

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

The Adverse Effects of Omitting Records in Differential Privacy: How Sampling and Suppression Degrade the Privacy–Utility Tradeoff

Àlex Miranda-Pascual, *Kalrsruhe Institute of Technology and Universitat Politècnica de Catalunya;* Javier Parra-Arnau, *Universitat Politècnica de Catalunya;* Thorsten Strufe, *Karlsruhe Institute of Technology*

https://www.usenix.org/conference/usenixsecurity26/presentation/miranda-pascual

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology



USENIX Security '26 Artifact Appendix: The Adverse Effects of Omitting Records in Differential Privacy: How Sampling and Suppression Degrade the Privacy–Utility Tradeoff

Àlex Miranda-Pascual
Karlsruhe Institute of Technology
Universitat Politècnica de Catalunya
alex.pascual@kit.edu

Javier Parra-Arnau
Universitat Politècnica de Catalunya
javier.parra@upc.edu

Thorsten Strufe
Karlsruhe Institute of Technology
thorsten.strufe@kit.edu

A Artifact Appendix

A.1 Abstract

The artifact contains the code necessary to reproduce all of the experiments and plots presented in the paper. The main folder is divided into six subfolders, each covering a different independent experiment. In addition to the main `README.md` file, each folder contains a specific `README.md` for that experiment. The six experiments are written in Python.

The first four folders cover the main paper evaluation for different mechanisms. Each folder covers the utility evaluation and comparison of a mechanism with and without sampling (Section 3), and with and without outlier-score suppression (Section 6). These folders are:

(E1). `NoisyAverage`, which covers the mean computation with NoisyAverage.

(E2). `ReportNoisyMax`, which covers the mode computation with report noisy max and the exponential mechanism.

(E3). `Clustering-DPLloyd`, which covers the DPLloyd clustering algorithm.

(E4). `Clustering-kmedian`, which covers on the k-median clustering algorithm.

In addition, we have two additional folders:

(E5). `PrivacyBound`, which checks that the empirical results obtained match our theorized values for Theorem 5.4.

(E6). `PrivacyBoundPlots`, which generates the remaining paper plots: Figures 2 and 4 (or 75 of the long version).

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our code consists of Python scripts that require only standard packages. We are not aware of any malicious content or risks associated with downloading and using our artifact. Our artifact is unable to collect any information from its users.

Our work uses either synthetic databases or commonly used, publicly available databases derived from US and Irish census data. These databases have all been previously and thoroughly anonymized and are explicitly released for public research, and statistical and educational use. Our artifact does not attempt to deanonymize or reidentify participants or their data. We include the database sources in the `README.md`.

We acknowledge the potential biases that can accompany any empirical evaluation and that our paper conclusions may differ for other scenarios, suppression algorithms, or variants of differential privacy (DP). We include the exact CSV files and plots of the execution presented in the paper since the code contains randomization.

A.2.2 How to access

The artifact is permanently hosted at Zenodo:
https://zenodo.org/records/17977527.

A.2.3 Hardware dependencies

We ran our experiments on a server equipped with an AMD EPYC 7702P 64-Core Processor. We note that approximately 1.7 GB of RAM is sufficient to run our code. Experiments **(E1)** to **(E5)** are parallelized, and the default number of processes corresponds to the number of CPU cores on the user's machine. This number can be modified in the `config.py` file. We ran our experiments with a parallelization with 64 cores.

A.2.4 Software dependencies

We run our code on Ubuntu 24.04. The code is written entirely in Python 3.8.20. The code requires the `sage` package

and its dependencies, as well as `matplotlib`, `numpy`, `pandas`, `scipy`, and `tqdm`.

For our experiments, we work with Conda and its environments. We recommend following their user guide to install Conda: https://docs.conda.io/projects/conda/en/stable/user-guide/index.html. We used the version `conda 24.7.1`.

A.2.5 Benchmarks

The artifact size is around 2.5 GB, as it contains the databases tested and the output from one run of each the experiment. Running the experiments again with the default databases and parameters will not require additional disk space.

A.3 Set-up

A.3.1 Installation

We provide an `environment.yml` file in the main folder of the artifact to easily set up an environment with all the necessary dependencies. After downloading the artifact, one can set up and activate the environment with the following commands:

```
conda env create -f environment.yml
conda activate SamplingAndSuppression
```

A.3.2 Basic Test

After setting up the environment, it is only necessary to run the main Python script in each folder to obtain the results for the variables in the paper. This can be done using the following command:

```
python main.py
```

No additional input parameters are required to obtain the paper results. The expected outputs are as follows:

(E1)–(E4): Each experiment returns a folder containing the CSV files with the numerical statistics and the respective plots of sampling and outlier-score suppression evaluations. Each folder contains a `README.md` file describing the output files and their location.

