Metadata of the chapter that will be visualized in SpringerLink

Book Title	From Lambda Calculus to Cybersecurity through Program Analysis	
Series Title		
Chapter Title	New Program Abstractions for Privacy	
Copyright Year	2020	
Copyright HolderName	Springer Nature Switzerland AG	
Corresponding Author	Family Name	**Hunt**
	Particle	
	Given Name	**Sebastian**
	Prefix	
	Suffix	
	Role	
	Division	
	Organization	City, University of London
	Address	London, UK
	Email	sebastian.hunt.1@city.ac.uk
Author	Family Name	**Sands**
	Particle	
	Given Name	**David**
	Prefix	
	Suffix	
	Role	
	Division	
	Organization	Chalmers University of Technology
	Address	Gothenburg, Sweden
	Email	

Abstract

Static program analysis, once seen primarily as a tool for optimising programs, is now increasingly important as a means to provide quality guarantees about programs. One measure of quality is the extent to which programs respect the privacy of user data. Differential privacy is a rigorous quantified definition of privacy which guarantees a bound on the loss of privacy due to the release of statistical queries. Among the benefits enjoyed by the definition of differential privacy are compositionality properties that allow differentially private analyses to be built from pieces and combined in various ways. This has led to the development of frameworks for the construction of differentially private program analyses which are private-by-construction. Past frameworks assume that the sensitive data is collected centrally, and processed by a trusted curator. However, the main examples of differential privacy applied in practice - for example in the use of differential privacy in Google Chrome's collection of browsing statistics, or Apple's training of predictive messaging in iOS 10 -use a purely local mechanism applied at the data source, thus avoiding the collection of sensitive data altogether. While this is a benefit of the local approach, with systems like Apple's, users are required to completely trust that the analysis running on their system has the claimed privacy properties.

In this position paper we outline some key challenges in developing static analyses for analysing differential privacy, and propose novel abstractions for describing the behaviour of probabilistic programs not previously used in static analyses.

New Program Abstractions for Privacy

Sebastian Hunt[1(✉)] and David Sands[2]

[1] City, University of London, London, UK
sebastian.hunt.1@city.ac.uk
[2] Chalmers University of Technology, Gothenburg, Sweden

Abstract. Static program analysis, once seen primarily as a tool for optimising programs, is now increasingly important as a means to provide quality guarantees about programs. One measure of quality is the extent to which programs respect the privacy of user data. Differential privacy is a rigorous quantified definition of privacy which guarantees a bound on the loss of privacy due to the release of statistical queries. Among the benefits enjoyed by the definition of differential privacy are compositionality properties that allow differentially private analyses to be built from pieces and combined in various ways. This has led to the development of frameworks for the construction of differentially private program analyses which are private-by-construction. Past frameworks assume that the sensitive data is collected centrally, and processed by a trusted curator. However, the main examples of differential privacy applied in practice - for example in the use of differential privacy in Google Chrome's collection of browsing statistics, or Apple's training of predictive messaging in iOS 10 -use a purely local mechanism applied at the data source, thus avoiding the collection of sensitive data altogether. While this is a benefit of the local approach, with systems like Apple's, users are required to completely trust that the analysis running on their system has the claimed privacy properties.

In this position paper we outline some key challenges in developing static analyses for analysing differential privacy, and propose novel abstractions for describing the behaviour of probabilistic programs not previously used in static analyses.

1 Purpose and Aims

Differential privacy [6] perhaps represents the most rigorous and robust approach to privacy today. Unlike anonymisation methods which focus on properties of the data such as ensuring that there are several records with a given attribute, or that certain fields have been deleted, it is a property of the general mechanism (algorithm) used to release the data (and thus independent of the data itself); an algorithm which inputs sensitive data and outputs public data (typically used to compute some statistical property if the data) satisfies differential privacy if, for any input, adding or removing the data for any one individual makes very little observable difference to the overall result. For the purpose of this work we will not delve into the specific technical details of the definition.

