Qrlew: Differentially Privacte SQL Query Rewriting

Anonymous submission

Abstract

AAAI creates proceedings, working notes, and technical reports directly from electronic source furnished by the authors. To ensure that all papers in the publication have a uniform appearance, authors must adhere to the following instructions.

Introduction

In recent years, the importance of safeguarding privacy when dealing with personal data has continuously increased. Traditional anonymization techniques have proven vulnerable to re-identification, as demonstrated by numerous works (Archie et al. 2018; Dwork et al. 2017; Narayanan and Shmatikov 2008; Sweeney, Abu, and Winn 2013). The total cost of data breaches has also significantly increased (IBM 2023) and governments have introduced stricter data protection laws. Yet, the collection, sharing, and utilization of data hold the potential to generate significant value across various industries, including healthcare, finance, transportation, and energy distribution.

To realize these benefits while managing privacy risks, researchers have turned to *differential privacy (DP)* (Wood et al. 2018; Dwork, Roth et al. 2014), which has become the gold standard in academia since its introduction by Dwork et al. in 2006 (Dwork et al. 2006) due to its provable and automatic privacy guarantees.

Despite the availability of open-source tools, DP adoption remains limited. One of the reasons for this lack of adoption is the relative complexity of the existing tools considered the utility of the results. *Qrlew* has been designed to solve these problems by providing the following features:

***Qrlew* provides automatic output privacy guarantees** With *Qrlew* a *data owner* can let an analyst (*data practitionner*) with no expertise in privacy protection run arbitrary SQL queries with strong privacy garantees on the output.

***Qrlew* leverages existing infrastructures** *Qrlew* rewrites a SQL query into a *differentially private* SQL query that can be run on any data-store with a SQL interface from lightweight DB to big-data stores. This removes the need for a custom execution engine and enables *differentially private analytics with virtually no technical integration*.

***Qrlew* leverages synthetic data** . Synthetic data are an increasingly popular way of *privatizing* a dataset. Using jointly *differentially private* mechanisms and *differentially private* synthetic data can be a simple, yet powerful, way of managing a privacy budget and reaching better utility-privacy tradeoffs.

Assumptions and Design Goals

In this work, we assume the *central model of differential privacy* (Near 2020), where a trusted central organization: Hospital, Insurance Company, Utility Provider, called the *data owner*, collects and stores personal data in a secured database. and whishes to let untrusted *data-practitionners* run SQL queries on its data. Furthermore, the *Qrlew* was designed to ease the

General architecture

In this work, we assume the *central model of differential privacy* (Near 2020), where a trusted central organization: Hospital, Insurance Company, Utility Provider, called the *data owner*, collects and stores personal data in a secured database. and whishes to let untrusted *data-practitionners* run SQL queries on its data. Furthermore, the *Qrlew* was designed to ease the

Paul on compilation

Privacy algorithms

At this stage, we examine a `Relation` that takes one or two protected `Relation` inputs, where the entity to be protected is specified. To secure the outcome of the SQL query, it is imperative to protect not only the results obtained from aggregation functions but also the grouping keys. We will briefly describe these two steps in the following section.

Protecting aggregation results The protection of aggregation functions is carried out in three sequential steps. Given that all currently supported aggregations (`COUNT`, `SUM`, `AVG`, `VARIANCE` `STDDEV`) can be expressed as compositions of sums, our focus will be on the `SUM` aggregation. Let's consider the scenario where we aim to compute the sum of a column.

1. **Limit the contribution per user within groups**: We represent the contribution of each user by a vector whose each component is the sum of the contributions within one group then we calculate its ℓ_2 norm. Subsequently,

we constrain the contributions of a specific user by scaling $\mathbf{x}$ with a factor chosen to maintain the original observations if the ℓ_2 norm of the user's observations does not exceed the clipping factor c. Conversely, if the ℓ_2 norm surpasses c, the ℓ_2 norm of the rescaled observations is restricted to c. More technical details can be found in the appendix .