(E5): Each individual script returns a CSV file containing the computer-found (`DiffEvol`) and hypothesized (`HypValue`) values, and their difference (`Difference` = `DiffEvol` − `HypValue`). The script is designed to output error messages in the terminal when a value would contradict our claimed result.

(E6): The script returns the plots in Figures 2 and 4 (Figures 2 and 75 of the long version).

A.4 Evaluation workflow

A.4.1 Major Claims

The major claims of our paper are as follows (only including claims that require experimentation, not including those that are theoretical or from mathematical theorems):

(C1): We conducted an evaluation on uniform Poisson sampling over classic unbounded approximate DP mechanisms. This evaluation reveals that, for fixed privacy levels, the utility guarantees of the DP mechanism with sampling are worse than those of the mechanism without sampling for the tested databases.

(C2): We empirically show that, even when factoring in the privacy amplification, outlier-score suppression in DP worsens the privacy–utility tradeoff analogously to sampling for our tested mechanisms. Despite both techniques providing unfavorable outcomes, our results show that outlier-score suppression rarely outperforms sampling.

In addition,

(C3): We computationally verify that the value for ε^S given in Theorem 5.4 is correct up to an error of $2 \cdot 10^{-7}$.

A.4.2 Experiments

We group Experiments **(E1)** to **(E4)** together because they all serve to justify Claims **(C1)** and **(C2)**.

(E1): **`NoisyAverage`** [5 human-minutes + 4.33 compute-hours[1] + 1.2 GB disk[2]]: Covers the mean computation with the NoisyAverage mechanism.

(E2): **`ReportNoisyMax`** [5 human-minutes + 10.5 compute-hours[1] + 1.1 GB disk[2]]: Covers the mode computation with report noisy max and the exponential mechanism.

(E3): **`Clustering-DPLloyd`** [5 human-minutes + 46 compute-hours[1] (slightly less than 2 days) + 50 MB disk[2]]: Covers the DPLloyd clustering algorithm.

(E4): **`Clustering-kmedian`** [5 human-minutes + 24 compute-hours[1] + 3.5 MB disk[2]]: Covers the k-median clustering algorithm.

These four experiments cover our entire evaluation of sampling (related to Claim **(C1)**) and outlier-score suppression (related to Claim **(C2)**). In these experiments, both the sampling and outlier-score suppression counterparts are run simultaneously because the former is a case of the latter.

Preparation: For each experiment, navigate to the corresponding folder and set up the environment[3].

Execution: For each experiment, run the `main.py` file.

Results: Each of these experiments generates a folder for each tested database. Each folder contains a subfolder with the CSV files (containing the numerical values from the experiment) and the plots. Each of these subfolders includes one or more additional subfolders with the names of the tested columns. Note that these

[1] The runtimes take into account our parallelization with 64 cores.

[2] These values correspond to respective experiment disk space. Note that we provide the output of a full run in the artifact and, therefore, running the experiment again will not require any (significant) additional disk space.

[3] All experiments can be run with the same environment, which is also contained in the main folder for convenience.

Expt.	Claim	File names
(E1)	**(C1)**	`[column]_uniform_Poisson_sampling_[laplace/gaussian]_MPE+Sd.pdf`
(E1)	**(C2)**	`[column]_eps=[epsilon]_delta=[delta]_difference_[laplace/gaussian]_M_minus_MoSChangeEpsDelta_MPE_10--90.pdf`
(E2)	**(C1)**	`[column]_uniform_Poisson_sampling_[laplace/gaussian/exponential/exponential_mechanism]_EmpProb+Sd.pdf`
(E2)	**(C2)**	`[column]_eps=[epsilon]_delta=[delta]_difference_[laplace/gaussian/exponential/exponential_mechanism]_error_M_minus_MoSChangeEpsDelta_10--90.pdf`
(E3)	**(C1)**	`[columns]_uniform_Poisson_sampling_Average+SD.pdf`
(E3)	**(C2)**	`eps=[epsilon]_difference_error_M_minus_MoSChangeEpsDelta_Average_10--90.pdf`
(E4)	**(C1)**	`[columns]_uniform_Poisson_sampling_Average+SD.pdf`
(E4)	**(C2)**	`eps=[epsilon]_difference_error_M_minus_MoSChangeEpsDelta_Average_10--90.pdf`

Table 1: Name of plots that verify the claims **(C1)** and **(C2)** for the experiments **(E1)**, **(E2)**, **(E3)**, and **(E4)**.

folders contain more CSV files and plots than are shown in the paper, covering additional data that may interest some readers. For more details on these plots, refer to the `README.md` file. The names of the specific plots used in the paper to support claims **(C1)** and **(C2)** are shown in Table 1. For convenience, the script also provides copies of only those relevant plots into the `PaperPlots` folders.

