Personalised Differential Privacy

Summary of POPL'15 paper "Differential Privacy: Now It's Getting Personal"

Hamid Ebadi and David Sands

Department of Computer Science and Engineering,
Chalmers University of Technology, Sweden
{hamid.ebadi,dave}@chalmers.se

Abstract. Differential privacy provides a way to get useful information about sensitive data without revealing much about any one individual. It enjoys many nice compositionality properties not shared by other approaches to privacy, including, in particular, robustness against side-knowledge.
Designing differentially private mechanisms from scratch can be a challenging task. One way to make it easier to construct new differential private mechanisms is to design a system which allows more complex mechanisms (programs) to be built from differentially private building blocks in principled way, so that the resulting programs are guaranteed to be differentially private by construction.
This paper is about a new accounting principle for building differentially private programs. This approach is based on a simple generalisation of classic differential privacy which we call Personalised Differential Privacy (PDP). In PDP each individual has its own personal privacy level. We describe ProPer, a interactive system for implementing PDP which maintains a privacy budget for each individual. When a primitive query is made on data derived from individuals, the provenance of the involved records determines how the privacy budget of an individual is affected: the number of records derived from Alice determines the multiplier for the privacy decrease in Alice's budget. This offers some advantages over previous systems, in particular its fine-grained character allows better utilisation of the privacy budget than mechanisms based purely on the concept of global sensitivity, and it applies naturally to the case of a live database where new individuals are added over time.
We provide a formal model of the ProPer approach, prove that it provides personalised differential privacy, and describe a prototype implementation based on McSherry's PINQ system.

Differential privacy is a relatively new notion of privacy [1–3]. The theory shows that by adding the right amount of noise to statistical queries, one can get useful results at the same time as providing a quantifiable notion of privacy. Its definition does not involve a syntactic condition on the data itself, but rather it is a condition formed by comparing the results of a query on any database with or without any one individual: a query Q (a randomised function) is ε-differentially private if the difference in probability of any query outcome on a data-set only varies by a factor of e^{ε} (approximately $1 + \varepsilon$ for small ε) whenever an individual is added or removed.

Research on differential privacy has developed a variety of query mechanisms that provide differential privacy for a useful range of statistical problems. A few works have focussed more on

composition principles that allow new differential private mechanisms to design a system which allows more complex mechanisms (programs) to be built from differentially private building blocks in principled way, so that the resulting programs are guaranteed to be differentially private by construction [6, 5, 7]. PINQ [6] is the starting point for the present work.

PINQ-style Global Privacy Budget PINQ is an implementation of interactive differential privacy which ensures, at runtime, that queries adhere to a global privacy budget. Third-party client code freely decides how sensitive data sets should be processed and queried. The run-time system ensures that this does not break a specified privacy budget ε. PINQ builds on a collection of standard differentially private primitive queries, together with simple composition principles – mathematical properties enjoyed by the definition of differential privacy. One central principle is that multiple queries (e.g. with differential privacy ε_1 and ε_2 respectively) have an additive effect ($\varepsilon_1+\varepsilon_2$) on the overall differential privacy. Another central idea is to track *sensitivity* of functions to measure how much a change in the input might affect the value of the data. Together, these components allow the system to track how much to deduct from the global privacy budget on each invocation of a primitive query.

Limitations of the Global Privacy Budget In a batch system where all computations are described up-front as a monolithic program, a global budget is reasonable. In an interactive system, however, there are several limitations to this style of accounting. Imagine a scenario involving a large data set of individuals – a cross-set of the population – containing various information about health and lifestyle. Let us suppose, further, that we aim for ε-differential privacy for some specified value of ε. On Monday the analyst selects all the people from the database who have a particular blood type, AB-negative, and constructs an algorithm which extracts information about them as part of medical research. Since just 0.6% of the population have this blood type, the proportion of the database involved in this study is relatively small, but the database is known to be big enough for it to be meaningful. Let us suppose that the cost of this analysis, according to the system, is ε_1. Now on Tuesday the analyst gets a new task, to extract information about the lifestyle of smokers, towards an advertising campaign for nicotine gum. This is a significantly larger portion of the database, possibly overlapping Monday's research group. The analyst has $\varepsilon - \varepsilon_1$ left to spend. If ε_1 is large, the analyst has blown the budget by analysing the small group, even though that study did not touch the data of the larger part of the population. PINQ offers a way around this problem by adding non-standard database primitives. Here we would partition the data into (AB^-, not AB^-) and perform the two studies in parallel, with cost being the maximum of the cost of the two studies.