2. **Add random noise**: The sum of the original data is substituted with the sum of the scaled data with the addition of Gaussian noise. The level of noise is parameterized by the clipping and privacy parameters.
3. **Restrict differentially private aggregation**: The final operation involves confining the differentially private aggregation within the bounds automatically computed for the $\texttt{SUM}(\mathbf{x})$.

Protecting grouping keys As explained by Wilson et al., making grouping keys public could result in privacy breaches. The treatment of grouping keys varies depending on whether the analyst knows them.

- **Public Grouping Keys.**We categorize grouping keys as public when the analyst specifies them in the `WHERE` clause. In this scenario, all user-specified keys must be disclosed, even if they do not exist in the database. If a key is absent in the database, differentially private results will include these missing keys, and the corresponding aggregations will be random noise.
- **Revealing Private Keys through τ-Thresholding** As introduced by Wilson et al., tau-thresholding involves releasing keys whose differentially private noise surpasses a threshold determined by privacy parameters.

When the SQL query involves both public and private grouping keys, we perform cross joins on the grouping keys obtained through the two algorithms.

Comparison to other systems

The Google differential-privacy extension for ZetaSQL enables differentially private analysis through SQL tools. Users can incorporate privacy guarantees by including a privacy clause in their queries, which replaces aggregations with their differentially private equivalent, limits the number of grouping keys per user, and applies tau-thresholding. While this framework does not support complex queries with joins or inner subqueries, it offers a straightforward integration with SQL databases.

Smartnoise SQL is a Python framework, built on the top of Open-DP, for doing SQL analysis with differential privacy guarantees. The framework is easy of use for non-experts, but it does not support all SQL grammar, and complex queries with joins or subqueries are not supported. Additionally, the execution of SQL analysis requires a Python post-processing step, potentially leading to increased execution times.

OpenDP is a Rust library with Python bidings designed for building SQL-like transformations with differential privacy guarantees. Each part of the code is well documented with mathematical proofs but its utilization can be challenging for for non experts.

Similarly, Tumult Analytics does not directly handle SQL queries; instead, it utilizes transformations that closely resemble SQL syntax. While offering support for budgeting various privacy tasks, it is important to note that the analysis must be executed by the data owner.

Chorus is the closest framework to *Qrlew*, and in the work by Johnson et al., there is a strong emphasis on conducting all analyses directly within the database. However, it's worth noting that the code for Chorus is no longer being actively maintained.

Known limitations

Qrlew relies on the random number generator of the SQL engine used. It is usually not a cryptographic noise.

Qrlew uses the floating-point numbers of the host SQL engine, therefore our system is liable to the vulnerabilities described in Casacuberta et al..

Useful links

PPAI

Last year papers: https://aaai-ppai23.github.io/#sp2 This year program: https://ppai-workshop.github.io/

Comparable open-source projects

- Paszke et al. 2017 - Automatic differentiation in PyTorch https://openreview.net/pdf?id=BJJsrmfCZ
- Frostig et al. 2018 - Compiling machine learning programs via high-level tracing https://mlsys.org/Conferences/2019/doc/2018/146.pdf

Comparable DP SQL papers

- Lessons Learned: Surveying the Practicality of Differential Privacy in the Industry (Garrido et al. 2022)
- Tumult Analytics: a robust, easy-to-use, scalable, and expressive framework for differential privacy (Berghel et al. 2022)
- Differentially Private SQL with Bounded User Contribution (Wilson et al. 2019)
- CHORUS: a Programming Framework for Building Scalable Differential Privacy Mechanisms (Johnson et al. 2020)
- Towards Practical Differential Privacy for SQL Queries (Johnson, Near, and Song 2018)

Thank you for reading these instructions carefully. We look forward to receiving your electronic files!

