# Research Statement

Ios Kotsogiannis

November 29, 2017

## 1 Introduction

In recent years, *differential privacy* (DP)[2] has emerged as the state-of-the-art privacy definition for analyzing sensitive data. Typically DP is achieved by specialized algorithms that add calibrated noise to the data according to a privacy parameter $\epsilon$. Algorithm design in DP has been highly impactful, as task-specific algorithms have been developed that dramatically reduce error rates when compared with general purpose DP mechanisms[5], while still enjoying the same level of privacy. Despite this success in the academic community the gap between theory and practice has not fully closed. The problematic deployment of DP algorithms, their highly non-trivial error analysis, and the limited support for general SQL queries are among the factors that hinder the utilization of DP in real systems. On top of that, the actual threat model of many applications is weaker than the one DP protects against, which leads to excessive noise addition when using DP for the privacy analysis. The focus of my work is to provide with the necessary tools that will help bridge the gap between current innovations in privacy research and real world systems where privacy is desired but not yet applied.

More specifically, using any of the sophisticated DP algorithms can be extremely difficult — multiple subroutine implementations, hidden hard-coded parameters, and unvetted code are some of the problems a practitioner needs to overcome before using existing algorithms. This leads to non-experts to resort to baselines like the Laplace mechanism[3], that offer far from optimal error rates[5], for their private data analysis. Another challenge that privacy practitioners face is the burden of choosing the DP algorithm that incurs the least error for their specific analysis. For a large class of DP algorithms there are no known analytical error bounds. Thus, selecting an algorithm that provides competitive error rates for a specific input task becomes a riddle, where wrong choices lead to unacceptable error rates. Next, even if the problems of implementation and algorithm selection for DP are resolved, an important limitation of the existing literature is that the current solutions for an end-to-end DP system that answers general database queries (such as SQL) on sensitive datasets are not satisfactory. Lastly, for many real-world applications DP is overly pessimistic in its assumptions and a more lax privacy definition is preferable. However, the lack of sophisticated algorithms for these definitions, prompts the privacy practitioner to choose a DP mechanism even if DP is an "overkill" for his privacy needs — i.e., in theory better error bounds are possible for the preferred privacy level.

As a researcher my goal is to provide with the necessary tools such that enable us to overcome these limitations, bridge the gap between theory and practice, and bring provable privacy guarantees to real applications with access to sensitive data.

## 2 Summary of Past Work

The projects I previously worked on helped address the limitations described earlier. More specifically, in [10] we propose $\epsilon$ktelo, a simple to use programming interface that safely allows users to author a large class of DP algorithms proposed in prior work. In [8] we introduce the problem of algorithm selection under DP and provide a solution with near-optimal error rates — and in [7] we provide with a live demonstration where users can interact with Pythia. Lastly, in [1], we propose *one-sided differential privacy*, a generalization of DP suitable for situations where only part of the data is sensitive.

### 2.1 Ektelo

Ektelo[10] is a programming interface that allows users to express their algorithms as *plans of sequential operators* that are vetted by the system to satisfy DP. The interface allows non-experts of privacy to safely and quickly write programs that implement DP algorithms. High expertise users are also benefited from $\epsilon$ktelo as they can easily author novel algorithms by mixing components of known DP algorithms. Lastly, $\epsilon$ktelo exposes the high level ideas behind algorithms so that researchers can focus their efforts on the very parts of the algorithms that will offer the highest advantages.

The operator classes of $\epsilon$ktelo help abstract DP algorithms as a sequence of distinct "phases" (e.g., domain reduction, noise addition, inference, etc.) where each phase is implemented from a different operator of a single class. This powerful abstraction allows: (a) algorithm comparison in the operator level, (b) the improvement of already competitive algorithms by carefully changing a set of operators, and (c) the construction of novel algorithms (i.e., plans) by using operators in different ways. More specifically, in a use case about releasing Census data we showed an error improvement of up to 47$\times$ to a popular algorithm [11] just by changing a single operator. In the same use case, we constructed novel algorithms via the expressiveness of $\epsilon$ktelo that proved to be the most competitive for that task. In a second use case about learning a DP naive Bayes classifier we were able to define complex adaptive plans that would execute a different sub-plan depending on characteristics of the data.

### 2.2 Pythia

A recent study of DP algorithms [5] showed that the algorithm performance in terms of error rates is dependent on the sensitive input. More specifically, the authors identified a large class of algorithms as "data-dependent" — i.e., their error rates depend on the input dataset — and it is unknown whether their error rates can be analytically computed. Moreover, they showed that across different input datasets, workloads, and values of the privacy parameter $\epsilon$ there is no clear "winner" algorithm that dominates. This leaves both the privacy expert and the non-expert with the problem of selecting a suitable algorithm for their task that achieves low error. In Pythia[8] we formalize the problem of algorithm selection under DP and propose Pythia, an end-to-end DP solution for the problem of algorithm selection that offers competitive error rates and can trivially include any algorithm from prior (and future) work.

