tfrac{\lambda_2}{\lambda_1+\lambda_2}\frac{\rho}{c+\rho}
\end{aligned}$$

Using an analogous derivation for M_2, we conclude that

$$\max_{\boldsymbol{\mathcal{S}}} \mathcal{E}^{\geq c}_{M_2}(a,a',\boldsymbol{\mathcal{S}}) - \mathcal{E}^{\geq c}_{M_2}(a,a',\boldsymbol{\mathcal{S}}^\sharp) \geq \tfrac{\lambda_1}{\lambda_1+\lambda_2}\frac{\rho}{c+\rho}.$$

Thm. 3 follows because $\max\left(\tfrac{\lambda_1}{\lambda_1+\lambda_2}, \tfrac{\lambda_2}{\lambda_1+\lambda_2}\right) \leq \tfrac{1}{2}$. □

F. Neyman-Pearson Lemma

In the following, we show the Neyman-Pearson lemma [49] and its connection to threshold attacks. For a thorough introduction into the results of this section, we refer to [20, §3.2].

Definition 1 (Likelihood Ratio Test)**.** Let A be selected uniformly at random from $\{a,a'\}$, and $M\colon \mathbb{A} \to \mathbb{B}$ be a randomized algorithm. We define the likelihood ratio $\Lambda\colon \mathbb{B} \to \mathbb{R}\cup\{\infty\}$ as

$$\Lambda(b) := \frac{\Pr[M(a)=b]}{\Pr[M(a')=b]}.$$

Then, for $t \in \mathbb{R}\cup\{\infty\}$ and $q \in [0,1]$, the likelihood ratio test $\boldsymbol{\mathcal{T}}^{t,q} \in \mathbb{B} \to [0,1]$ is defined by

$$\Pr\left[b \in \boldsymbol{\mathcal{T}}^{t,q}\right] = \Pr[\Lambda(b) \succeq (t,q)].$$

In the domain of hypothesis testing, $A=a'$ is typically called the null hypothesis, and $A=a$ is called the alternative hypothesis. We note that we must assume $\infty > \infty$ to remain consistent with the conventions on handling 0 and ∞ in [20].

Lemma 9 (Neyman-Pearson lemma)**.** For any (randomized) test $\boldsymbol{\mathcal{T}} \in \mathbb{B} \to [0,1]$ and any $t \in \mathbb{R}\cup\{\infty\}, q \in [0,1]$ with

$$\Pr[M(a') \in \boldsymbol{\mathcal{T}}] = \Pr\left[M(a') \in \boldsymbol{\mathcal{T}}^{t,q}\right], \text{we have}$$
$$\Pr\left[M(a) \in \boldsymbol{\mathcal{T}}^{t,q}\right] \geq \Pr[M(a) \in \boldsymbol{\mathcal{T}}].$$

Proof. A proof is provided in [20, Thm. 3.2.1]. □

Lemma 10 (Neyman-Pearson existence)**.** For any $p' \in [0,1]$, there exist $t \in \mathbb{R}\cup\{\infty\}$ and $q \in [0,1]$ with

$$\Pr\left[M(a') \in \boldsymbol{\mathcal{T}}^{t,q}\right] = p'.$$

Proof. A proof is provided in [20, Thm. 3.2.1]. □

G. Key Properties of Threshold Attacks

Lemma 11 (Threshold Attacks are Likelihood Ratio Tests)**.** For any $t \in [0,1]$ and $q \in [0,1]$, we have $\boldsymbol{\mathcal{S}}^{t,q} = \boldsymbol{\mathcal{T}}^{\sigma(t),q}$, for $\sigma(p) = \frac{p}{1-p}$, if $t<1$ or $t=q=1$.

Likewise, for any $t \in \mathbb{R}\cup\{\infty\}$ and $q \in [0,1]$, we have $\boldsymbol{\mathcal{T}}^{t,q} = \boldsymbol{\mathcal{S}}^{\sigma^{-1}(t),q'}$, where

$$q' = \begin{cases} q & \text{if } t<\infty \\ 1 & \text{if } t=\infty \end{cases}.$$

Proof. Proof omitted due to space constraints. □

Lemma 12 (Threshold to Reach Probability)**.** For any probability $p' \in [0,1]$, there exist t,q with

$$\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\right] = p'.$$

Moreover, there exist such t,q which additionally satisfy $t<1$ or $t=q=1$.

