DP-Opt: Identify High Differential Privacy Violation by Optimization

Ben Niu[1], Zejun Zhou[1,2], Yahong Chen[1,2], Jin Cao[3], and Fenghua Li[1,2(✉)]

[1] Institute of Information Engineering, CAS, Beijing, China
zhouzejun@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] School of Cyber Engineering, Xidian University, Xi'an, China

Abstract. Differential privacy has become a golden standard for designing privacy-preserving randomized algorithms. However, such algorithms are subtle to design, as many of them are found to have incorrect privacy claim. To help identify this problem, one approach is designing disprovers to search for counterexamples that demonstrate high violation of claimed privacy level. In this paper, we present DP-Opt(mizer), a disprover that tries to search for counterexamples whose lower bounds on differential privacy exceed the claimed level of privacy guaranteed by the algorithm. We leverage the insights of counterexample construction proposed by the latest work, meanwhile resolve their limitations. We transform the search task into an improved optimization objective which takes into account the empirical error, then solve it with various off-the-shelf optimizers. An evaluation on a variety of both correct and incorrect algorithms illustrates that DP-Opt almost always produces stronger guarantees than the latest work up to a factor of 9.42, with runtime reduced by an average of 19.2%.

Keywords: Differential privacy · Disprover · Lower bounds

1 Introduction

Differential Privacy (DP) [11] has become a golden standard that measures the level of privacy guaranteed by randomized mechanisms. DP protects individual's information because attackers cannot tell if an output was generated from database a_1, or its neighbor $A = a_2$ with that individual's record changed. However, designing such differentially private mechanisms can be error-prone, as existing papers have already identified incorrect privacy claims of published mechanisms [8,15]. Therefore, an important area of research is to verify the privacy level of a differentially private mechanism.

Generally, related works are divided into three types: formal verification, disprover, and the combination of both. Formal verification methods develop a proof system and use it to *prove* that mechanisms satisfy differential privacy [1,4,5,18,20]. However, these techniques are not able to *disprove* an incorrect privacy

L. Wang et al. (Eds.): WASA 2022, LNCS 13472, pp. 406–416, 2022.
https://doi.org/10.1007/978-3-031-19214-2_34

claim. On the contrary, disprovers try to search for counterexamples of a mechanism that violate the claimed differential privacy level [2,6,7,10,19]. Typically, two approaches are taken. On one hand, StatDP [10] constructs and tests a statistical hypothesis. Given a preconceived privacy parameter $\epsilon_0 > 0$, it tries to find a counterexample that violates the privacy condition, therefore rejects incorrect mechanisms. On the other hand, works like DP-Finder [6] and DP-Sniper [7] search for the lower bound on differential privacy. Such lower bound is found by maximizing the privacy loss function derived from DP definition. Inputs to this function is considered a counterexample if the corresponding lower bound exceeds the claimed privacy level. Both approaches try to identify counterexamples so as to demonstrate that the privacy claim is incorrect, and further provide insights for developers to fix the bugged mechanism. As opposed to formal verification methods, a disprover cannot prove that a mechanism satisfies the claimed privacy if it fails to generate any counterexample. Another type of methods [3,14,17] combines the previous two methods, and either synthesizes proofs for correct mechanisms or generates counterexamples for incorrect mechanisms.

This Work. We present an enhanced disprover DP-Opt, which aims to resolve the limitations in counterexample construction of the latest work DP-Sniper [7] and produce higher privacy violations. Specifically, our contributions are:

- DP-Opt, an algorithm that leverages the idea of optimization to resolve the limitations of DP-Sniper by transforming the search task into an improved optimization objective to be solved with off-the-shelf numerical optimizers.
- An implementation[1] and evaluation of DP-Opt on a wide variety of randomized algorithms demonstrating significantly higher guarantees on privacy by a factor up to 9.42 with an average reduced runtime of 19.2%, compared with the latest work.