An overview of Pythia can be seen in figure 1. More specifically, Pythia learns a feature-based algorithm selection model (FAS) by executing DP algorithms on public non-sensitive datasets and recording their errors. We implement FAS as a decision classification tree, whose leaves are labeled with an algorithm and internal nodes are labeled with a feature and a value for that feature. Figure 2 shows an example of

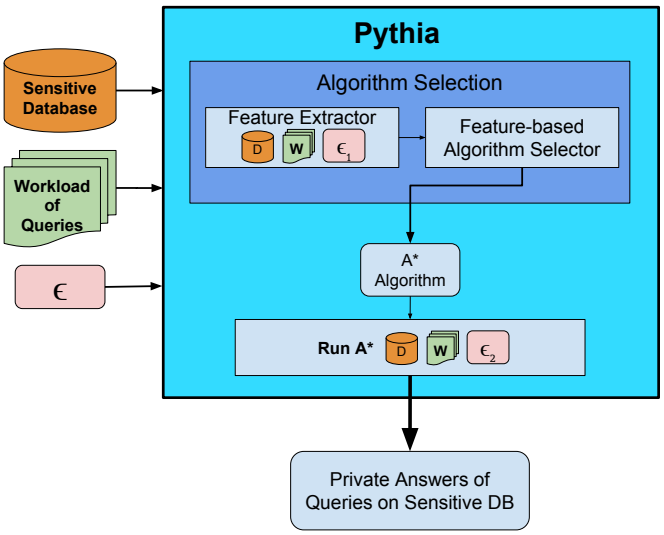


Figure 1: The Pythia meta-algorithm computes private query answers given the input data, workload, and epsilon. Internally, it models the performance of a set of algorithms, automatically selects one of them, and executes it.

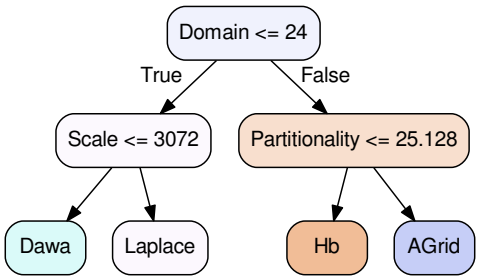


Figure 2: Example of an FAS for 2D range queries.

a possible tree for a specific task. At run-time, for a new sensitive input, Pythia first privately estimates a set of features of the input, then traverses the FAS and selects an algorithm. Lastly, it executes the selected algorithm on the sensitive input with the appropriate privacy parameter. The end-to-end privacy guarantee of Pythia follows from the sequential composition theorem of DP[2].

One important design decision of Pythia, is that the FAS is learned in a data-independent fashion, which saves privacy budget for the later tasks of Pythia and more importantly allows the re-usability of the FAS for multiple instances of similar tasks (e.g., 2D range queries). Another key aspect of our design is that the FAS is learned via a novel classification criterion that allows classifying training instances by what *groups of algorithms* perform similarly good — instead of what is the best algorithm. This technique allows Pythia to achieve more than 30% better error rate compared to using the standard Gini impurity classification criterion.

### 2.3 One-sided Differential Privacy

In [1] we propose One-sided Differential Privacy (OSDP), a new privacy definition applicable in settings where only part of the data is sensitive. Such scenarios naturally occur due to user preferences, privacy policies, or as a result of legislature. The baseline approach that treats all data as equally sensitive fails to take advantage of the non-sensitive data to improve the quality of the answers. On the other hand, other techniques that release the sensitive data under some provable privacy guarantee (like DP) while truthfully releasing the non-sensitive data are susceptible to *exclusion attacks*, i.e., an attack based on the absence of a target record from the non-sensitive truthful release.

In this work, we formally define exclusion attacks, show how previous privacy definitions fail to protect against them and propose OSDP, a privacy definition that provides DP-like guarantees for the sensitive data and enjoys freedom from exclusion attacks. We also propose primitive mechanisms that satisfy OSDP as well as a general recipe that transforms specialized DP algorithms to satisfy OSDP. By doing so, the hybrid algorithms leverage on the presence of non-sensitive records and improve the error rates by a factor of $10\times$.

## 3 Ongoing and Future Work

In future work I concentrate my efforts to bring privacy to real database systems. Most DP algorithms in literature are well engineered algorithms that privately estimate a set of counting queries on a sensitive dataset. While this is a large class of queries that supports multiple applications (e.g., release of $k$-way marginals, build statistical models, learn classifiers, etc.) it is nonetheless limiting as it doesn't fully capture the expressiveness of general database queries (e.g., SQL). I propose a promising new line of work that will provide the privacy practitioner with the necessary tools to apply any of these sophisticated algorithms to answer general database queries.

**Goal:** Our primary goal is to provide an end-to-end DP system that supports answering general SQL queries with arbitrary joins and multi-level group by clauses over a database with a single private table. We define the following desiderata that such a system should satisfy: (1) use any of the excellent algorithms of DP literature, (2) the exact and efficient computation of the privacy loss, and (3) the use of sophisticated inference techniques that refine the private answers. These desiderata are crucial to the success of the system as not satisfying any of them highly impacts the error rates on the queries.

