USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Assuring Certified Database Utility in Privacy-Preserving Database Fingerprinting

Mingyang Song and Zhongyun Hua, *Harbin Institute of Technology, Shenzhen;* Yifeng Zheng, *The Hong Kong Polytechnic University;* Tao Xiang, *Chongqing University;* Guoai Xu, *Harbin Institute of Technology, Shenzhen;* Xingliang Yuan, *The University of Melbourne*

https://www.usenix.org/conference/usenixsecurity25/presentation/song-mingyang

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix: Assuring Certified Database Utility in Privacy-Preserving Database Fingerprinting

Mingyang Song[1], Zhongyun Hua[1], Yifeng Zheng[2], Tao Xiang[3], Guoai Xu[1], and Xingliang Yuan[4]

[1]*School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen*
[2]*Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University*
[3]*College of Computer Science, Chongqing University*
[4]*School of Computing and Information Systems, University of Melbourne*

A Artifact Appendix

A.1 Abstract

We provide the source code used for experimental evaluations, along with a roadmap for deploying and executing the source code. The open source code includes three parts: (i) the source code for dataset extraction and encoding, (ii) the source code of UtiliClear, and (iii) the source code for robustness and utility testing. The experiments in the paper involve evaluating efficiency, fingerprint robustness, and database utility. The efficiency evaluation results can be obtained during the execution of the source code of UtiliClear, while the results of fingerprint robustness and database utility can be obtained by executing the third part of the artifact.

A.2 Description & Requirements

This section outlines the required setups for executing the artifact. Before running the code, please set up the execution environment according to the following hardware and software configurations. Afterward, follow the execution steps presented in the artifact precisely to run the source code.

A.2.1 Security, privacy, and ethical concerns

The artifact involves neither malicious nor destructive operations. It poses no risk to machines security, data privacy or others ethical concerns for evaluators while executing the artifact.

A.2.2 How to access

The artifact and detailed deployment roadmap for executing UtiliClear are now available at Zenodo.

A.2.3 Hardware dependencies

Any testing environment is sufficient to reproduce only the overall efficiency trends. However, to reproduce the similar efficiency results in the paper, two desktops, each equipped with a 24-core Intel Core i9-13900 processor and 16GB of memory, are required to execute the source codes for the database owner and recipient, respectively.

A.2.4 Software dependencies

The source code rely on standard cryptographic libraries, including, OpenSSL, PBC, GMP, and MySQL. The dataset used for evaluation is available for download at https://nijianmo.github.io/amazon/index.html.

A.2.5 Benchmarks

We evaluate UtiliClear using the Amazon Review Data.

A.3 Set-up

Install Visual Studio 2015 and PyCharm 2021.3.1.

A.3.1 Installation

Before running the programs, the libraries of OpenSSL-3.0.4-Win32, PBC-0.5.14-Win32, GMP-Win32, and MySQL-5.1.51-Win32 should be installed and included in the programs

A.3.2 Basic Test

To run a simple functionality test, one can use the encoded subset (186 MB) along with its corresponding dictionary files provided in the artifact. With this small dataset, the first part (i.e., dataset extraction and encoding) can be skipped. Directly use the small subset as the input to execute the second part

(i.e., the source code of UtiliClear). Before executing the programs, we recommend setting the number of group to 400, the bit-length of insignificant bits to 1, and the number of modifiable bits to 8. Then run the source code following the execution roadmap provided by the readme file of the artifact. Both the code files 'Com_Verify_Recipient.cpp' and 'InsignificantBits_Verify3_Recipient.cpp' will output 'success', indicating that software components are correctly utilized and functioning fine.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): During the preprocessing and verification phases, the time costs for both the recipient and the database owner increase with the number of groups and the number of insignificant bits. In contrast, during the fingerprint embedding and extraction phases, the database owner's computation time increases with the number of insignificant bits but remains unaffected by the number of groups. This is demonstrated by the experiment (E1), whose results are reported in Figures 7-8.

(C2): During the preprocessing phase, the communication overhead increase with the number of groups and the number of insignificant bits. In contrast, during the verification phase, the communication overhead increases with the number of insignificant bits. This is demonstrated by the experiment (E1), whose results are reported in Figure 9.

(C3): More groups cause smaller data sizes per group, reducing memory requirements for both the database owner and the recipient. This is demonstrated by the experiment (E1), whose results are reported in Table 2.

(C4): UtiliClear achieves comparable fingerprint robustness and database utility with the state-of-the-art scheme. This is demonstrated by the experiment (E2), whose results are reported in Figures 12-14.

A.4.2 Experiments

(E1): It takes approximately 10 hours to obtain a single experimental result and several days are needed to obtain all the efficiency evaluation results presented in Section 6.2 using the suggested hardware configuration above.
How to: Use the first part of the source code to process the database, then take the encoded database as input to execute the second prat of the source code.
Preparation: Before executing the code, set the parameters (e.g., the ratio of removed records, the ratio of added records, and the ratio of flipping bits) in code files.
Execution: Use the first part of the source code to process the database and obtain encoded databases. Following the detailed roadmap described in the open artifact, execute the source code of UtiliClear to obtain the experimental results.
Results: Track the execution time of the code to obtain the computational time consumption. For communication consumption, monitor the files exchanged between the recipient and the database owner. To observe memory cost during code execution using Visual Studio.

(E2): It takes approximately two days to obtain the experimental results of fingerprint robustness. Four days are needed to obtain the results of database utility.
How to: Use the outputs (i.e., fingerprinted databases) of the source code of UtiliClear to evaluate database utility and fingerprint robustness by executing the third part of the source code.
Preparation: Before executing the code, set the parameters in each code file and ensure the consistency of parameter settings in all code files.
Execution: Following the detailed roadmap described in the open artifact, execute the third part code files to test the fingerprint robustness and database utility.
Results: Obtain the fingerprint robustness results by comparing the fingerprint extracted from the databases under various types of attack with the original fingerprint. To obtain the database query result, decode the fingerprinted database and import the decoded dataset and the original dataset into the MySQL database, and then test the SQL query results. One code file (i.e., 'Classifier_Training.py') is provided to train and evaluate the classifier's accuracy.

A.5 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/.