USENIX Security '26 Artifact Appendix: High-Accuracy, Poisoning-Resilient Frequency Estimation in the Shuffle Model

Shaoqiang Wu[†], Jingyu Jia[†‡], Yikuan Zhu[†], Xinhao Li[†], Changyu Dong[§], Zheli Liu[†]

[†]CS&CCS, DISSec, AAIS, Nankai University, Tianjin, China

[‡]Automotive New Technology Research Institute of BYD, Shenzhen, China

[§]Guangzhou University, Guangzhou, China

A Artifact Appendix

A.1 Abstract

The artifact is a self-contained Python implementation of the paper's two histogram frequency-estimation protocols, $\mathcal{P}^{\mathrm{H1}}$ (small-domain) and $\mathcal{P}^{\mathrm{H2}}$ (large-domain, compressed), both built on the paper's symmetric binomial-sum noise design. It also includes the five baselines we compare against (LWY, CSUZZ, BC, CZ, GKMP), the aggregated input histograms, and three scripts that regenerate the data behind every empirical result (Table 2, Table 3, Figure 2, and the Appendix E.1/E.2 supplements that appear only in the extended version[1]). All randomness is explicitly seeded, and the result CSVs in the artifact are exactly the ones used to produce the paper's tables and figures. A `--quick` flag on each script provides a functional check that runs in seconds to about a minute; the unflagged scripts perform the full reproduction. The artifact runs on a single CPU core, with no GPU, no specialized hardware, and no network access.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact is non-destructive. It runs entirely in user space, requires no root privileges, makes no network connections, and modifies nothing outside its own `results/` directory (it only reads the provided input histograms and writes result CSVs). It contains no record-level personal data, since every dataset is used only as an aggregated count histogram. The AOL query log is a standard large-domain benchmark in this literature, and also a sensitive case whose 2006 raw release was retracted, so the artifact contains only a 3-character query-prefix histogram over 2^{24} keys, with no raw queries, user identifiers, or timestamps (see `data/README.md`). The poisoning module is a controlled-benchmark simulator used only to measure robustness; it is not a tool for attacking a deployed system.

A.2.2 How to access

The artifact is permanently archived on Zenodo under the concept DOI `10.5281/zenodo.20405418`, which always resolves to the latest version.

A.2.3 Hardware dependencies

None beyond a standard CPU. Any x86-64 machine suffices; the workloads are single-threaded with a resident-memory footprint of at most 4 GB. No GPU, accelerators, or specialized hardware are required.

A.2.4 Software dependencies

Python 3.11–3.14 (validated on 3.14) and the two scientific-computing packages pinned in `requirements.txt`, `numpy==2.4.4` and `scipy==1.17.1`. Both provide prebuilt binary wheels for CPython 3.11–3.14, so installation succeeds on a clean Ubuntu 24.04 LTS (Python 3.12) or Windows 11 machine through `pip` alone, without a compiler. Any `numpy` $\geq$ 1.26 with `scipy` $\geq$ 1.11 reproduces the reported numbers within the tolerance documented in the README.

A.2.5 Benchmarks

All input data is included; nothing is downloaded at run time. The four aggregated histograms in `data/` are `city.csv` (d=40, $n\approx$156k), `fire.csv` (d=272, $n\approx$681k), `occupation.csv` (d=529, $n\approx$123k), and `aol_3char_hist.npz` (d=2^{24}, subsampled to n=10^5). Provenance and licensing are documented in `data/README.md`.

[1]The extended version (https://eprint.iacr.org/2026/1218) reports MAE±SD and bit-level communication cost in its Appendix E.1, and the full poisoning grid in its Appendix E.2 (Figure 3); both are finer-grained views of the Table 2 and Figure 2 data, produced from the same result CSVs. The camera-ready version omits these two empirical supplements; its appendices contain the two protocol proofs and the anonymous message-control mechanism.

A.3 Set-up

A.3.1 Installation

From the artifact root, create a virtual environment and install the two dependencies:

```
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

On Windows, activate the environment with `venv\Scripts\activate` instead of the `source` line. No `PYTHONPATH` export is needed, because each script in `reproduce/` adds the artifact root to `sys.path` itself.

A.3.2 Basic Test

Run a quick functional check that exercises the full protocol pipeline (preprocessing, local randomizer, shuffler, analyzer) and all baselines:

```
python reproduce/table2.py --quick
```

This prints a six-row table for City at ε=1.0 and writes `results/quick/table2_quick.csv`. Each row reports, for one protocol, the mean absolute error (MAE), the 95th-percentile error, and the messages per user. The run is functioning correctly if it shows the expected qualitative pattern. Specifically, the two low-message protocols are Ours (≈ 1.5 messages per user) and LWY (≈ 1.0); in MAE, Ours $\leq$ LWY, while CSUZZ, BC, and CZ are several-fold higher; and GKMP reaches low error only at a very large message count (~28,700 messages per user for City).

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): On small-domain categorical data, $\mathcal{P}^{\mathrm{H1}}$'s MAE is essentially equal to that of the strongest bounded-message baseline (LWY) at ε=0.25 and lower for $\varepsilon \geq 0.5$, by up to nearly 2×, at comparable messages per user, and far below that of CSUZZ, BC, and CZ. GKMP reaches low error only with thousands to millions of messages per user, far above the per-user message limit this work targets. Proven by (E1); reported in paper Table 2 and Appendix E.1.

(C2): On the AOL large-domain workload, the compressed protocol $\mathcal{P}^{\mathrm{H2}}$ matches LWY's accuracy at the same hashed dimension d_h and comparable per-user communication. Proven by (E2); reported in paper Table 3.

(C3): The poisoning influence of $\mathcal{P}^{\mathrm{H1}}$ grows roughly linearly in the corrupted-user ratio m/n (up to 30%) and stays far below that of CSUZZ, BC, and CZ. Proven by (E3); reported in paper Figure 2 and Appendix E.2 (Figure 3).

A.4.2 Experiments

The result CSV shipped for each experiment holds the exact values behind the paper's tables and figures. Each full run is a single, unattended command that recomputes that CSV from scratch, overwriting it. The recomputed numbers reproduce the paper's tables and figures bit-for-bit on the pinned NumPy/SciPy versions (the CSUZZ and CZ columns of Table 2 to within one or two in the last printed digit, which affects no comparison in the paper), and within about 0.1% on MAE on any other `numpy` $\geq$ 1.26 (see the README). The `--quick` run instead writes to `results/quick/` and leaves the shipped CSVs untouched.

(E1): *Small-domain MAE* [≈5 human-minutes + 30–60 compute-minutes full, or ~5 s with `--quick`; <1 GB disk].
How to: Reproduce paper Table 2 (Claim C1). The full run evaluates all six protocols across the three small-domain datasets and six privacy levels.
Preparation: Complete the Set-up above; no per-experiment preparation is needed.
Execution: `python reproduce/table2.py` (full), or with `--quick`.
Results: Output `results/table2_with_p95.csv`. The MAE of Ours is at or below that of LWY for $\varepsilon \geq 0.5$ and essentially equal at ε=0.25, at comparable messages per user, and well below CSUZZ/BC/CZ. E.g., for City at ε=1.0, Ours 5.42 and LWY 5.96.

(E2): *AOL matched-d_h accuracy* [≈5 human-minutes + 2–3 compute-hours full, or ~30 s with `--quick`; <1 GB disk].
How to: Reproduce paper Table 3 (Claim C2). The full run compares $\mathcal{P}^{\mathrm{H2}}$ with LWY's P_{FE1} on the AOL histogram at two matched hashed dimensions d_h across six privacy levels.
Preparation: None beyond the Set-up.
Execution: `python reproduce/table3.py` (full), or with `--quick`.
Results: Output `results/aol_matched_b.csv`. `our_mae` ≈ `lwy_mae` and `our_msgs` ≈ `lwy_msgs` in each row at both d_h values. E.g., at d_h=65 and ε=1.0, Ours ≈ LWY ≈ 32.5.

(E3): *Poisoning influence* [≈5 human-minutes + ~5 compute-hours full, or ~1 min with `--quick`; <1 GB disk].
How to: Reproduce paper Figure 2 and Appendix E.2 (Claim C3). The full run measures each protocol's poisoning influence on the three small-domain datasets as the corrupted-user fraction m/n sweeps from 0.1% to 30%.
Preparation: None beyond the Set-up.
Execution: `python reproduce/figure2.py` (full), or with `--quick`.
Results: Output `results/poisoning_baselines.csv`. The influence of Ours and LWY stays far below CSUZZ/BC/CZ at every m/n, by roughly one to nearly

three orders of magnitude depending on the dataset and ε. E.g., for Fire at ε=1.0 and m/n=30%, Ours $\approx$ 1.0, LWY $\approx$ 0.87, CSUZZ $\approx$ 82.9, BC and CZ $\approx$ 163.

A.5 Notes on Reusability

Each `protocols/*.py` module is self-contained and can be imported independently of the runners. The modules are `small_domain.py` ($\mathcal{P}^{\mathrm{H1}}$), `large_domain.py` ($\mathcal{P}^{\mathrm{H2}}$ and LWY's P_{FE1}), `baselines.py` (the small-domain baselines), and `poisoning.py` (the poisoning simulator). New datasets are supplied as count histograms (one integer per category per line for the CSV format). The implementation approach, determinism, the per-script RNG seeds, and the cross-NumPy-version reproduction tolerance are documented in the top-level `README.md`.

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/.