USENIX Security '25 Artifact Appendix: General-Purpose f-DP Estimation and Auditing in a Black-Box Setting

Önder Askin[1], Holger Dette[1], Martin Dunsche[1,*], Tim Kutta[2], Yun Lu[3], Yu Wei[4], Vassilis Zikas[4†]

[1]Ruhr-University Bochum
[2]Aarhus University
[3]University of Victoria
[4]Georgia Institute of Technology

A Artifact Appendix

A.1 Abstract

In this paper we propose new methods to statistically assess f-Differential Privacy (f-DP), a recent refinement of differential privacy (DP) that remedies certain weaknesses of standard DP (including tightness under algorithmic composition). A challenge when deploying differentially private mechanisms is that DP is hard to validate, especially in the black-box setting. This has led to numerous empirical methods for auditing standard DP, while f-DP remains less explored. We introduce new black-box methods for f-DP that, unlike existing approaches for this privacy notion, do not require prior knowledge of the investigated algorithm. Our procedure yields a complete estimate of the f-DP trade-off curve, with theoretical guarantees of convergence. Additionally, we propose an efficient auditing method that empirically detects f-DP violations with statistical certainty, merging techniques from non-parametric estimation and optimal classification theory. The artifact will demonstrate the effectiveness for a range of DP mechanisms. The plug-and-play design allows easy replacement of mechanisms and extension to new ones.

A.2 Description & Requirements

This artifact implements a black-box f-DP estimator using two approaches: a perturbed likelihood ratio (PLRT) test and a classifier-based Baybox estimator. The codebase is tested on an Ubuntu 22.04.5 LTS machine and requires Python 3 with venv, pip, and JupyterLab for execution, alongside R version 4.1.2 and the fdrtool package for subsampling experiments. Pretrained models are provided to avoid expensive retraining, but scripts are available to regenerate them if needed. Both the estimator and auditor can be evaluated using Jupyter notebooks located in the `results/` directory, which reproduce all key experiments and figures from the paper.

A.2.1 Security, privacy, and ethical concerns

There are no security, privacy or ethical concerns.

A.2.2 How to access

For long-term accessibility, an archived version is hosted on Zenodo at: https://doi.org/10.5281/zenodo.15599462. The link provides full access to the source code, pretrained models, result folders, and example notebooks needed to reproduce all experimental results. Please refer to the Zenodo record for a permanent, citable reference to the evaluated version of the artifact. Furthermore we update a GitHub-repository accessible at: https://github.com/stoneboat/fdp-estimation.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

The artifact requires a standard software stack including Python 3, R (version 4.1.2), and commonly used open-source packages:

- *Python libraries are listed in requirements.txt .*
- *R requires fdrtool .*

A.2.5 Benchmarks

The artifact includes precomputed outputs and models used in the experiments reported in the paper:

- ***Pretrained models** used for the CIFAR-10 experiments are provided in the `trained_models/` directory.*

*Corresponding author: martin.dunsche@rub.de
†Authors are listed in alphabetical order.

While training from scratch is computationally expensive, reproducibility is supported via scripts in `sgd_experiment_cnn/`.

- ***Benchmark results*** *for Figures 1–12 are stored in the* `results/` *directory, alongside Jupyter notebooks for reproducing f-DP curve estimation and auditing.*

A.3 Set-up

A.3.1 Installation

The artifact is tested on Ubuntu 22.04.5 LTS, and installation has been verified on a Google Cloud VM instance. The following steps set up both the Python and R environments:

1. **System update (optional but recommended):**

```
sudo apt update sudo apt upgrade -y
```

2. **Install Python environment:**

```
sudo apt install python3-venv python3-pip
python3 -m venv fdp-env
source fdp-env/bin/activate
pip install -upgrade -r requirements.txt
```

3. **Install R for Subsampling experiments:**

```
sudo apt install r-base -y sudo R
```

Inside the R console:

```
install.packages("fdrtool")
```

4. **Configure Jupyter for interactive use:**

```
python -m ipykernel install -user
-name=fdp-env -display-name "Python
(fdp-env)"
jupyter lab -ip=0.0.0.0 -port=8888
-no-browser
```

A.3.2 Basic Test

A simple functionality test can be run using the provided script for constructing result folders and generating estimators. For example, to generate 10 estimators for the Gaussian mechanism with a sample size of 1000:

```
python3 construction_of_results_folders.py
Gaussian 1000 10
```

This command creates a subfolder with results in the `results/results_Gaussian/` *directory. Successful output includes printed confirmation of estimator generation and creation of result files.*

A.4 Evaluation workflow

This section outlines the steps to validate that the artifact is functional and reproduces the key claims in the paper.