2 DP Disprover Background

2.1 Differntial Privacy

Formally, given a mechanism $M : \mathbb{A} \to \mathbb{B}$ that inputs database $a \in \mathbb{A}$ and outputs $b \in \mathbb{B}$, M is ϵ-differentially private (ϵ-DP) if for every pair of neighboring inputs $(a_1, a_2) \in \mathcal{N}$ and for every attack $\mathcal{S} \subseteq \mathbb{B}$,

$$\ln\left(\Pr[M(a_1) \in \mathcal{S}]\right) - \ln\left(\Pr[M(a_2) \in \mathcal{S}]\right) \leq \epsilon, \tag{1}$$

where the neighborhood $\mathcal{N} \subseteq \mathbb{A} \times \mathbb{A}$ consists of neighboring database pairs that differ in only one record. The privacy parameter $\epsilon \in [0, \infty)$ quantifies the privacy level guaranteed by M, where smaller ϵ corresponds to higher privacy guarantees, and contrarily, $\epsilon = \infty$ means no privacy at all.

2.2 Prior Knowledge of DP-Sniper

Power. Derived from Eq. 1, *power* [7] of a witness $(a_1, a_2, \mathcal{S})$ is defined as

$$\mathcal{E}(a_1, a_2, \mathcal{S}) := \ln\left(\Pr[M(a_1) \in \mathcal{S}]\right) - \ln\left(\Pr[M(a_2) \in \mathcal{S}]\right).$$

[1] Available at https://github.com/barryZZJ/dp-opt.

The highest power found by disprover is regarded as the *lower bound* on the privacy level of M. Therefore, we aim to find the maximum power so as to measure the level of violation against the claimed privacy of M.

Estimation. With samples $b^{(0)}, \ldots, b^{(N-1)} \sim M(a)$, we can estimate the probability $\Pr[M(a) \in \mathcal{S}]$ as $\hat{P}^N_{M(a)\in\mathcal{S}} = \frac{1}{N}\sum_{i=0}^{N-1} \Pr[b^{(i)} \in \mathcal{S}]$. Therefore, estimation of power $\hat{\mathcal{E}}(a_1, a_2, \mathcal{S})$ is computed by replacing the probability terms with their estimations.

Threshold Attack. Threshold attack [7] is a type of randomized attack that selects b probabilistically according to the membership function $\mathcal{S}^{t,q} : \mathbb{B} \to [0, 1]$. Specifically, it utilizes the novel idea of posterior probability $p(a_1|b)$ that defines the probability that an output b originates from $M(a_1)$, as opposed to $M(a_2)$. A threshold attack incorporates the output whose posterior probability is above some threshold t, in order to produce high power. Additionally, an output is only included with probability q if its posterior probability is equal to t. This limits the size of the threshold attack and ensures continuousness of power. Formally, the membership function of threshold attack $\mathcal{S}^{t,q}(b)$ is defined as

$$\Pr[b \in \mathcal{S}^{t,q}] = [p(a_1|b) > t] + q \cdot [p(a_1|b) = t], \tag{2}$$

where the Iverson bracket $[\phi]$ outputs 1 if ϕ is true, and 0 otherwise. Moreover, estimation of $\Pr[M(a) \in \mathcal{S}^{t,q}]$ is computed as

$$\hat{P}^N_{M(a)\in\mathcal{S}^{t,q}} = \frac{1}{N}\sum_{i=0}^{N-1}[p(a_1|b^{(i)}) > t] + \frac{1}{N} \cdot q \sum_{i=0}^{N-1}[p(a_1|b^{(i)}) = t]. \tag{3}$$

Parameter c. According to [7], the deviation of $\hat{P}^N_{M(a)\in\mathcal{S}}$ increases rapidly when it becomes smaller, causing estimation on power unreliable. To avoid this issue, DP-Sniper discards small probabilities below some constant $c \in (0, 1]$. However, this predefined parameter makes a considerable impact on results, as illustrated in the next section.

3 Motivation and Ideas

3.1 Limitations of DP-Sniper

We now demonstrate the issue of predefining c with the following example.