Proof. Due to Lem. 10, there exists $\boldsymbol{\mathcal{T}}^{t,q}$ with the right property, which is equivalent to some $\boldsymbol{\mathcal{S}}^{t',q'}$ by Lem. 11. □

Lemma 13 (Highest True Positive Rate for Threshold Attacks)**.** Let $\boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]$ and $t,q \in [0,1]$ with $t<1$ or $t=q=1$. If

$$\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\right] = \Pr[M(a') \in \boldsymbol{\mathcal{S}}], \text{ then}$$
$$\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}\right] \geq \Pr[M(a) \in \boldsymbol{\mathcal{S}}].$$

We note that if t,q originate from Lem. 12, the condition $t<1$ or $t=q=1$ can always be satisfied.

Proof. Threshold attack $\boldsymbol{\mathcal{S}}^{t,q}$ is equivalent to a likelihood ratio test (Lem. 11). Thus, we can apply apply the Neyman-Pearson lemma Lem. 9. □

Definition 2 (τ)**.** Function $\tau\colon [0,1] \to [0,1]$ is defined by

$$\tau(\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\right]) = \Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}\right],$$

for $t<1$ or $t=q=1$.

Importantly, function τ is well-defined, as for every input $p' \in [0,1]$ to τ, there is exactly one output $\tau(p')$. First, there is at least one output value because there exist t, q with $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}] = p'$, due to Lem. 12. Second, there is at most one output for $\tau(p')$, because for two (t,q) and (t',q'), if $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}] = \Pr\Big[M(a') \in \boldsymbol{\mathcal{S}}^{t',q'}\Big] = p'$, then $\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}] = \Pr\Big[M(a) \in \boldsymbol{\mathcal{S}}^{t',q'}\Big]$, because both $\boldsymbol{\mathcal{S}}^{t,q}$ and $\boldsymbol{\mathcal{S}}^{t',q'}$ have highest true-positive rate according to Lem. 13.

Lemma 14 (τ concave)**.** Function τ is concave: For all $\lambda \in [0,1]$ and $p'_1, p'_2 \in [0,1]$, we have

$$\tau\Big((1-\lambda)p'_1 + \lambda p'_2\Big) \geq (1-\lambda)\tau(p'_1) + \lambda\tau(p'_2).$$

Further, $\tau(0) \geq 0$ and $\tau(1) = 1$.

Proof. This fact is a consequence from [20, §3.2]. It shows that the following set is convex:

$$\left\{\big(\Pr[M(a' \in \boldsymbol{\mathcal{S}})], \Pr[M(a) \in \boldsymbol{\mathcal{S}}]\big) \,\middle|\, \boldsymbol{\mathcal{S}} \in \mathbb{B} \to [0,1]\right\}.$$

Further, it shows that the upper boundary of this set corresponds to τ, which is sufficient to prove Lem. 14. □

Lemma 15 (τ monotone)**.** Function τ is monotone:

$$\Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\big] \leq \Pr\Big[M(a') \in \boldsymbol{\mathcal{S}}^{t',q'}\Big]$$
$$\Longrightarrow$$
$$\Pr\big[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}\big] \leq \Pr\Big[M(a) \in \boldsymbol{\mathcal{S}}^{t',q'}\Big]$$

Proof. This follows from the nature of $\boldsymbol{\mathcal{S}}^{t,q}$, which is essentially a threshold. □

Lemma 16 (True Positive Rate Larger)**.** For all $t, q \in [0,1]$,

$$\Pr\big[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}\big] \geq \Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\big]$$

Proof. If $t = 1$, then the lemma holds because

$$\begin{aligned}\Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\big] &= \Pr[p(a'|M(a)) \succeq (t,q)] \\ &= q \cdot \Pr[p(a|M(a')) = 1] \\ &= 0.\end{aligned}$$

Otherwise, this is a consequence from Lem. 14. □

Lemma 17 (Ratio Decreasing)**.** If

$$\Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\big] \leq \Pr\Big[M(a') \in \boldsymbol{\mathcal{S}}^{t',q'}\Big],$$

then

$$\frac{\Pr[M(a) \in \boldsymbol{\mathcal{S}}_{t,q}]}{\Pr[M(a') \in \boldsymbol{\mathcal{S}}_{t,q}]} \geq \frac{\Pr[M(a) \in \boldsymbol{\mathcal{S}}_{t',q'}]}{\Pr[M(a') \in \boldsymbol{\mathcal{S}}_{t',q'}]}.$$