This leads to a batch-style reasoning and an unnatural programming style. But it also has another limitation. What if the database is live – we obtain new data over time, or if data is continually being added? A global budget forces us to be unnecessarily pessimistic about new as-yet unexploited data.

Personalised Differential Privacy This paper addresses these issues by offering a simple generalisation of differential privacy called *personalised differential privacy (PDP)* [4] which permits each individual to have a personalised privacy budget. The definition supports generalised versions of the composition principles upon which systems like PINQ are based, and moreover enjoys a

number of properties which allow for less wasteful compositional principles. For example, any query about the drinking habits of adults offers 0-differential privacy for Adrian, aged 13, as it does for any records of individuals which enter the database after the query has been made.

Definition 1 (Personalised (Big Epsilon) Differential Privacy). *We say that data sets A and B differ on record r, written $A \overset{r}{\sim} B$, if A can be obtained from B by adding the record r, or vice-versa.*

Let $\mathscr{E}$ be a function from records to non-negative real numbers. A randomized query Q provides $\mathscr{E}$-differential privacy *if for all records r, and all $A \overset{r}{\sim} B$, and any set of outputs $S \subseteq \mathsf{range}(Q)$, we have: $Pr[Q(A) \in S] \leq Pr[Q(B) \in S] \times e^{\mathscr{E}(r)}$*

Personalised differential privacy allows each individual (record) to have its own personal privacy level. This may turn out to be a useful concept in its own right, but its main purpose in this work is as a generalisation that permits a more fine-grained accounting in the construction of classical differentially private mechanisms, and one which plays well with dynamic databases. The following proposition summarises the relation to "small-epsilon" differential privacy:

Proposition 1.

(i) If Q is ε-differentially private, then Q is $\lambda x.\varepsilon$-differentially private.
(ii) If Q is $\mathscr{E}$-differentially private, and $\sup(\mathsf{range}(\mathscr{E})) = \varepsilon$ then Q is ε-differentially private.

Now we consider the composition principles analogous to those above. We keep the presentation informal since we will not apply these principles directly in our formal developments – rather they provide an intuition behind the approach. Most of the principles above generalise to personalised differential privacy.

Query Composition In the sequential composition of queries, if Q_1 and Q_2 are $\mathscr{E}_1$ and $\mathscr{E}_2$-differentially private, respectively, then applied in sequence they yield a $\lambda x.\mathscr{E}_1(x)+\mathscr{E}_2(x)$-differentially private query. For parallel queries let us be a little more precise:

Let $\{R_i\}_{i\in I}$ be a partition of the set of all records, and $\{Q_i\}_{i\in I}$ be a set of queries. we define a parallel query $P(A) = \Pi_{i\in I} Q_i(A \cap R_i)$ where Π is just the n-ary cartesian product of sets. Now we have the following natural generalisation of the parallel query principle:

> If Q_i is $\mathscr{E}_i$-differentially private then P is $\mathscr{E}$-differentially private, where $\mathscr{E}(r) = \mathscr{E}_i(r)$ if $r \in R_i$.

Now we introduce the first specialised principle which takes advantage of the fine-grained nature of personalised differential privacy, the *selection principle*:

> For set A, define $\mathsf{select}_A(x) = x \cap A$. If Q is $\mathscr{E}$-differentially private, then $Q \circ \mathsf{select}_A$ is $\mathscr{E}[r \mapsto 0 \mid r \notin A]$-differentially private.

Here $\mathscr{E}[r \mapsto 0 \mid r \notin A]$ denotes the function which maps every element outside A to 0, and behaves as $\mathscr{E}$ otherwise. In simple terms, a query which operates on A is perfectly private for individuals outside of A. In contrast, the composition principle of ε-differential privacy has nothing helpful to say here: the sensitivity of the selection function is 1.

How does this help us? It can show how the sequential composition principle for $\mathscr{E}$-differential privacy gives greater accounting precision. Specifically, parallel composition is simply no longer necessary to give a reasonably accurate estimate of privacy cost. Suppose we compute $P(A)$ by sequentially computing $Q_i \circ \mathsf{select}_{R_i}$. Then the sequential composition principle calculates the cost of this iterated sequential composition as

$$\lambda x.\Sigma_{i \in I}(\text{if } x \in R_i \text{ then } \mathscr{E}_i(x) \text{ else } 0)$$

which is precisely the cost calculated for the parallel query.