A.4.1 Major Claims

(C1): *The PLRT estimator provides a uniformly convergent estimate of the f-DP curve across mechanisms and sample sizes. This is demonstrated by experiment (E1) and supported by results in Figures 1–3,9-10.*

(C2): *The Baybox auditor can reliably test a claimed privacy guarantee by checking for violations of f-DP. This is demonstrated in experiment (E2) and illustrated in Figure 4-5, 12.*

(C3): *The estimators and auditing procedure are applicable to real-world scenarios, such as evaluating trained DP-SGD models on CIFAR-10. This is demonstrated by experiment (E3) and supported by Figure 6-8.*

A.4.2 Experiments

(E1): *PLRT Estimator Evaluation [5 human-minutes + 1-120 compute-minutes per example, depending on the example]: Evaluate the PLRT estimator for various mechanisms (Gaussian, SGD, ...) and varying sample sizes. We recommend evaluating only one sample size, e.g. $n_1 = 10^4$.*

Preparation: Set up the environment and generate results using (e.g. Gaussian):

```
python3 construction_of_results_folders.py
<mechanism> <n_1> <repetitions>
```

Execution: Open the notebook `PLRT_shade_plots.ipynb` in `results/Figure2_3_9_10/` and run the cells to compute and visualize estimated f-DP curves. For Figure 1 open `visualize_MSE.ipynb` in `results/Figure1/` and execute file.

Results: If all mechanisms and all samples size (takes a lot of time) were run, the jupyter notebook includes all estimated f-DP curves, shaded error plots, and comparisons to theoretical reference curves. These should replicate Figures 2-3,9-10.

(E2): *Baybox Auditor Test [1 human-minutes + 10 compute-minutes per Figure]: Evaluate the Baybox classifier-based auditor to test a privacy claim.*

Preparation: Use existing audit-ready folders in `results/Figure4/` and rerun the jupyter.

Execution: Open and execute `auditing_correct_mechanisms.ipynb` and/or `auditing_faulty_mechanisms.ipynb` in `results/Figure5/`, which runs the whole auditing process and checks if the privacy claim is violated.

Results: The notebook outputs an audit decision and

plots showing the estimated rejection region. This validates results like those in Figure 4 and Figure 5.

(E3): *DP-SGD Audit on CIFAR-10 [5 human-minutes + 15 compute-minutes]: Apply the PLRT estimator and Bay-box auditor to real-world DP-SGD models trained on CIFAR-10.*

Preparation: Use pretrained models in `trained_models/`, or regenerate them with: `./run_generate_sgd_models`
Please specify the NUM_TRAIN_SAMPLES (default 2) and NUM_WORKERS (also default 2) inside the file (this takes a lot of time –depending on the NUM_WORKERS and storage space).

Execution: Open notebooks in Figure6, Figure7, Figure8 folders (e.g., `lowerbound_DPSGD_different_loss.ipynb`) in the corresponding figure folder and apply the auditor or estimator on model predictions.

Results: The notebook outputs estimated or audited privacy guarantees. We replicate Figure6-8.

All experiments are expected to replicate the corresponding figures and validate the main findings of the paper. Minor deviations may occur due to randomness in sampling or non-deterministic model execution.

A.5 Notes on Reusability

This artifact is implemented in a modular and extensible way, enabling its reuse beyond the specific experiments reported in the paper.

The core estimator and auditor implementations are designed to be plug-and-play. Users can easily substitute one mechanism for another by modifying the input arguments to the estimator scripts or by generating their own model outputs. Specifically:

- New mechanisms (e.g. subsampled Laplace) can be evaluated by generating samples and formatting them in the same structure as the provided `results_mech/` folders.
- The `construction_of_results_folders.py` script allows users to easily scale up the number of estimators or change sample sizes for empirical experiments.
- The estimators are agnostic to the data distribution used; with minor preprocessing adjustments, users can audit mechanisms trained on datasets beyond CIFAR-10.

The artifact can also serve as a testing ground for new privacy estimation methods or auditing strategies by modifying or extending the implementations in the `src/` folder.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.