Proof. If suffices to show that for $p'_1 \leq p'_2$,

$$\frac{\tau(p'_1)}{p'_1} = \frac{\tau\left(\frac{p'_1}{p'_2}p'_2\right)}{p'_1} \geq \frac{\frac{p'_1}{p'_2}\tau(p'_2)}{p'_1} = \frac{\tau(p'_2)}{p'_2},$$

which holds since τ is concave, for $\lambda = 0$. □

H. Optimal Threshold Attack

Lemma 4 (Optimal Attack)**.** For $c \in (0,1]$ and all neighboring inputs $(a, a') \in \mathcal{N}$, there exist $(t^\star, q^\star)$ that satisfy

$$\Pr\Big[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\Big] = c. \tag{13}$$

For any such $t^\star, q^\star$, we have

$$\max_{t\in[0,1], q\in[0,1]} \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t,q}) \leq \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t^\star,q^\star}). \tag{14}$$

Proof. Eq. (13) immediately follows from Lem. 12.
To show Eq. (14), we prove that for any $t \in [0,1], q \in [0,1]$,

$$\mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t,q}) \leq \mathcal{E}^{\geq c}(a, a', \boldsymbol{\mathcal{S}}^{t^\star,q^\star}).$$

Our proof proceeds by case distinction.

Case I. If $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}] < \Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\big]$, then $\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}] \leq \Pr\big[M(a) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\big]$ by Lem. 15.
Therefore, we have

$$\ln^{\geq c}\left(\Pr\big[M(a\,) \in \boldsymbol{\mathcal{S}}^{t,q}\big]\right) \leq \ln^{\geq c}\left(\Pr\Big[M(a\,) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\Big]\right)$$
and
$$\ln^{\geq c}\left(\Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\big]\right) = \underbrace{\ln^{\geq c}\left(\Pr\Big[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\Big]\right)}_{=c},$$

which implies Lem. 4.

Case II. If $\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}] > \Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\big]$, then

$$\begin{aligned}\Pr\big[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}\big] &\geq \Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\big] && \text{Lem. 16} \\ &\geq \Pr\Big[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\Big] && \text{Lem. 15} \\ &= c.\end{aligned}$$

Thus,

$$\begin{aligned}&\ln^{\geq c}\left(\Pr\big[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}\big]\right) \\ &\quad - \ln^{\geq c}\left(\Pr\big[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}\big]\right) \\ &= \ln\left(\frac{\Pr[M(a) \in \boldsymbol{\mathcal{S}}^{t,q}]}{\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t,q}]}\right) && \text{probs} \geq c \\ &\leq \ln\left(\frac{\Pr\big[M(a) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\big]}{\Pr[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}]}\right) && \text{Lem. 17}\end{aligned}$$

□

I. Dvoretzky-Kiefer-Wolfowitz Inequality

Theorem 4 (Dvoretzky-Kiefer-Wolfowitz inequality)**.** Let $\alpha > 0$, and $p_1, \ldots, p_N$ be independent samples from a random variable $P \in \Delta(\mathbb{R})$. Then, the event

$$\sup_{\substack{t\in\mathbb{R},\\ q\in[0,1]}} \left|\left(\frac{1}{N}\sum_{i=1}^{N} \Pr[p_i \succeq (t,q)]\right) - \Pr[P \succeq (t,q)]\right| \leq \rho$$

occurs with probability at least $1-\alpha$, for $\rho = \sqrt{\frac{\ln(2/\alpha)}{2n}}$.

Proof. See [25] for the original proof, and [50, Thm. 11.5] for a generalization to randomized thresholds. □

J. Approximate the Optimal Attack

Lemma 5 (Approximate Optimal Attack)**.** With probability at least $1-\alpha$,

$$\mathcal{E}^{\geq c}(a,a',\boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}) \geq \mathcal{E}^{\geq c}(a,a',\boldsymbol{\mathcal{S}}^{t^\star,q^\star}) - \tfrac{\rho}{c} - 2\left(\tfrac{\rho}{c}\right)^2,$$

for $\rho = \sqrt{\frac{\ln(2/\alpha)}{2n}}$ and assuming $\frac{\rho}{c} \leq \frac{1}{2}$.