Sensitivity Composition The sensitivity composition principle also lifts into the world of personalised differential privacy:

If data-set transformer F has sensitivity c, and Q is
$\mathscr{E}$-differentially private, then $Q \circ F$ is $\lambda x.(\mathscr{E}(x) \times c)$-differentially private.

The proof analogously follows easily from the following scaling property: If data sets A and B differ on elements C, then

$$Pr[Q(A) \in S] \leq Pr[Q(B) \in S] \times \exp(\Sigma_{r \in C} \mathscr{E}(r)).$$

The Personalised Sensitivity Principle A key feature of personalised differential privacy is that it supports a fine-grained composition property for a large and important class of functions, namely functions that are union preserving. A function F from multisets to multisets is union preserving if $F(A \uplus B) = F(A) \uplus F(B)$, $A \uplus B$ denotes the additive union of multisets A and B. In standard relational algebra, for example, all functions are union preserving in each of their arguments, with the exception of the (multi-)set difference operator which is not union preserving in its second argument. Complex union-preserving functions may be built from simple ones as they are closed under compositions.

The characteristic property of union preserving functions is that their behaviour can be completely characterised by their behaviour on individual records. This gives us a completely precise way to compute the influence of a single record on the result of the function, since $F(A \uplus \{r\}) = F(A) \uplus F\{r\}$. This leads us to the following.

Lemma 1. *If F is a union-preserving function and Q is $\mathscr{E}$-differentially private, then $Q \circ F$ is $(\lambda x.\Sigma_{s \in F\{x\}} \mathscr{E}(s))$-differentially private.*

Taking $\mathscr{E} = \lambda r.\varepsilon$ yields the following useful corollary which is the core of our approach to combining existing differentially private mechanisms with our personalised approach:

Corollary 1. *If F is a union-preserving function and Q is ε-differentially private, then $Q \circ F$ is $(\lambda x.\mathsf{size}(F\{x\}) \times \varepsilon)$-differentially private.*

From these principles we design a system, in the style of PINQ, called ProPer (**Pro**venance for **Per**sonalised Differential Privacy. The ProPer system maintains a personal budget for every record entering the system. Instead of using sensitivity, ProPer tracks the *provenance* of every record in the system, and uses the exact provenance to calculate how a query should affect the budgets of the individuals. Unlike PINQ, the system is described as an abstract formal model

for which we prove personalised differential privacy. This is important because the correctness of ProPer is not obvious for two reasons. Firstly, the individual budgets become highly sensitive and how we handle them is novel. More specifically, if a query involves records that would break the budget of an individual they are silently dropped from the data set upon which the query is calculated. In the example above, Tuesday's analysis of smokers will automatically exclude data derived from any diseased individuals as soon as the cost of the queries exceeds their budgets. Secondly, it is necessary to restrict the domain of computations over data sets to a class which guarantees that the provenance of any derived record is *affine* (zero or one record), otherwise the number of records which might get excluded due to a small change in the input might be too big to give privacy guarantees.

The approach is suitable for integration with other systems, since we assume the existence of basic primitives providing classical differential privacy. We have implemented a prototype of the ProPer approach which extends the PINQ system with personalised budgets and the ability to input live data. We compare the performance of our provenance-based implementation with PINQ to show that the runtime overhead is not significant.

1 Conclusion

We have introduced a new concept of personalised differential privacy (PDP) that improves the bookkeeping regarding the cost of composed queries, and makes it easy to include dynamic expansion of the data base. We have realised this idea in the design of ProPer, a system which enforces PDP for all (deterministic) client programs that compute against a simple API. We have proved that the ProPer model provides personalised – and therefore also standard – differential privacy.

References

1. Cynthia Dwork. Differential privacy. In *ICALP (2)*, volume 4052 of *LNCS*, pages 1–12. Springer, 2006.
2. Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.
3. Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1), January 2011.
4. Hamid Ebadi, David Sands, and Gerardo Schneider. Differential privacy: Now it's getting personal. In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15. ACM, 2015.
5. Andreas Haeberlen, Benjamin C Pierce, and Arjun Narayan. Differential privacy under fire. In *USENIX Security Symposium*, 2011.
6. Frank 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*, pages 19–30. ACM, 2009.
7. Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, pages 297–312. USENIX Association, 2010.