Example 1. Consider the 0.5-DP Laplace mechanism $\mathcal{L}_{0.5}(a) = a + \text{lap}(0, 2)$, which adds Laplace noise with mean 0 and scale $1/0.5 = 2$ to its input $a \in \mathbb{R}$ [7, Ex. 1]. The top plot in Fig. 1 shows the cumulative distribution function of $\mathcal{L}_{0.5}(0)$ and $\mathcal{L}_{0.5}(1)$ (blue and orange solid line respectively) by constructing the attack $\mathcal{S}^{t,q} = (-\infty, b)$, with c set to $c^* = 0.2$ (red dashed line). The bottom plot demonstrates the corresponding power by the brown solid line.

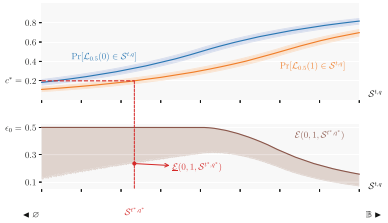
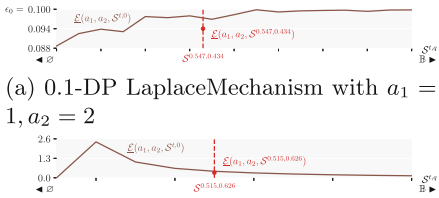


Fig. 1. Cumulative distribution function and power of $\mathcal{L}_{0.5}$, with confidence intervals indicated by the shaded area. (Color figure online)



(a) 0.1-DP LaplaceMechanism with $a_1 = 1, a_2 = 2$

(b) ∞-DP NoisyMax4 with $a_1 = [1, 1, 1, 1, 1], a_2 = [2, 2, 2, 2, 2]$

Fig. 2. Experiment values of $\underline{\mathcal{E}}(a_1, a_2, \mathcal{S}^{t,0})$.

In Example 1, for a pair of neighboring inputs $(0, 1) \in \mathcal{N}$, DP-Sniper constructs the threshold attack $\mathcal{S}^{t^*,q^*}$ by selecting t^*, q^* that satisfy $\Pr[\mathcal{L}_{0.5}(1) \in \mathcal{S}^{t^*,q^*}] = 0.2$. This automatically ensures $\Pr[\mathcal{L}_{0.5}(0) \in \mathcal{S}^{t^*,q^*}] \geq 0.2$ according to the properties of posterior probability. After the external algorithm DD-Search [7] generates different neighboring inputs, it invokes DP-Sniper and selects the best witness constructed, in this case $(0, 1, \mathcal{S}^{t^*,q^*})$. With this, it calculates the lower bound on power $\underline{\mathcal{E}} \approx 0.2$ (discussed in Sect. 3.2), as indicated by the red dot in Fig. 1. However, as the brown shaded area demonstrates, better lower bound on power can be achieved if c^* was initialized otherwise (in this case to around 0.3).

In fact, this problem occurs for almost all mechanisms, as confirmed by our experiments shown in Fig. 2. We enumerated attacks $\mathcal{S}^{t,0}$ of various t (with $q = 0$ for simplicity) and computed each $\underline{\mathcal{E}}$ for 0.1-DP LaplaceMechanism (Fig. 2a) and ∞-DP NoisyMax4 (Fig. 2b). The red dot in each plot is the final lower bound produced by DD-Search.

3.2 Ideas

Determine Optimization Objective. Inspired by Fig. 1, we decide to skip the procedure of determining c, and aims to find a threshold attack $\mathcal{S}^{t^\Delta,q^\Delta}$ that directly maximizes the lower bound on power $\underline{\mathcal{E}}$ for given $(a_1, a_2) \in \mathcal{N}$:

$$\mathcal{S}^{t^\Delta,q^\Delta} = \underset{t\in[0,1],\ q\in[0,1]}{\arg\max}\ \underline{\mathcal{E}}\left(a_1, a_2, \mathcal{S}^{t,q}\right).$$

We note that our work also discards small probabilities which induce high deviation, because the lower bound $\underline{\mathcal{E}}$ represents the worst-case scenario, finding the highest $\underline{\mathcal{E}}$ automatically leaves out imprecise probability estimations.