A. Di Pierro et al. (Eds.): Festschrift Hankin, LNCS 12065, pp. 1–12, 2020.
https://doi.org/10.1007/978-3-030-41103-9_10

Example: Randomized Response. As an example, suppose that a data analyst wishes to answer the question: *what percentage of browser users have visited websites commonly used to facilitate the download of copyrighted material?* Suppose this information can be determined from the browsing history stored in your browser. One way to give a useful but necessarily approximate answer to this question, at the same time as limiting the privacy risk for the individual, is to use the following procedure: each respondent flips two coins; if both are heads then answer "Yes", if both are tails then answer "No", and otherwise answer the query truthfully. The data analyst is able to make a statistical estimate of the true percentage: if there are y "Yes" answers from 10000 respondents then we expect 2500 random "Yes" answers, 2500 random "No" answers, so the answer to the question can be estimated as $(y - 2500)/5000$. At the same time, anyone intercepting a "Yes" answer from any one individual cannot know whether it was generated by honesty or randomness – even if the response becomes public data the respondent can plausibly deny having visited such websites. In this specific example we can informally think of the increase in privacy risk as a multiplicative factor of $0.75/0.25 = 3$, so if there was already a 0.1% chance of, say, someone launching an investigation into whether a given IP address has been used to share copyrighted data, the risk in participation would at most increase to $3 \times 0.1\%$. By adjusting the probabilities of the coin flips one can increase the degree of privacy at the expense of either having lower accuracy in the reported result, or of requiring more data points to compensate for the increased noise.

This algorithm is differentially private [5]. Differential privacy is a parameterised definition, and the parameter, referred to as "epsilon", bounds (the natural logarithm of) how much multiplicative difference there is between an analysis using my data or someone else's. In the case of this particular algorithm we would say that it is ϵ-differentially private, with $\epsilon = \ln 3$. Every nontrivial differentially private algorithm operates by the addition of noise in some form. The particular algorithm described in the example is based on a 50-year-old survey technique called *randomised response*, and was designed to persuade respondents to tell truthful answers to potentially embarrassing or incriminating questions.

Frameworks for Differential Privacy. Differential privacy enjoys a number of useful properties that make it, in theory, an excellent foundation for robust privacy-aware information release. In particular it satisfies a number of useful compositional properties that allow the construction of differentially private algorithms from well-behaved data transformations and differentially private components. Making use of these, a number of differential privacy frameworks have emerged which support the construction of differentially private mechanisms. These leverage general compositional properties of differential privacy to simplify the static verification or dynamic enforcement of a desired amount of privacy. Examples of systems of this ilk are,

– PINQ [22], wPINQ [23], ProPer [11], and EKTELO [25] which dynamically monitor how data is used, and ensure that the computation never exceeds a given privacy budget, or

– Fuzz [13,15] and LightDP [26] where programs are written in a language with a custom type system or special verification annotations, and where static type checking/verification provides differential privacy guarantees.

All of these frameworks focus on verification of mechanisms which are assumed to have access to the whole data set. This implies the existence of a trusted database curator who holds the sensitive data, and who has the responsibility to apply the mechanism to the data and to keep track of a global privacy risk (Fig. 1). We refer to this as the *centralised model*.

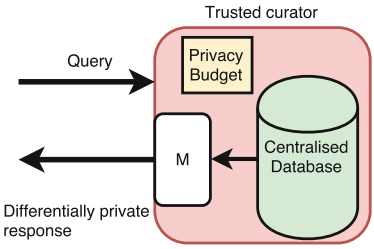

Fig. 1. Centralised Model (figure from [10])

Local Differential Privacy. The randomised response mechanism described above has a different trust model, and is called *local differential privacy* [4,19] or simply "the local model" [7]. In the local model the privacy mechanism is applied locally, at each respondent (Fig. 2).

The local model benefits from the fact that there is no longer a need to centrally store sensitive data – privacy is managed at the source (your cell-phone, car, web browser...). This removes the need for a trusted curator, and lowers the security risk of data breach. Perhaps for these reasons, the local model is the flavour of differential privacy which was first to be used in the actual "real-world" instances of differential privacy, by Google (in the Chrome browser) [12] and Apple (in iOS 10 and MacOS) [1].

