One-off Disclosure Control by Heterogeneous Generalization

Olga Gkountouna, *University of Liverpool;* Katerina Doka, *National Technical University of Athens;* Mingqiang Xue, *Tower Research;* Jianneng Cao, *Bank Jago;* Panagiotis Karras, *Aarhus University*

https://www.usenix.org/conference/usenixsecurity22/presentation/gkountouna

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

A Artifact Appendix

A.1 Abstract

This appendix describes the software artifact that implements and evaluates all algorithms proposed in this paper. Specifically, it provides Java implementations for the *Greedy*, *SortGreedy* and *Hungarian* algorithms for anonymization by both ℓ-diversity and β-likeness; it also contains implementations of the algorithms we compare against, namely NH (in C++), BuReL (in Java) and PrivBayes (in C++), as obtained by their authors and properly enhanced to record the same metrics.

No specialized hardware is required to reproduce the results of the paper; however, the anonymization of the largest dataset requires at least 64GB of RAM. We provide instructions on how to install the artifact, execute the experiments, and validate the results in the form of a README document that describes the process step by step. This is intended to help the reader reproduce the results presented in the paper. The artifact is available as a GitHub repository.

A.2 Artifact check-list (meta-information)

- **Algorithm:** We present three novel algorithms for disclosure control through syntactic anonymization based on the notion of heterogeneous generalization. These algorithms are (i) *Greedy*, denoted as GR, which employs an $O(n^2)$ heuristic for assignment extraction, (ii) *SortGreedy*, denoted as SG, which employs an $O(n^2 \log n)$ heuristic for tuple matching and (iii) *Hungarian*, denoted as HG, which utilizes the $O(n^3)$ Hungarian algorithm to build assignments. These algorithms are customized to both ℓ-diversity and β-likeness.
- **Data set:** We use real and synthetic datasets of up to 500k tuples and 8 dimensions. All data are included in the repository.
- **Run-time environment:** Our artifact is not OS-specific. However, all experiments have been performed on an Ubuntu 16.04 LTS server with jre 1.8.0_11. Perl scripts are provided for batch experiment submission (perl v5.20.2). No root access is required.
- **Hardware:** No special hardware is required. However the anonymization of the largest dataset (500k tuples) requires at least 64GB of RAM.
- **Execution:** The anonymization of the largest dataset (500k) may take 3-4 days to complete. Most of the experiments are performed with the default dataset of 10k tuples and each of them lasts up to a couple of minutes.
- **Security, privacy, and ethical concerns:** Since only synthetic as well as open and publicly available datasets have been used, there are no security, privacy, or ethical implications in running the experiments.
- **Metrics:** The evaluation metrics include execution time, information loss incurred by the anonymization, accuracy of queries and attacks on the anonymized dataset.
- **Output:** The output is printed in text (.txt) files, following a specific format.
- **Experiments:** All experiments can be replicated and results reproduced simply by cloning the repo, compiling the code (C++ for NH and PrivBayes, and Java for all the rest) and running the perl scripts provided - one for each of the experiments - following the instructions in the README file.
- **How much disk space required (approximately)?:** In the order of MB.
- **How much time is needed to prepare workflow (approximately)?:** A few minutes.
- **How much time is needed to complete experiments (approximately)?:** 4-5 days.
- **Publicly available (explicitly provide evolving version reference)?:** Our artifact, excluding the source code of the NH, BuReL and PrivBayes algorithms that we compare against, are publicly available under an open source license.
- **Code licenses (if publicly available)?:** Apache Licence 2.0.
- **Archived (explicitly provide DOI or stable reference)?:** (https://github.com/discont/disclosurecontrol/releases/tag/artifact-evaluation).

A.3 Description

A.3.1 How to access

The artifact is publicly available and hosted by GitHub here:

https://github.com/discont/disclosurecontrol

To download the latest version, clone the repository using the command

```
git clone https://github.com/discont/disclosurecontrol.git
```

A.3.2 Hardware dependencies

No specific hardware features are required to evaluate the artifact. To be able to execute our algorithms using the largest input dataset at least 64GB of RAM are required. For the default dataset of 10k tuples, used in the majority of experiments, 8GB of RAM will suffice. As for disk space, our artifact has minimal requirements of a few MB.

A.3.3 Software dependencies

All experiments ran on an Ubuntu 16.04 LTS server. Java code has been compiled with jdk1.8.9_291. Perl scripts were used for batch experiment submission (perl v5.20.2).

A.3.4 Data sets

We use real data drawn form the CENSUS and the COIL 2000 datasets, which are publicly available. Additionally, we generate synthetic datasets of up to 500K tuples and 8 attributes based on the CENSUS data, varying the bias of the sensitive value distribution. All datasets are included in the repository.

A.4 Installation

First, clone the relevant GitHub repository. Compile the source code using plain javac or your favorite Java IDE. Place the .class files in a folder within the local directory where the GitHub repo has been cloned, named bin. To reproduce the experiments, use the provided perl scripts, along with the input datasets. The README file offers a step-by-step guide.

A.5 Experiment workflow

All experiments can be executed by invoking the relevant perl scripts as described in the accompanying README file.

A.6 Evaluation and expected results

Our experiments start by evaluating the application of our algorithms on the achievement of ℓ-diversity. Our findings show that our methods outperform the state-of-the-art NH in utility under various values of the privacy parameter ℓ, the number of *QI* dimensions d, the dataset size n and the skewness of the data distribution (Figures 2, 3, 4a, 4c, 5a and 5c). Our schemes can be applied on partitions of the input dataset in a data-parallel environment, slightly sacrificing utility for the sake of scalability (Figures 4a, 4b, 5a and 5b).

Then, we adapt our algorithms to achieve β-likeness. The experiments demonstrate that our methods offer better utility than the state-of-the-art β-likeness algorithm, BuReL, regardless of the β value and the distribution of tuple values (Figure 6). The utility gain of our schemes compared to BuReL grows with data size (Figure 7a) but shrinks with the skewness of dataset values (Figure 7c). In all cases, at least one of our methods outperforms BuReL in terms of utility. Moreover, our algorithms provide anonymized datasets that can serve range and prefix queries of various selectivities with significantly better accuracy — in terms of median query relative error — compared to BuReL and PrivBayes (Figures 8 and 9). Last, our schemes provide stronger resistance than state-of-the-art differential privacy schemes (PrivBayes) to learning-based attacks under our adversary model on real-world data (Figure 11).

Each of the aforementioned results can be reproduced by simply running automated perl scripts accompanying the code. The scripts execute the provided code with the necessary parameters and record the metrics of interest. The README file walks the user through this process.

Due to the intentional introduction of randomness in the tuple assignment extraction stage, information loss results may slightly differ in each run. The same applies to the accuracy of range and prefix queries, which are randomly generated at run time. Query accuracy may exhibit larger deviations, thus we suggest to execute relevant experiments multiple times and adopt the median value.

A.7 Version

Based on the LaTeX template for Artifact Evaluation V20220119.