Now we describe the derivation of the optimization objective. Given mechanism M and neighboring inputs (a_1, a_2), for each output b, its posterior probability $p(a_1|b)$ is determined. Thus $\Pr[b \in \mathcal{S}^{t,q}]$ only varies by different combination of $t \in [0, 1]$ and $q \in [0, 1]$ (recall Eq. 2). Then, according to Eq. 3, $\hat{P}^N_{M(a)\in\mathcal{S}^{t,q}}$ also

only relies on t, q. Therefore, given neighboring inputs $(a_1, a_2) \in \mathcal{N}$ of mechanism M, lower bound on power $\underline{\mathcal{E}}(a_1, a_2, \mathcal{S}^{t,q})$ can be regarded as a function of $\mathcal{S}^{t,q}$ determined by t, q.

As a result, the aim of our work is to search for the best combination of variables t, q such that the corresponding threshold attack $\mathcal{S}^{t,q}$ produces the highest $\underline{\mathcal{E}}$. This is a maximization problem of a bivariate scalar function $\underline{\mathcal{E}}(t, q)$ under the constraint of $t \in [0, 1], q \in [0, 1]$:

$$t^{\Delta}, q^{\Delta} = \underset{t\in[0,1],\ q\in[0,1]}{\arg\max}\ \underline{\mathcal{E}}(t, q). \tag{4}$$

In addition, the impact of q can be ignored for some mechanisms if $p(a_1|b^{(i)}) = t$ in Eq. 3 rarely occurs. Hence, we transform Eq. 4 into a maximization problem of a univariate scalar function $\underline{\mathcal{E}}(t, 0)$ constrained by $t \in [0, 1]$:

$$t^{\Delta} = \underset{t\in[0,1]}{\arg\max}\ \underline{\mathcal{E}}(t, 0), \tag{5}$$

in expectation of better results in special cases.

Confidence Intervals of Power. We now discuss confidence intervals and derive bounds on power inspired by [6,7]. First, we apply the Clopper-Pearson confidence interval [9] on $\hat{P}^{N}_{M(a)\in\mathcal{S}}$ in order to derive the upper bound $\overline{P}^{N,\alpha/2}_{M(a)\in\mathcal{S}}$ and the lower bound $\underline{P}^{N,\alpha/2}_{M(a)\in\mathcal{S}}$, which both hold except with probability $\alpha/2$. In the top plot of Fig. 1, such bounds on probabilities are illustrated by the blue and orange shaded areas around respective solid lines. Then, we can use them to derive the bounds on power $\hat{\mathcal{E}}(a_1, a_2, \mathcal{S})$.

Theorem 1. *For neighboring inputs $(a_1, a_2) \in \mathcal{N}$, lower bound $\underline{\mathcal{E}}(a_1, a_2, \mathcal{S})$ and upper bound $\overline{\mathcal{E}}(a_1, a_2, \mathcal{S})$ on $\hat{\mathcal{E}}(a_1, a_2, \mathcal{S})$ both hold with probability $1 - \alpha$, where*

$$\underline{\mathcal{E}}\left(a_1, a_2, \mathcal{S}\right) = \ln\left(\underline{P}^{N,\alpha/2}_{M(a_1)\in\mathcal{S}}\right) - \ln\left(\overline{P}^{N,\alpha/2}_{M(a_2)\in\mathcal{S}}\right),$$
$$\overline{\mathcal{E}}\left(a_1, a_2, \mathcal{S}\right) = \ln\left(\overline{P}^{N,\alpha/2}_{M(a_1)\in\mathcal{S}}\right) - \ln\left(\underline{P}^{N,\alpha/2}_{M(a_2)\in\mathcal{S}}\right).$$

The lower bound on power is depicted by the brown shaded area below the brown solid line in Fig. 1. This bound holds even if probability estimations $\hat{p}$ are imprecise, because Clopper-Pearson interval is a type of *exact* interval [16] which has a coverage probability of *at least* $1 - \alpha$ for all values of $\hat{p}$. For this reason, we use $\underline{\mathcal{E}}(a_1, a_2, \mathcal{S})$ as both the optimization objective and final output, and furthermore conclude that the privacy level of M is at best $\underline{\mathcal{E}}$ with probability $1 - \alpha$ at least.