The sampling plots (those with the **(C1)** tag in Table 1) show the utility values of the mechanism with and without sampling under the same privacy guarantees. Clearly, the mechanism with sampling provides worse utility for almost all sampling rates, as seen is evident across all plots[4]. This result confirms Claim **(C1)** for our study.

The outlier-score suppression plots (those with the **(C2)** tag in Table 1) directly show the difference in utility values between the mechanism with and without outlier-score suppression. A negative value indicates that the mechanism utility with outlier-score suppression is worse than that without it. All of our plots show negative values for nearly all points[4], which confirms Claim **(C2)**. In addition, we can see by that the diagonal, representing sampling, usually has smaller absolute values than other nearby values (more precisely, when compared to those values with the same proportion of deleted records). This suggests that outlier-score suppression rarely outperforms sampling for our tested mechanisms and databases.

In addition, the `ViewPaperPlots.html` file in the main folder allows the reader to easily find and open all the figures shown in the paper and its long version.

[4]The modes of the `hours-per-week` column of the Adult database and the `HighestEducationCompleted` column of the Irish database correspond to around half of the records in each database. Therefore, the modes are thus extremely robust to noise, and the respective DP mechanisms (with and without sampling and outlier-score suppression) return no empirical error. Nevertheless, we still included the plots for completeness. Please refer to the paper for more details.

(E5): `PrivacyBound` [1 human-minute + 17 compute-hours[1] + 200 MB disk[2]]: This experiment verifies that the ε^S expression we give in Theorem 5.4 is correct (up to an error of $2 \cdot 10^{-7}$) for the wide range of values of ε, m, and M as discussed in the paper. More precisely, our proof of Theorem 5.4 requires computational assistance to find the maximum of two functions over a domain defined by four or five continuous variables. We use this experiment to check that our theorized maximum is indeed the actual maximum by using optimization functions of `sage`. Please see the exact functions and more details in Remark 11 of the paper, and Remarks 22 and 24 of the long version.

Preparation: Redirect to the `PrivacyBound` folder and set up the environment[3].

Execution: Run the `main.py` file.

Results: Two CSV files are generated, one for each function, each containing the empirical value obtained by the computational optimization (`DiffEvol`), our hypothesized value given by our formula (`HypValue`), and the difference (`Difference` = `DiffEvol` − `HypValue`). In addition, the terminal outputs an error message if `DiffEvol` is larger than `HypValue` (up to some floating error), which would contradict our result that `HypValue` is the true maximum. Additional error messages are also output if the maximum is obtained in an unexpected case, which would also contradict our claim. Our experiment execution did not provide any errors, supporting Claim **(C3)**. Since our hypothesized value is within the maximization domain, we are only interested in measuring when the computational value exceeds our hypothesized value. In this case, the error of less than $2 \cdot 10^{-7}$ (this value is displayed in the terminal at the end of the execution).

(E6): `PrivacyBoundsPlot` [1 human-minute + 10 compute-seconds + 600 KB disk[2]]: This experiment gen-

erates Figures 2 and 4 in the paper (Figures 2 and 75 in the long version).
Preparation: Redirect to the `PrivacyBoundsPlot` folder and set up the environment[3].
Execution: Run the `main.py` file.
Results: The outputs are the plots used in the aforementioned figures. We included this code to confirm that our plots precisely describe the values given by our formulas.

A.5 Notes on Reusability

Experiments **(E1)** to **(E4)** can be run with different parameters and variables (e.g., ε, δ, number of iterations, number of clusters). To do so it suffices add arguments to the execution command. Please refer to the corresponding `README.md` file for the precise arguments necessary, which vary by experiments. Different databases can also be tested as long as they are saved as CSV files with labeled columns (i.e., in the same format as our tested databases).

The evaluations on sampling and outlier-score suppression can be run for new mechanisms, but most files in the folder must be redefined and adapted to these mechanisms. Certain files, such as `suppression_privacy_parameters.py`, should not vary between mechanisms.

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/usenixsec2026/.