# Differentially Private Programming

Marco Vassena

Chalmers University
vassena@chalmers.se

**Abstract.** Data analysts mine large databases and crunch data in order to extrapolate statistics and interesting patterns. However people's privacy is jeopardized in the process, whenever a database contains private data. *Differential privacy* has emerged recently as an appealing rigourous definition of privacy, which protects individuals in a database, while allowing data analysts to learn facts about the underling population, by adding noise to queries. Unfortunately, proving differential privacy of programs is a difficult and error-prone task. In this paper, we survey the state-of-the-art applications of programming languages techniques to develop principled approaches and tool support to ease the analysis and verification of probabilistic differential private programs.

## 1 Introduction

Our society has gone through a digitalization process in the past few decades, which has produced large amounts of data, that have been collected and stored in databases over the years. Data analysts dig into this vast source of information in order to discover useful information, extrapolate patterns and statistics and support decision-making processes. Unfortunately, data analysis puts people's privacy at stake, since archives often contain private personal data. How to process and analyze large amounts of personal data, while respecting people's privacy is an intriguing open research problem. *Differential privacy* has emerged recently as an appealing rigourous definition of privacy, which protects individuals in a database, and allows to learn facts about the underling population, by adding noise to queries. In this paper, we survey the state-of-the-art applications of programming languages techniques to differential privacy, whose goal is to develop principled approaches and tool support for probabilistic differential private programming, in order to make privacy-preserving data-analysis feasible.

## 2 Background

Privacy is hard to guarantee, when lots of rich data is independently available. A number of naive approaches to the problem revealed to be shockingly broken. For instance, *data anonymization* is vulnerable to *linkage attacks*, i.e., matching anonymous records with auxiliary non-anonymous data from a public data-set allows identification [7, 5]. Collective queries, i.e., queries over large sets, are not

safe either as they enable *differencing attack*, while query *auditing*, i.e., refusing to reply to a query that would compromise privacy in combination with data queried previously, can also be *disclosive*. Furthermore deciding whether a query constitutes a privacy breach may also be *undecidable* for a rich query language. These subtleties indicate the need for a more formal treatment of privacy.

### 2.1 Differential Privacy

Informally, differential privacy guarantees that the *participation* of an individual in a statistical database, e.g., a survey, shall not significantly affect him or her in any way, therefore encouraging participation and data collection.

More formally, $(\epsilon, \delta)$-differential privacy is *property* of a randomized algorithm that has access to a database. Parameter $\epsilon$ indicates the *privacy loss*, while parameter $\delta$ determines the *accuracy* of the algorithm. Intuitively, differential privacy ensures that the algorithm will behave *similarly* on any pair of *adjacent* databases, i.e., databases that differ only in a row, i.e., the "individual" whose privacy we want to protect. One way to make an algorithm *differentially private* is to inject noise from some probability distribution in the result of queries. The shape and the amount of noise needed depends on the privacy loss[1], i.e., parameter $\epsilon$, at the expense of the precision of the algorithm, i.e., parameter $\delta$. For instance, the Laplacian mechanism injects noise from the Laplacian distribution on the result of *numeric* queries, while the exponential mechanism, perturbs *discrete* queries. Crucially, depending on what the algorithm wants to achieve, noise injection can be *coordinated*, resulting in better performances, in terms of $\epsilon$ and $\delta$—something that it is in general hard, tedious and error-prone. In [8, 6, 3, 2, 1], programming languages techniques have been employed to simplify the process of formally verifying and composing differentially private programs.

## 3 Distance Makes the Types Grow Stronger

In their paper, Reed and Pierce present a functional calculus[2], which ensures that well-typed programs are differentially private. The type-system of the calculus tracks *function sensitivity*, i.e., a measure, which gives an upper bound on how much a function can magnify the distance between similar inputs, which determines *proportionally* the noise required to make a query differentially private.

The notion of *distance* between values is *type-based*. The type-system includes linear types [8], a special kind of types that model terms as consumable resources—a fundamental aspect to correctly capture sensitivity. For example the special arrow type, i.e., $\multimap$, guarantees that the argument of the function is used at most once, which is sufficient to ensure that the function has sensitivity 1. In particular, in order to safely *share* the output of $c$-sensitive functions, the

[1] Alternatively, it is also possible to compute the privacy-loss from the algorithm. In this case $\epsilon$ is referred as *budget*.

[2] Implemented as Fuzz `http://privacy.cis.upenn.edu/software.html`

calculus provides the type of *additive conjunction*, i.e., $\langle t_1, t_2 \rangle : \tau_1 \,\&\, \tau_2$, which provides any of $t_1$ or $t_2$, but not both. The special type $!_r\ \tau$ allows to reuse a term of type $\tau$ up to $r$ times, for any number $r$, including $\infty$, at the price of multiplying the scaling factor $r$ to the distance metric.