4 Our Disprover

In this section, we present the flow of our disprover. We first introduce DP-Opt that searches for optimal threshold attack by solving the optimization objective. Then we propose the external algorithm PowerSearcher that utilizes DP-Opt and produces the highest lower bound on power.

4.1 DP-Opt: Search for Optimal Threshold Attack

Given a neighboring input pair (a_1, a_2), DP-Opt searches for the optimal threshold attack with the following steps. First, a machine learning classifier $p_\theta(a_1|b)$ parametrized by θ is trained with N_{train} samples. It approximates the posterior probability $p(a_1|b)$ described in Sect. 2.2. We refer to [7] for more details as this is not focused in our work. Then, we exploit off-the-shelf numerical optimizers to solve the optimization objectives Eq. 4 and Eq. 5. Each optimizer tries to maximize $\underline{\mathcal{E}}(t, q)$ or $\underline{\mathcal{E}}(t, 0)$ with N_{check} samples. Among them, the maximum $\underline{\mathcal{E}}$ is selected, along with the inputs t^Δ, q^Δ. Finally, the optimal threshold attack $\mathcal{S}^{t^\Delta, q^\Delta}$ for the given input pair is constructed using parameters t^Δ, q^Δ, and returned by DP-Opt.

4.2 PowerSearcher: Search for High Privacy Violation

Guided by DD-Search [7], we discuss the details of PowerSearcher that leverages DP-Opt to find the highest $\underline{\mathcal{E}}$. First, different neighboring input pairs $(a_1^{(i)}, a_2^{(i)})$ are generated based on heuristic patterns [10]. For each input pair, a candidate witness is constructed by combining the input pair with corresponding optimal attack $\mathcal{S}^{(i)}$. Then, among all candidate witnesses, the optimal witness is selected according to its lower bound $\underline{\mathcal{E}}(a_1, a_2, \mathcal{S})$ computed with N_{check} samples. While most works compare the estimation on power, we compare the lower bound in order to avoid high deviation caused by small probability. In implementation, we reuse the maximum value found in step two to reduce computational cost. Finally, the lower bound on power of the optimal witness is computed again with fresh N_{final} samples and returned by PowerSearcher, along the witness. In this step, the sample size N_{final} is larger than N_{check} to produce a tighter bound.

5 Evaluation

5.1 Implementation

Inherited from [7], we implemented DP-Opt and PowerSearcher in Python based on the notion from Li et al. [13]. Since different classifiers have insignificant impact on performance [7], we only choose logistic regression classifier due to time limitation. Additionally, in attack searching, we applied binary search and reused the same sample on different optimizers. This substantially reduced runtime as computing and optimizing $\underline{\mathcal{E}}(t, q)$ need to repeatedly estimate probabilities and try various combinations of t, q.

Input Pattern Generation. We used the heuristic patterns proposed by Ding et al. [10] for input generation. For example, category *one above* corresponds to $(a_1, a_2) = ([1, 1, 1, 1, 1], [2, 1, 1, 1, 1])$.

Parameters. Following the guideline in [7], we used sample sizes $N_{\text{check}} = N_{\text{train}} = 10.7 \cdot 10^6$, and $N_{\text{final}} = 2 \cdot 10^8$, with $\alpha = 0.1$. The logistic regression model is trained using regularized stochastic gradient descent optimization and binary cross entropy loss, with epoch number 10, learning rate 0.3, momentum 0.3 and regularization weight 0.001.