Proof. Due to the Dvoretzky-Kiefer-Wolfowitz inequality (Thm. 4), we know that with probability $1-\alpha$, for all $t \in \mathbb{R}$ and $q \in [0,1]$,

$$\left|\left(\frac{1}{N}\sum_{i=1}^{N}\Pr[p_i' \succeq (t,q)]\right) - \Pr[P' \succeq (t,q)]\right| \leq \rho. \quad (18)$$

In this case, since $\frac{1}{N}\sum_{i=1}^{N}\Pr\left[p_i' \succeq (t^\sharp,q^\sharp)\right] = c$, we have $\left|\Pr\left[P' \succeq (t^\sharp,q^\sharp)\right] - c\right| \leq \rho$. We can thus conclude:

$$\left|\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right] - c\right| \leq \rho. \quad (19)$$

We now show that this is sufficient to establish Lem. 5.

Case I. If $\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right] \leq \Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]$, then

$$\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right] \leq \Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right] = c.$$

Therefore,

$$\ln^{\geq c}\left(\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right]\right) = \ln^{\geq c}\left(\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]\right).$$

Thus, it suffices to show that

$$\ln^{\geq c}\left(\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right]\right) \geq \ln^{\geq c}\left(\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]\right) - \frac{\rho}{c} - 2\left(\frac{\rho}{c}\right)^2$$

To this end, we derive

$$\begin{aligned}
&\ln^{\geq c}\left(\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right]\right) \\
&\geq \ln^{\geq c}\left(\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]\left(1-\frac{\rho}{c}\right)\right) && (\dagger) \\
&\geq \ln^{\geq c}\left(\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]\right) - \tfrac{\rho}{c} - 2\left(\tfrac{\rho}{c}\right)^2 && \text{Lem. 19}
\end{aligned}$$

For $(\dagger)$, note that

$$\begin{aligned}
&\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right] \geq \Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]\left(1-\tfrac{\rho}{c}\right) \\
&\qquad \iff \quad (t \neq 1 \text{ as } p' > 0) \\
&\tau\big(\underbrace{\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right]}_{:=p'}\big) \geq \tau\big(\underbrace{\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]}_{=c}\big)(1-\tfrac{\rho}{c}) \\
&\qquad \iff \\
&\qquad \tau(p') \geq \tau(c)(1-\tfrac{\rho}{c}) \\
&\qquad \impliedby \\
&\qquad \tau(p') \geq \tau(c)(1-\tfrac{\rho}{c}) + \underbrace{\tau(0)\tfrac{\rho}{c}}_{\geq 0} \\
&\qquad \impliedby \\
&\qquad \tau(p') \geq \tau\left(c\left(1-\tfrac{\rho}{c}\right)\right) \qquad \text{Lem. 14} \\
&\qquad \iff \\
&\qquad \tau(p') \geq \tau(c-\rho) \\
&\qquad \iff \\
&\qquad p' \geq c-\rho \qquad \text{Lem. 15}
\end{aligned}$$

which holds due to Eq. (19).

Case II. If $\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right] \geq \Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]$, then

$$\Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right] \geq \Pr\left[M(a) \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right] \quad \text{Lem. 15}$$

Thus, it suffices to show that

$$\ln^{\geq c}\left(\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right]\right) \leq \ln^{\geq c}\left(\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\star,q^\star}\right]\right) + \frac{c}{\rho} + \left(\frac{c}{\rho}\right)^2.$$

To this end, we derive

$$\begin{aligned}
\ln^{\geq c}\left(\Pr\left[M(a') \in \boldsymbol{\mathcal{S}}^{t^\sharp,q^\sharp}\right]\right) &\leq \ln^{\geq c}(c+\rho) && \text{Eq. (19)} \\
&= \ln(c+\rho) \\
&\leq \ln(c) + \frac{1}{c}\rho && \text{Lem. 18}
\end{aligned}$$

□

K. Helper Lemmas

In the following, we prove useful helper lemmas.

Lemma 18 (Lipschitz)**.** For all x, y, and c, we have

$$\left|\ln^{\geq c}(x) - \ln^{\geq c}(y)\right| \leq \frac{1}{c}\left|x-y\right|.$$

Proof. Proof omitted due to space constraints. □

Lemma 19. For all $p \in [0,1]$ and $R \in [0, \frac{1}{2}]$:

$$\ln^{\geq c}(p \cdot (1-R)) \geq \ln^{\geq c}(p) - R - 2R^2.$$

Proof. Proof omitted due to space constraints. □