Recursive types provides the language supports with standard functional programming data-types, such as natural numbers and lists and recursive patterns, i.e., a fixpoint combinator $Y$, *map* and *fold*. The calculus provides as primitives sets and finite maps, which are useful to model databases and relational algebra operations respectively, and equip them with appropriate distance metrics. By extending the calculus with the *probability monad*, i.e., probabilistic computations, and defining a suitable distance measure, the authors capture the type of differential private computations *within* the language. Guarantees about differential privacy of programs with this type follow directly from the soundness of the type-system. The paper concludes with a few examples of classic differentially private algorithms that can be expressed in their language, including histogram queries and $k$-means.

## 4 State of the Art

The type-system of Fuzz is limited to types with constant sensitivity annotations, which must be statically known [8]. For example, Fuzz supports a $k$-means clustering algorithm for a *fixed* number of iterations, i.e., $k = 1, 2, 3...$, but does not allow a unique algorithm that works for any $k$, e.g., an adaptive version of the algorithm that, depending on the data, stops when a suitable $k$ is found.

*DFuzz* lifts these restrictions by providing *dependent types*, a form of types which may depend on run-time *values* [6]. Even though more expressive, DFuzz is a significantly more complex language, which includes simple types, linear types, a kind system, a probability type layer and a sub-typing relation, which requires an external constraint solver to *type-check*.

HOARe$^2$ is a relational refinement type system for high-order probabilistic programs [2]. Refinement types are an expressive type discipline that captures fine-grained properties of computations by enriching types with assertions, about their values.

Proving differential privacy for certain algorithms requires various mathematical techniques, that are beyond the reach of type-systems and constraint solvers. CertiPriv is a machine-checked framework for reasoning about differential privacy built on top of the Coq proof assistant from first principles—for example, it can verify the correctness of the Laplacian and Exponential mechanisms too.

## 5 Differentially Private Bayesian Programming

This work combines advances in probabilistic programming, relational refinement type systems and differential privacy and presents PrivInfer, a framework that extends HOARe$^2$ with symbolic probability distribution, to support provably differentially private Bayesian machine learning algorithms [1].

*Probabilistic programming* is concerned with the design of programming languages as tools for machine learning. For example, probabilistic programming allows to describe a probabilistic model, its parameters and the data observations as a program, which is then used to perform inference. *Bayesian inference* is a statistical method that updates the probability for a hypothesis as more evidence, i.e., observations, becomes available, with many applications, including machine learning.

Unfortunately, when the observation include private data, releasing the posterior distribution directly represents a privacy violation [4, 9, 10]. In fact, despite Bayesian inference outputs a probability distribution, the inference process is completely *deterministic*, therefore it reveals private information. Differential privacy requires to inject randomness in the process, but this can be done in different ways, on the input data, on the parameters, or on the distribution itself. Then, in order to reason about differential privacy in this setting, the paper explores $f$-divergence, a class of distance measures on probability distributions, which are given a relational interpretation based on relational lifting. PrivInfer distinguishes between different kinds of distributions syntactically: the inference process yields *symbolic* distributions, while *actual* distributions represent differentially private probabilistic computations. The paper concludes with three different differential privacy conditions which can be formally verified by PrivInfer, depending on how noise is injected.

## 6 Conclusion

Differential privacy is an appealing desirable property that formally establishes what privacy means in the context of data-analysis and encourages users to participate in statistical databases. Unfortunately, proving that algorithms are differential private is a difficult and error-prone task that calls for principled approaches and tool support. This paper surveys the state-of-the-art programming languages techniques that support systematic analysis and verification of privacy-preserving data-analysis programs, including machine learning algorithms. While this research area is still in its early stages, these papers show initial promising results.

# Bibliography


[1] Gilles Barthe, Gian Pietro Farina, Marco Gaboardi, Emilio Jesus Gallego Arias, Andy Gordon, Justin Hsu, and Pierre-Yves Strub. Differentially private bayesian programming. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 68–79, New York, NY, USA, 2016. ACM.

[2] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Pierre-Yves Strub. Higher-order approximate relational refinement types for mechanism design and differential privacy. *SIGPLAN Not.*, 50(1):55–68, January 2015.

[3] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella-Béguelin. Probabilistic relational reasoning for differential privacy. *ACM Trans. Program. Lang. Syst.*, 35(3):9:1–9:49, November 2013.

[4] Christos Dimitrakakis, Blaine Nelson, Aikaterini Mitrokotsa, and Benjamin I. P. Rubinstein. *Robust and Private Bayesian Inference*, pages 291–305. Springer International Publishing, Cham, 2014.

[5] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 202–210, New York, NY, USA, 2003. ACM.

[6] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. Linear dependent types for differential privacy. *SIGPLAN Not.*, 48(1):357–370, January 2013.

[7] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[8] Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: A calculus for differential privacy. In *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*, ICFP '10, pages 157–168, New York, NY, USA, 2010. ACM.

[9] Oliver Williams and Frank Mcsherry. Probabilistic inference and differential privacy. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2451–2459. Curran Associates, Inc., 2010.

[10] Zuhe Zhang, Benjamin I. P. Rubinstein, and Christos Dimitrakakis. On the differential privacy of bayesian inference. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2365–2371. AAAI Press, 2016.