Optimizers. Upon comparison, we selected several optimizers provided by SciPy in consideration of both performance and runtime cost. Their orders are as follows: Nelder-Mead(bi), Nelder-Mead(uni), COBYLA(bi), Differential Evolution(bi), Differential Evolution(uni), Powell(bi), COBYLA(uni), where *bi* and *uni* correspond to bivariate optimization objective $\underline{\mathcal{E}}(t, q)$ and univariate optimization objective $\underline{\mathcal{E}}(t)$ respectively. In implementation, we set initial guesses $t_0 = 0.5, q_0 = 0.5$, and kept the default values for the rest optional parameters.

5.2 Mechanisms Evaluated

We evaluated mechanisms listed in Table 1, including widely used mechanisms and their variations. They cover a variety of output types such as reals, integers and boolean values. The second column is their neighborhood definition, where $\|\cdot\|_p$ is the p-norm neighborhood $\mathcal{N} = \{(a_1, a_2) \mid \|a_1 - a_2\|_p \leq 1\}$.

Table 1. Evaluated mechanisms with their neighborhoods, expected DP and optimization objectives.

Mechanism	$\mathcal{N}$	ϵ	Objective
LaplaceMechanism [11]	$\|\cdot\|_1$	0.1	$\underline{\mathcal{E}}(t,q), \underline{\mathcal{E}}(t)$
NoisyHist1 [10, Alg. 9]	$\|\cdot\|_1$	0.1	$\underline{\mathcal{E}}(t,q), \underline{\mathcal{E}}(t)$
NoisyHist2 [10, Alg. 10]	$\|\cdot\|_1$	10	$\underline{\mathcal{E}}(t,q), \underline{\mathcal{E}}(t)$
NoisyMax1 [10, Alg. 5]	$\|\cdot\|_\infty$	0.1	$\underline{\mathcal{E}}(t,q)$
NoisyMax2 [10, Alg. 6]	$\|\cdot\|_\infty$	0.1	$\underline{\mathcal{E}}(t,q)$
NoisyMax3 [10, Alg. 7]	$\|\cdot\|_\infty$	∞	$\underline{\mathcal{E}}(t,q), \underline{\mathcal{E}}(t)$
NoisyMax4 [10, Alg. 8]	$\|\cdot\|_\infty$	∞	$\underline{\mathcal{E}}(t,q), \underline{\mathcal{E}}(t)$
SVT1 [15, Alg. 1]	$\|\cdot\|_\infty$	0.1	$\underline{\mathcal{E}}(t,q)$
SVT2 [15, Alg. 2]	$\|\cdot\|_\infty$	0.1	$\underline{\mathcal{E}}(t,q)$
SVT3 [15, Alg. 3]	$\|\cdot\|_\infty$	∞	$\underline{\mathcal{E}}(t,q)$
SVT4 [15, Alg. 4]	$\|\cdot\|_\infty$	0.175	$\underline{\mathcal{E}}(t,q)$
SVT5 [15, Alg. 5]	$\|\cdot\|_\infty$	∞	$\underline{\mathcal{E}}(t,q)$
SVT6 [15, Alg. 6]	$\|\cdot\|_\infty$	∞	$\underline{\mathcal{E}}(t,q)$
OneTimeRAPPOR [12, Steps 1–2]	$\|\cdot\|_1$	0.8	$\underline{\mathcal{E}}(t,q), \underline{\mathcal{E}}(t)$
RAPPOR [12, Steps 1–3]	$\|\cdot\|_1$	0.4	$\underline{\mathcal{E}}(t,q), \underline{\mathcal{E}}(t)$

Parameter Configuration. We set the parameters for each mechanism in accordance with DP-Sniper. Specifically, let ϵ_0 be the target DP guarantee,

- LaplaceMechanism uses $\epsilon_0 = 0.1$.
- NoisyHist1-2 and NoisyMax1-4 set $\epsilon_0 = 0.1$ with input length 5.
- SVT1-6 are instantiated by $\epsilon_0 = 0.1$ with input length 10 and additional parameters $c = 1, \Delta = 1, T = 1$ (except $T = 0.5$ for SVT1).
- OneTimeRAPPOR is initialized with parameters $k = 20, h = 4, f = 0.95$.
- RAPPOR is parametrized by $k = 20, h = 4, f = 0.75, q = 0.55$.

