USENIX Security '24 Artifact Appendix: Closed-Form Bounds for DP-SGD against Record-level Inference

Giovanni Cherubin*
Microsoft Security Response Center

Boris Köpf
Microsoft Azure Research

Andrew Paverd
Microsoft Security Response Center

Shruti Tople
Microsoft Azure Research

Lukas Wutschitz
Microsoft M365 Research

Santiago Zanella-Béguelin*
Microsoft Azure Research

A Artifact Appendix

A.1 Abstract

Our paper presents a new approach to evaluate the privacy of machine learning models against specific record-level threats, such as membership inference (MI) and attribute inference (AI), without the indirection through differential privacy (DP). We focus on the popular DP-SGD algorithm and derive closed-form bounds for the Bayes Security metric. Our artifacts allow reproducing *all figures* and *numerical results* that appear in the accepted paper. The code is split into two parts: `Part-A` contains experiments to assess the quality of our theoretical bounds and compare them with prior work; `Part-B` contains experiments to evaluate our bounds on real datasets. The artifacts consist of publicly accessible source code hosted on GitHub.

A.2 Description & Requirements

`Part-A` is provided as a `Jupyter` notebook as this is a commonly used format for running these types of experiments. `Part-B` is implemented as an extension of an example script provided with the `Opacus` library; we made this choice to illustrate how one might incorporate our methods into an existing library. Together, both parts reproduce the following figures and numerical results:

Result	Code
Figures 2-7	`Part-A`
Numerical results in Section 4 (*DP-SGD parameters selection*), Section 4.2 (*TPR@FPR*), Section 6 (*Privacy parameters*)	`Part-A`
Figures 1, 8, and 9	`Part-B`

*Corresponding author.

In addition to code for reproducing our experiments (`Part-A`, `Part-B`), we provide a web-based interactive tool, publicly accessible at `https://microsoft.github.io/dpsgd-calculator`. As described in Section A.5, this tool is based upon one of our manuscript's results and it exemplifies how one might use them in practice. *This tool is not needed to reproduce our main claims in this artifact; it is an additional byproduct of our work illustrating the use of our results in practical scenarios.*

A.2.1 Security, privacy, and ethical concerns

We do not anticipate that executing our artifacts will lead to any increased risk of security, data privacy, or ethical concerns. All data used in our evaluations is either artificial data or drawn from public datasets.

A.2.2 How to access

The code for reproducing our experiments is available at: `https://github.com/microsoft/dpsgd-calculator/tree/f0222fa9308e6b65b006ea15680698da70e08951/submission-code`.

A.2.3 Hardware dependencies

`Part-A`. Commodity hardware is sufficient for running the code contained in the `Jupyter` notebook.

`Part-B`. These experiments require training machine learning models. While it may be possible to do this on CPU, an accelerator is recommended for performance reasons. We tested this on NVIDIA Tesla K80 and V100 GPUs, but other accelerators supported by PyTorch could be used instead.

A.2.4 Software dependencies

Experiments were run on `Ubuntu 22.04 LTS`, with `Python 3.10`. Python dependencies are specified in a `requirements.txt` file, as explained below.

A.2.5 Benchmarks

`Part-B` uses the `Purchase-100`[1] and `Adult`[2] datasets. Our code automatically fetches and processes the required data.

A.3 Set-up

A.3.1 Installation

Start by cloning the repository and checking out the stable commit hash of the artifact.

```
git clone \
  https://github.com/microsoft/dpsgd-calculator
cd dpsgd-calculator
git reset --hard f0222fa
```

Please ensure that you use an environment with `Python 3.10`. For `Part-B`, if you are using an NVIDIA GPU, ensure that NVIDIA drivers and CUDA are properly set up. The latter can be checked by running `nvidia-smi` from a terminal.

From the `submission-code` folder, install the pinned Python dependencies by running:

```
pip install -r requirements.txt
```

To avoid any conflicts with local installations, you may want to do this in a virtual environment (e.g., using venv or Conda).