References

Archie, M.; Gershon, S.; Katcoff, A.; and Zeng, A. 2018. Who's watching? de-anonymization of netflix reviews using amazon reviews.

Berghel, S.; Bohannon, P.; Desfontaines, D.; Estes, C.; Haney, S.; Hartman, L.; Hay, M.; Machanavajjhala, A.; Magerlein, T.; Miklau, G.; et al. 2022. Tumult Analytics: a robust, easy-to-use, scalable, and expressive framework for differential privacy. *arXiv preprint arXiv:2212.04133*.

Casacuberta, S.; Shoemate, M.; Vadhan, S.; and Wagaman, C. 2022. Widespread Underestimation of Sensitivity in Differentially Private Libraries and How to Fix It. arXiv:2207.10635.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, 265–284. Springer.

Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4): 211–407.

Dwork, C.; Smith, A.; Steinke, T.; and Ullman, J. 2017. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4: 61–84.

Garrido, G. M.; Liu, X.; Matthes, F.; and Song, D. 2022. Lessons learned: Surveying the practicality of differential privacy in the industry. *arXiv preprint arXiv:2211.03898*.

IBM. 2023. Cost of a Data Breach Report 2023.

Johnson, N.; Near, J. P.; Hellerstein, J. M.; and Song, D. 2020. Chorus: a programming framework for building scalable differential privacy mechanisms. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, 535–551. IEEE.

Johnson, N.; Near, J. P.; and Song, D. 2018. Towards Practical Differential Privacy for SQL Queries. *Proc. VLDB Endow.*, 11(5): 526–539.

Narayanan, A.; and Shmatikov, V. 2008. Robust deanonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 111–125. IEEE.

Near, J. 2020. Threat Models for Differential Privacy.

Sweeney, L.; Abu, A.; and Winn, J. 2013. Identifying participants in the personal genome project by name (a re-identification experiment). *arXiv preprint arXiv:1304.7605*.

Wilson, R. J.; Zhang, C. Y.; Lam, W.; Desfontaines, D.; Simmons-Marengo, D.; and Gipson, B. 2019. Differentially private SQL with bounded user contribution. *arXiv preprint arXiv:1909.01917*.

Wood, A.; Altman, M.; Bembenek, A.; Bun, M.; Gaboardi, M.; Honaker, J.; Nissim, K.; O'Brien, D. R.; Steinke, T.; and Vadhan, S. 2018. Differential privacy: A primer for a non-technical audience. *Vand. J. Ent. & Tech. L.*, 21: 209.

Appendix: Limit the contribution per user within the aggregations

The observations can be represented as a matrix whose first dimension is the user and second dimension is the group:

$$\begin{array}{c} \overbrace{\phantom{x_{11}}}^{\text{Group 1}}\ \overbrace{\phantom{x_{12}}}^{\text{Group 2}}\ \ \ \overbrace{\phantom{x_{1n}}}^{\text{Group n}} \\ \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix} \begin{array}{l} \} \text{User 1} \\ \} \text{User 2} \\ \\ \} \text{User } m \end{array} \end{array}$$

The coordinate x_{ij} is the sum of all the observations of user i within the group j.

For each user i, we compute the scale factor that constrains his contribution to a maximum value of c:

$$s_i = \frac{1}{\max(1, \frac{1}{c}\ell_2(x_{i,.}))} \text{ with } \ell_2(x_{i,.}) = \sqrt{\sum_j x_{ij}^2}. \quad (1)$$

The original data is then rescaled by multiplying each observation by the corresponding user's scale factor. This process ensures that the ℓ_2 contribution of each user is restricted to c within the rescaled data.

In our algorithm, the clipping value c is given by:

$$c = \max(|\min \mathbf{x}|, |\max \mathbf{x}|), \quad (2)$$

where $\min \mathbf{x}$ and $\max \mathbf{x}$ are the automatically computed bounds of $\mathbf{x}$.