The corresponding expected privacy guarantees are listed in the third column of Table 1. For all mechanisms, we ran our disprover on each optimization objective (indicated by the last column) for seven times with suitable optimizers.

5.3 Results

Power. Figure 3 compares the average value of the final lower bound $\underline{\mathcal{E}}$ found between PowerSearcher and DD-Search. Results show that PowerSearcher is generally better with at least equal results in certain cases. Specifically, for most mechanisms with finite privacy target, PowerSearcher found tighter bounds, resolving the uncertainty of DD-Search's conclusion. For example, for NoisyHist1, DD-Search only narrows ϵ to $[0.098, 0.1]$ while PowerSearcher ensures it to be 0.1-DP. Especially, we manage to demonstrate NoisyHist2 to be 10-DP correctly in contrast to DD-Search only results in 4.605. For mechanisms known to be ∞-DP, PowerSearcher performs significantly better by a factor up to 9.42, except for NoisyMax3 which is 0.25-DP when input length is 5 (our configuration) [7, Sect. VI]. Unfortunately, for state-of-the-art mechanisms such as RAPPOR, PowerSearcher fails to derive better results. We attribute this to be the fundamental inability of threshold attacks.

Runtime. Figure 4 compares the runtime between PowerSearcher and DD-Search for each mechanism. We managed to reduce an average of 19.2% of runtime consumption, after exploiting the improvement methods mentioned in Sect. 4.2 and Sect. 5.1. We note that since our method is more flexible in choosing optimizers, trade-off between performance and runtime can be further made.

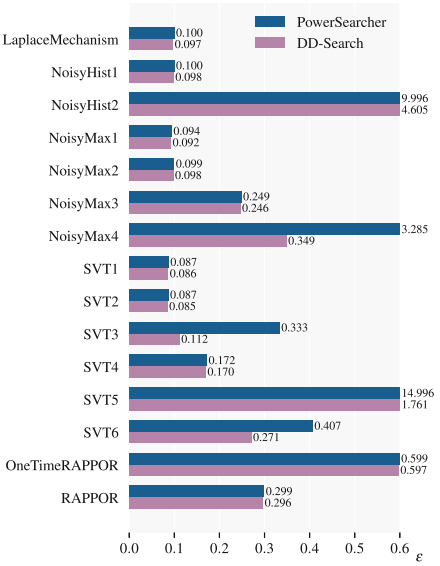


Fig. 3. Average $\underline{\mathcal{E}}$ found between PowerSearcher and DD-Search, where higher values are better.

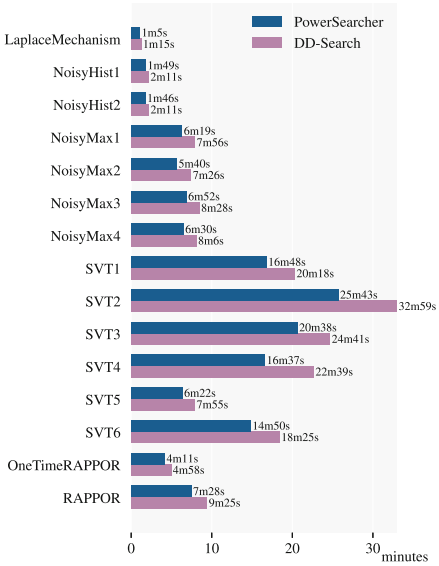


Fig. 4. Runtime of PowerSearcher and DD-Search.