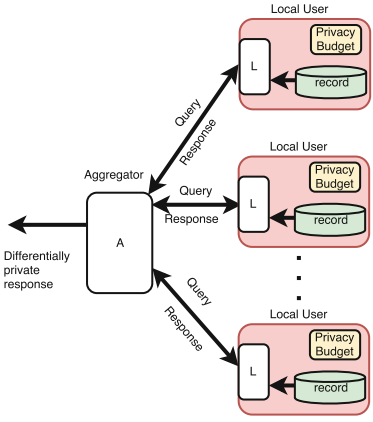

Fig. 2. Local Model (fig. from [10])

Research Goals

The local analyses by Apple and Google require a great deal of trust on the part of the user: you have to trust that they implemented their algorithms correctly on your device, and that the algorithms are indeed differentially private, not just for a single round of communication, but even when statistics are reported over time. Apple, in particular, did not initially report on the intended quantity of privacy ("epsilon") secret, and a recent reverse-engineering study of their algorithms [24] suggests that this trust is not well founded, and concludes

> "We call for Apple to make its implementation of privacy-preserving algorithms public and to make the rate of privacy loss fully transparent and tuneable by the user"

This comment aligns well with what we view as key design criteria for a **framework for local differential privacy**: the differential privacy properties of the mechanisms which deliver results based on sensitive data should be statically verifiable, and the verification should be simple enough to be done on the fly, for example in the respondent's device.

In rest of this paper we outline some key problems and possibilities in working towards statically verifiable local differential privacy. Most prior general frameworks for enforcing or verifying differential privacy are focused on the centralised model of a trusted curator providing access to a raw database. In Sect. 2 we outline one approach to verifiable local differential privacy, PreTPost, due to Ebadi and Sands [10], and discuss its limitations.

The limitations of PreTPost motivate a more general static analysis approach. We follow the philosophy outlined by Malacaria in factoring a quantitative information flow (QIF) analysis via a dependency analysis: [21]:

> "⟨the lattice of information⟩ allows for an elegant analysis decomposition of QIF into two steps, the first being an algebraic interpretation, the second being a numerical evaluation"

The "algebraic interpretation" referred to here is the use of equivalence relations, dubbed "the lattice of information" [20], but also known (in a more general form) as the lattice of *partial equivalence relations* where it was first used by the authors to express static analysis of dependency, referred to as a "binding time analysis" in [17], and as a "constancy analysis" in [16].

The gist of the approach articulated by Malacaria is to first determine which public outputs depend on which sensitive inputs, and then to instantiate that dependency numerically as a quantity. In our setting the quantity we want to measure is the epsilon of differential privacy: a bound on the largest proportional change in probability of obtaining any particular output when the user's sensitive input data is changed.

To this end we outline some specific challenges and opportunities in realising this programme for (local) differential privacy.

In Sect. 3 we discuss a shortcoming in dependency analyses that leads to imprecision in the quantitative step, namely the inability to describe a disjunctive dependency (to depend on one thing in some executions, or another thing in others, but never both in the same execution). This turns out to be a crucial distinction for properties such as differential privacy, because it allows us to use max rather than sum when we perform the quantitative instantiation step.

In Sect. 4 we propose a new way to instantiate privacy cost. Rather than directly instantiating with differential privacy costs (or their logarithm), our proposal is to abstract the behaviour of probabilistic programs (i.e. a new abstract domain) in a way which will provide greater precision and versatility; it is based on the idea of a geometric interpretation of differential privacy, a *privacy region* introduced by Kairouz, Oh, and Viswanath [18].

2 Verifiable Local Differential Privacy

PreTPost is a framework for implementing verifiable local differential privacy [10]. PreTPost leverages the simple observation that local differential privacy is preserved by arbitrary data pre-processing. The PreTPost framework requires a data analysis to be decomposed into a pre-processor (Pre), a simple core probabilistic transformation (T), and a post-processor (Post) (Fig. 3).

