USENIX Security '25 Artifact Appendix: Further Study on Frequency Estimation under Local Differential Privacy

Huiyu Fang
Southeast University

Liquan Chen
Southeast University

Suhui Liu
Southeast University

A Artifact Appendix

A.1 Abstract

The artifact consists of Python-based code and scripts designed to evaluate LDP frequency protocols using synthetic and real-world datasets. The artifact comprises three main components: (1) the source code of all LDP frequency protocols evaluated in the paper, including the baseline and proposed protocols, (2) the scripts to conduct the experiments and plot the results, and (3) a comprehensive README file. With all these components, the results of our paper can be reproduced.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None.

A.2.2 How to access

The artifact is publicly available on Zenodo at https://zenodo.org/records/14715748. Please use the latest version of this record.

A.2.3 Hardware dependencies

The artifact is compatible with any commodity computer with a minimum of 16GB of RAM and requires approximately 300MB of disk space. The memory requirement mainly comes from the runtime experiment and more RAM is required if the customized domain size in the runtime experiment is larger than 4096. Multi-core CPUs can significantly speed up the experiments. We run the experiments on a PC with AMD Ryzen 9 7950X and 64GB memory.

A.2.4 Software dependencies

The artifact is developed in Python 3.10.4 and should be compatible with recent Python versions. We run the experiments on Windows, but Python is a cross-platform language, which means it can run on any OS with Python installed, including Windows, macOS, and Linux. The artifact requires only four Python packages, including NumPy, xxHash, PyArrow, and Matplotlib. All these packages can be installed from PyPI using pip. The *requirements.txt* file is provided for convenient installation.

A.2.5 Benchmarks

The real-world data set is from the NYC Taxi and Limousine Commission and is integrated into the artifact.

A.3 Set-up

A.3.1 Installation

Before installation, a recent Python environment should be created. We recommend Python 3.10.4. After downloading and unzipping the artifact, enter the artifact folder and run the following command in the terminal:

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

The command will install all dependencies required by the artifact.

A.3.2 Basic Test

Once the installation is completed, users can run the following command in the terminal to verify that all required software components are functioning properly:

```
python test.py
```

The script will run all unit tests and print a success message in the terminal after passing each unit test.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *The empirical MSE matches our analytical MSE of all frequency protocols in the paper. This is proven by the experiment (E1) described in Section 5.2 whose results are illustrated in Figure 4.*

(C2): *RUE and RLH achieve better MSE or accuracy than OUE and OLH, especially for the small domain and top frequent values. RWS achieves the same optimal accuracy as SS but with less communication costs for large domains. This is proven by the experiments (E1) described in Section 5.3 whose results are illustrated in Figures 5 and 6.*

(C3): *The aggregation time of RLH and RWS are much smaller than that of OLH for large domains. This is proven by the experiments (E2) described in Section 5.4 whose results are illustrated in Figure 7.*

A.4.2 Experiments

We give complete instructions for users to run all experiments and plot all experimental results in the paper in the README file. More details, especially the experimental parameters, can be found in the README file.

(E1): *main.py [approximately 105 compute-hour + 250MB disk]: This experiment is to achieve the empirical MSE of all frequency protocols on both synthetic and real-world datasets with a specified range of* ε *and domain size.*

Preparation: We default to running the experiment 100 times to take the average MSE but very time-consuming. For quick verification, users can customize the parameter *run_repeat_time* in the *main.py* and *drawTopk.py* files to a small number (e.g. 10), but the results for small domains would fluctuate more. The ranges of *run_d_range* and *run_epsilon_range* can also be reduced.

Execution: Enter the artifact folder and run the following command in the terminal:

```
python main.py
```

Experimental results will be saved in the *results* folder. More specifically, each estimated frequency result will be saved in the *results/protocol_name* folder and the summary of average empirical MSE results will be saved in CSV files.

Results: After the main experiment, plot Figures 4, 5, and 6 in the paper by running:

```
python drawAnaEmpMSE.py
python drawEmpMSE.py
python drawTopk.py
```

The plotted figures will be saved in the *draw* folder.

(E2): *runtime.py [approximately 6 compute-hour + 5.5GB RAM]: This experiment is to demonstrate the aggregation time of all frequency protocols in the paper.*

Preparation: Runtime and memory costs are linearly related to data size, so the taxi dataset requires approximately 36 times as much runtime and RAM as the synthetic dataset. For quick verification or if memory is limited, users can run the experiment only on the synthetic dataset by commenting out lines from 156 to 158 of the *runtime.py* code and lines from 66 to 79 of the *drawRuntime.py* code.

Execution: Enter the artifact folder and run the following command in the terminal:

```
python runtime.py
```

The summary of runtime results will be saved in CSV files in the *results* folder.

Results: After the runtime experiment, plot Figures 7 in the paper by running:

```
python drawRuntime.py
```

The plotted figure will be saved in the *draw* folder. Additionally, Figure 5, 6, and 7 share the same legend, which is generated by *drawEmpMSE.py* and saved as *legend.eps* in the *draw* folder.

A.5 Notes on Reusability

The source code of frequency protocols is modularized for reusability. The Python-based scripts of experiments can be used as examples for users to conduct experiments on different datasets. However, to be more practical, the source code may need to be divided into client-side and server-side. More specifically, the perturb function should be implemented on the client side, while the aggregate function should be implemented on the server side.

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/.