6 Conclusion

We proposed DP-Opt, an improved disprover on the latest work by maximizing the lower bound on privacy level for a given mechanism. It exploits off-the-shelf optimizers to produce threshold attacks that yield optimal lower bound on power, and also avoids small probabilities that are difficult to estimate accurately. Results demonstrate significant improvement on privacy bounds compared with the latest baseline, with a fair amount of runtime saved. Future works are expected to employ an optimal optimizer that generalizes well on all optimization objectives to greatly reduce runtime while preserving better results.

Acknowledgements. This work is supported by the National Key R&D Program of China (2021YFB3100300), the National Natural Science Foundation of China (61872441, 61932015), and the Youth Innovation Promotion Association, Chinese Academy of Sciences (2019160, 2021154).

References

1. Albarghouthi, A., Hsu, J.: Synthesizing coupling proofs of differential privacy. Proc. ACM Program. Lang. **2**(POPL), 1–30 (2017)
2. Askin, Ö., Kutta, T., Dette, H.: Statistical quantification of differential privacy: a local approach. arXiv preprint arXiv:2108.09528 (2021)

3. Barthe, G., Chadha, R., Jagannath, V., Sistla, A.P., Viswanathan, M.: Deciding differential privacy for programs with finite inputs and outputs. In: Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 141–154 (2020)
4. Barthe, G., Gaboardi, M., Grégoire, B., Hsu, J., Strub, P.Y.: Proving differential privacy via probabilistic couplings. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 749–758 (2016)
5. Barthe, G., Köpf, B., Olmedo, F., Zanella Beguelin, S.: Probabilistic relational reasoning for differential privacy. In: Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 97–110 (2012)
6. Bichsel, B., Gehr, T., Drachsler-Cohen, D., Tsankov, P., Vechev, M.: DP-Finder: finding differential privacy violations by sampling and optimization. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 508–524 (2018)
7. Bichsel, B., Steffen, S., Bogunovic, I., Vechev, M.: DP-Sniper: black-box discovery of differential privacy violations using classifiers. In: 2021 IEEE Symposium on Security and Privacy (SP), pp. 391–409 (2021)
8. Chen, Y., Machanavajjhala, A.: On the privacy properties of variants on the sparse vector technique. arXiv preprint arXiv:1508.07306 (2015)
9. Clopper, C.J., Pearson, E.S.: The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika **26**(4), 404–413 (1934)
10. Ding, Z., Wang, Y., Wang, G., Zhang, D., Kifer, D.: Detecting violations of differential privacy. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 475–489 (2018)
11. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_14
12. Erlingsson, Ú., Pihur, V., Korolova, A.: RAPPOR: randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 1054–1067 (2014)
13. Li, F., Li, H., Niu, B., Chen, J.: Privacy computing: concept, computing framework, and future development trends. Engineering **5**(6), 1179–1192 (2019)
14. Liu, D., Wang, B.Y., Zhang, L.: Verifying pufferfish privacy in hidden Markov models. In: International Conference on Verification, Model Checking, and Abstract Interpretation, pp. 174–196 (2022)
15. Lyu, M., Su, D., Li, N.: Understanding the sparse vector technique for differential privacy. arXiv preprint arXiv:1603.01699 (2016)
16. Thulin, M.: The cost of using exact confidence intervals for a binomial proportion. Electron. J. Stat. **8**(1), 817–840 (2014)
17. Wang, Y., Ding, Z., Kifer, D., Zhang, D.: CheckDP: an automated and integrated approach for proving differential privacy or finding precise counterexamples. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 919–938 (2020)
18. Wang, Y., Ding, Z., Wang, G., Kifer, D., Zhang, D.: Proving differential privacy with shadow execution. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 655–669 (2019)

19. Wilson, R.J., Zhang, C.Y., Lam, W., Desfontaines, D., Simmons-Marengo, D., Gipson, B.: Differentially private SQL with bounded user contribution. arXiv preprint arXiv:1909.01917 (2019)
20. Zhang, D., Kifer, D.: LightDP: towards automating differential privacy proofs. In: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, pp. 888–901 (2017)