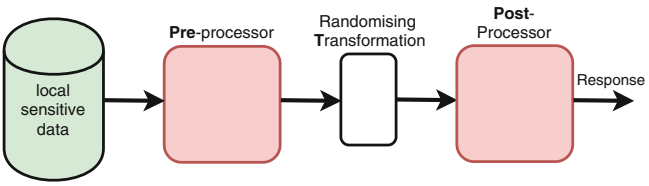

Fig. 3. PreTPost schema

The point of this schema is that an analysis delivered to a respondent in the form of a $\langle \text{Pre}, T, \text{Post} \rangle$ triple can be easily analysed: the quantity of differential privacy, often just referred to as "epsilon" but what we will refer to as the *privacy cost*, can be bounded by the cost of T alone since, unlike in the case of centralised differential privacy, the pre-processing cannot inflate the privacy cost of a subsequent differentially private operation. It is shown that this decomposition is possible and straightforward for a range of local analysis algorithms from the literature. Ebadi has implemented a prototype implementation of the PreTPost framework using sandboxed execution to prevent a malicious pre-processor from communicating the sensitive data directly, from bypassing the randomising transformation, or communicating via a covert timing channel [9].

Limitations of PreTPost. While the PreTPost schema provides a simple route to analysis of a proposed differentially private data processing algorithm, there are some limitations:

- The proposed algorithm must be refactored into the PreTPost format, which might not be the most natural way to express the algorithm.
- In practice (including in the PreTPost implementation) there are not only sensitive inputs, but also public inputs from other sources (data which is not considered sensitive) such as local data, as well as local public outputs (which may be used to modify subsequent public inputs). Although we don't anticipate major problems with these generalisations, they are still outside the simple schema of PreTPost.
- It cannot account for algorithms which reduce the sensitivity of the input data. For example, suppose an algorithm works by bitwise randomisation of a bit vector generated by a pre-processing of the sensitive input. If pre-processing yields an n-bit vector in which only a fixed number of bits k depend

on the sensitive data, then the privacy cost is k times the cost associated with the randomisation, whereas PreTPost necessarily assumes all n bits are sensitive; this example is reminiscent of the Bloom filter used in the full Rappor system [12];
- it cannot account for repeated randomisation (also a feature of the full Rappor system), i.e., where some randomised data is further randomised (not a common operation in differentially private algorithms, but used by the Rappor system).

While it is not clear the extent to which these limitations are show stoppers for PreTPost, by developing a more general and expressive static analysis that is not based on a fixed program schema we aim to gain a more fundamental understanding of the problem of abstracting and verifying differentially private algorithms. The remaining sections discuss some of the building blocks for such an analysis.

3 Dependency Analysis: The Need for Disjunction

Our aim is to address the limitations of PreTPost described above, in part by using dependency analysis to deal with the complex and subtle interdependencies that arise in realistic implementations and which prevent a simple, clean separation into three phases. However, it turns out that to get good results we need a semantic notion of dependency that is more expressive than the one used in standard dependency analyses. Moreover, this richer notion of dependency is relevant and potentially useful even when a mechanism *can* be cleanly separated in PreTPost style.

In this section we outline *why* we need a more general notion (a notion of *disjunction*), and *how* this might be represented in an analysis. We do not, however, go into the details of the semantic model for this generalised form, which is work in progress.

As it stands, PreTPost has nothing to say about how the post-processing phase uses the data supplied by the probabilistic transformation T. While post-processing the outputs of T can never increase their privacy cost, it can *decrease* it. Consider the code in Figs. 4, 5 and 6. Taken together these pieces of code define three alternative versions of a mechanism we call Three-Bits, which processes a 3-bit private value (an integer between 0 and 7, stored in `x`) and outputs a randomised result in `y`. All three versions share the same pre-processing and probabilistic phases but differ in their post-processing. The Pre-phase projects out the three bits of `x` into `a`, `b`, `c` and the T-phase independently randomises the bits. Suppose that the privacy cost of `Ran` is ϵ.

Post-processing (A) simply reassembles the randomised bits into a new 3-bit integer. Version (B) does the same but neglects to include bit `b`. Version (C) includes bit `a` in the result and, depending on `a`, includes either `b` or `c`, but never both together.