A.3.2 Basic Test

If using an NVIDIA GPU, you may want to check that `PyTorch` is able to use it. You can do this by checking that the following Python program prints `True`:

```
import torch
print(torch.cuda.is_available())
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1, `Part-A`): A mixture of Gaussian distributions as described in the paper (Proposition 4) can be approximated by a Gaussian distribution, with an error that depends on the parameters p, T, σ. In the specific case $T = 1$, we can compare our error (upper) bound to the actual error, and show that it overestimates the actual error (Figure 2). This figure confirms that our error estimate is indeed an upper bound for the true error.

(C2, `Part-A`): We demonstrate that the privacy estimates produced by our proposal match existing techniques in computing bounds for membership inference (MI) (sub-claim **C2.a**), while requiring orders of magnitude less computation time than prior work (sub-claim **C2.b**).

(C3, `Part-B`): We use our new approach to compute bounds for attribute inference (AI), and observe that DP-SGD is significantly more secure against AI than MI. This is important because, if an application requires security against AI but not MI, one can achieve a better utility whilst maintaining acceptable privacy (Figure 1).

A.4.2 Experiments

`Part-A`: experiments E1-E2

Experiments E1-E2, supporting the respective claims C1-C2, are based on the same `Jupyter` notebook. They have the following steps in common:

Preparation: The code for these experiments is in `submission-code/`. Follow the instructions in Section A.3 to install `Jupyter` and all other `Python` requirements.

Execution: From the directory `submission-code/`, start a `Jupyter` instance from the terminal, by running `jupyter notebook`. Open the link that is shown (normally, `https://localhost:8888`) and open the `evaluation.ipynb` notebook. Starting from the top of the notebook, execute all the cells until the sections specified for E1 and E2 below.

(E1): *Gaussian approximates a Gaussian Mixture (1 human-minutes + 2 compute minutes + negligible disk): This experiment computes the approximation error committed by approximating a Gaussian Mixture distribution with a single Gaussian via numerical integration, and it compares it to the upper bound proven in Proposition 4.*
Execution: Execute all the cells until (and including) the section *"Figure 2: When is approximation of mixture to Gaussian a good approximation?"*.
Results: The resulting figure matches Figure 2.

(E2): *Comparison to PLD accountant (1 human-minutes + 5 compute minutes + negligible disk): This experiment compares our MI risk calculation method to the PLD accountant. The former is based on a closed-form expression. The latter estimates the DP parameters (ε, δ), and from them we obtain the MI risk by setting $\varepsilon \approx 0$. Our first claim (C2.a) is that the two methods produce similar estimates for a wide range of parameters. Our second claim (C2.b) is that our method is orders of magnitude faster than the PLD accountant.*
Execution: Run all the cells until (and including) section *"Figure 4: comparison with PLD"* for claim C2.a; execute until (and including) *"Figure 5: Computational efficiency comparison"* for claim C2.b.
Results: The first part produces an identical figure to Figure 4, and it also produces all the figures that are

[1] https://github.com/privacytrustlab/datasets/blob/master/dataset_purchase.tgz

[2] https://hf.co/datasets/scikit-learn/adult-census-income

present in Figure 10 in Appendix B in the paper. Since the PLD accountant we compare against does not work well for $p = 0.0001, \sigma = 0.5$, expect a number of errors to be reported when running the first cell. The second part produces a figure that is equivalent to Figure 5. *Note: the figures may not be identical, since this experiment depends on timing measurements. However, we expect that the claim that "our method is orders of magnitude faster" should hold regardless of the computational hardware.*

(E2-extra): *Using Corollary 7 to pick DP-SGD hyperparameters (1 human-minutes + 1 compute minutes + negligible disk): This experiment shows how to use Theorem 6 to guarantee a certain level of Bayes security against MI attacks. It produces Figure 3 in the paper.*
Execution: Evaluate all the cells in the section *"Figure 3: Membership inference bound from Corollary 7"*.
Results: A figure is produced that matches Figure 3 in the paper. The relation $p \approx 0.00035\sigma$ reported in the *DP-SGD parameter selection* paragraph in the paper is computed from Equation (4). The choices of σ for the Adult and Purchase-100 experiments in Section 6 are similarly derived. A plot not included in the paper illustrating how Bayes security varies with the number of DP-SGD steps for different values of σ is produced.

(E2-extra): *Relation between Bayes security and MI advantage and TPR@FPR metrics (1 human-minutes + 10 compute minutes + negligible disk): This experiment compares the Bayes security metric to other MI metrics.*
Execution: Run all the remaining cells in the notebook.
Results: This produces the plots in Figures 6 and 7 in the paper. The last cell in the notebook uses the PLD accountant to numerically compute the membership inference TPR@FPR for some exemplary values and compares it to approximations via Bayes security.

`Part-B`: experiment E3

(E3): *Security comparison w.r.t. MI and AI (10 human minutes + 3 compute-hours + 1.2GB disk): In our paper, security against MI is measured in terms of worst-case data and security against AI is measured in a data-dependent fashion (i.e., specific to the given dataset). In this experiment, we compare these two security evaluations.*
Preparation: The code for these experiments (`Part-B`) is in `submission-code/real-data-experiments/`. Please follow the instructions in Section A.3 to install `Jupyter` and all other `Python` requirements.
Execution: From a command line, run

```
bash launch-adult.sh
bash launch-purchase100.sh
```

This will create 5 log files (one per run) for each dataset: `adult-approximate-{1..5}.jsonl` and `purchase100-{1..5}.jsonl`. To plot the results, open the `Jupyter` notebook `plot_results.ipynb`. To produce Figure 1 (which substantiates claim C3), run all the cells in the notebook until (and including) the section *"Privacy vs utility"*.
Results: The plot obtained when running the notebook should be similar to the one appearing in Figure 1 in our paper; slight differences may be present due to noise in the training process, but no major difference is to be expected. The results indicate that, according to the AI analysis, the model can be considered more secure for higher levels of utility (model's accuracy). This is attributable to two factors: i) it is (provably) easier to protect against AI attacks than MI; ii) measuring security for a specific dataset (*data-dependent* analysis) can provide utility advantages compared to a worst-case data study (*data-independent* security analysis).

A.5 Notes on Reusability

Beyond reproducing our experiments, we hope the techniques described in this paper will be adopted in practice.

Interactive tool: We implemented an interactive tool that uses our closed-form bounds to visualize the security of a model trained with DP-SGD against MI with chosen hyperparameters. We host this tool as a static webpage at https://microsoft.github.io/dpsgd-calculator and make available its code at https://github.com/microsoft/dpsgd-calculator/tree/gh-pages.

Integration into existing pipelines: Our MI and AI analysis techniques can be integrated into training pipelines without requiring modification of existing libraries. For example, our code in `Part-B` shows how to extend a training script from the popular `Opacus` library. This did not require any changes to the `Opacus` code base. Similar approaches can be used to integrate our techniques into other training pipelines.

Integration into `Opacus`: It is also possible to integrate our techniques directly into the `Opacus` library. `Opacus` currently integrates its DP accountant functionality via the `PrivacyEngine` class. Internally, an object of this class is updated by a `PyTorch`'s optimizer at every DP-SGD step, enabling estimation of the respective (ε, δ) values. A similar approach could be used to integrate our techniques into `Opacus`. Since our data-dependent AI analysis requires access to the training data (batch), the `PrivacyEngine` class would need to be extended to provide access to batch data.

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/usenixsec2024/.