**Prior Work:** Prior work like PINQ [9] and Elastic Sensitivity [6] provide end-to-end systems that answer SQL queries under $\epsilon$-DP. However, both works fail in the 3 desiderata we stated earlier, as they (1) use the Laplace mechanism for query answering, (2) answer queries sequentially under the composition theorem of DP, and (3) do not perform any inference to improve the quality of the noisy estimates. Other work like PrivBayes [11] can be used to produce an $\epsilon$-DP synthetic version of the data, this is known as the non-interactive privacy model which allows for multiple query releases with no additional privacy leaks. Two caveats we discovered with such approaches are that they do not scale for large relational tables with multiple attributes and they do not consider the workload in the private analysis, which leads to high error rates for certain workloads.

**Overview:** For our solution we focus on the *vectorization* of both the queries and (a view of) the data. By representing queries and the data as vectors we are able to satisfies all three desiderata and construct a system that will allow us to leverage on the rich literature of DP in order to privately answer general database queries.
**Notation:**

Let $D$ be a SQL database with multiple relations $T_i$.

Relational view: $V(A_1, \dots A_k)$ with domain $dom(V) = dom(A_1) \times \dots \times dom(A_k)$.

Instance $I$ of $V$: a multiset of $N$ records $t \in dom(V)$.

Linear query (LQ): $q(I) = \sum_{t \in I} f(t)$ where $f : dom(V) \to \{0, 1\}$.

**Problem Statements:** Towards creating such a system we first focus on two main problems whose solutions will be integral parts of the system: (a) given $I$ an instance of a view $V$ and $\mathcal{W}$ a set of linear queries on $V$, we want to produce a vector representation of $I$ and $\mathcal{W}$. The second and more general problem is that given a database $D$ and a set of general queries $Q$ on $D$, we want to find a set of views $\mathcal{V}$ of $D$ such that each query can be represented as linear query in at least one view. More formally:
**Vectorization:** Let $I$ be an instance of $V$ and $\mathcal{W} = \{q_1, \dots q_m\}$ a workload of $m$ LQs under $V$. Find vectors $\mathbf{x}$ and $\{\mathbf{q}_i\}_{\forall i \in [m]}$ s.t.:

$$\forall i : \; q_i(I) = \mathbf{q}_i^\intercal \mathbf{x}$$

**View Selection:** Let $D$ be a database with multiple relations and $Q$ be a workload of $m$ general SQL queries on $D$. Find a minimal set of views of $D$: $\mathcal{V} = \{V_1, \cdots, V_k\}$ s.t. $\exists P(Q) = Q_1, \dots Q_k$ s.t. $\forall i \in [k], \forall q \in Q_i : q$ can be answered from $q'$ a LQ on $V_i$, s.t. removing any view $V_i$ from $\mathcal{V}$ would leave some queries with no LQ representation.

**Challenges:** When we perform vectorization the main challenges we face are: (a) space complexity of $\mathbf{x}$, (b) high sparsity of $\mathbf{x}$ which leads to low signal to noise ratio, (c) presence of structural zeros in $\mathbf{x}$ due to functional dependencies, attribute value correlations, etc. Note that all three challenges are highly correlated: a loose vector representation of $I$ where all structural zeros are fully represented will be prohibitively large and highly sparse with the average cell count being much smaller than the noise added. Moreover the addition of noise to structural zeros will unnecessarily increase the error rate as well as create synthetic tuples that will not make sense to a user — e.g., adding noise to a Census database might result to a household owner that is 3 years old.

The challenges we face in view selection are equally important. While the literature on view selection is rich (see [4] for a survey) there is no prior work that focuses on linear queries. First, given a query $q$ and a view $V$ it is not trivial to estimate whether or not a query $\bar{q}$ exists such that it is linear on $V$ and provides the same answer. Another challenge we need to resolve is that while the set of views returned $\mathcal{V}$ should be minimal (i.e., $k$ should be small) they should also be general enough to allow for new queries to be expressed as linear queries on them.

# References


[1] S. Doudalis, I. Kotsogiannis, A. Machanavajjhala, and M. Sharad. One-sided privacy. In *TPDP*, 2017.

[2] C. Dwork. Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2006.

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[4] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, Dec. 2001.

[5] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang. Principled evaluation of differentially private algorithms using dpbench. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, 2016.

[6] N. M. Johnson, J. P. Near, and D. X. Song. Practical differential privacy for SQL queries using elastic sensitivity. *CoRR*, abs/1706.09479, 2017.

[7] I. Kotsogiannis, M. Hay, A. Machanavajjhala, G. Miklau, and M. Orr. Dias: Differentially private interactive algorithm selection using pythia. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1679–1682. ACM, 2017.

[8] I. Kotsogiannis, A. Machanavajjhala, M. Hay, and G. Miklau. Pythia: Data dependent differentially private algorithm selection. In *SIGMOD*, 2017.

[9] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM.

[10] D. Zhang, R. McKenna, I. Kotsogiannis, G. Miklau, M. Hay, and A. Machanavajjhala. Ektelo: A framework for defining differentially-private computations. In *TPDP*, 2017.

[11] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1423–1434, New York, NY, USA, 2014. ACM.