A standard dependency analysis will infer that, after (A), `y` depends on all three of `a`, `b`, `c` whereas after (B) it depends on `a` and `c` but not `b`. This allows

```
a = x % 2;
b = (x/2) % 2;
c = (x/4) % 2;
```

Fig. 4. Three-Bits: Pre

```
a = Ran(a);
b = Ran(b);
c = Ran(c);
```

Fig. 5. Three-Bits: T

```
y = a + 2*b + 4*c;
```

(A)

```
y = a + 4*c;
```

(B)

```
if (a == 0)
   y = a + 2*b;
else
   y = a + 4*c;
```

(C)

Fig. 6. Three-Bits: Post

us to infer that the privacy cost for Three-Bits-A is the sum of the costs for a, b, c (a total of 3ϵ) but the privacy cost for Three-Bits-B is the sum only of the costs for a and c (a total of 2ϵ).

Now consider (C). A standard dependency analysis will infer that, as for (A), y depends on a, b, c and an analysis using this dependency information would therefore also assign a privacy cost of 3ϵ to Three-Bits-C. But this is overly conservative. On any given run of (C), the value of y reveals *either* the values of a and c, *or* the values of a andc, but never all three together. This disjunctive dependency behaviour is reflected in the true privacy cost of Three-Bits-C, which is only 2ϵ. More generally, for a disjunctive dependency it turns out to be sound for differential privacy to take the *maximum* cost across the disjuncts, rather than the sum. Hence the cost for (C) may be calculated as $\max(\text{cost}(\mathtt{a}) + \text{cost}(\mathtt{b}), \text{cost}(\mathtt{a}) + \text{cost}(\mathtt{c})) = 2\epsilon$, in contrast with the cost for (A) which is $\text{cost}(\mathtt{a}) + \text{cost}(\mathtt{b}) + \text{cost}(\mathtt{c}) = 3\epsilon$.

A standard dependency analysis assigns to each output variable y a dependency set of input variables on which it may depend, ie a set X such that any choice of initial values for the variables in X completely determines the resulting value of y. Example (C) suggests that for our purposes it might be natural to lift such an analysis to represent a disjunctive dependency by a *set of sets* of variables. The dependency for y could then be represented as

$$\{\{\mathtt{a}, \mathtt{b}\}, \{\mathtt{a}, \mathtt{c}\}\}$$

Adapting standard approaches from abstract interpretation ([3,14]) it is relatively straightforward to lift an existing dependency analysis in this way (though the resulting algorithmic complexity may present practical challenges). However, it is not immediately obvious how to give a satisfactory *semantics* to such a set of sets. In particular, the required semantics is *not* simple logical disjunction of dependency properties: in example (C) it is *neither* true that the value of y is determined solely by $\{\mathtt{a}, \mathtt{b}\}$ *nor* that it is determined solely by $\{\mathtt{a}, \mathtt{c}\}$.

Our work in progress suggests that a satisfactory semantics for disjunctive dependency is obtainable by generalising the usual non-interference condition in

an appropriate way. We conclude this section with some hints at the direction in which we are aiming.

The standard non-interference condition can give meaning to e.g., "output x depends on inputs y and z" by using equivalence relations over states (mappings from variables to values). For this example we need two such relations, $=_{\{x\}}$, which relates two stores if they agree on the value of variable x, and $=_{\{y,z\}}$, which relates two stores if they agree on the values of both y and z. Then the semantics of "output x depends on inputs y and z" is: when the program is run on two stores related by $=_{\{y,z\}}$, you end up with two stores related by $=_{\{x\}}$.

While there are *specific* relations within the full lattice of equivalence relations on stores that exhibit disjunctive dependencies such as "output x depends on y and z *or* y and w", they can only express such a disjunction by saying exactly *how* it arises. For example we could build an equivalence relation on input stores that allows us to express the more specific disjunction "when $p(y)$ then x depends on y and z but otherwise it depends on y and w". What we are aiming for is a method that will allow us to give a semantics for such a disjunction more abstractly, directly from the relations $=_{\{y,z\}}$ and $=_{\{y,w\}}$, by working with suitable *sets* of relations that take into account all the ways that such a disjunction might arise.

4 The Abstract Domain of Privacy Regions

As mentioned in the introduction, for the quantitative phase of our analysis, we propose a novel abstract domain for analysis of probabilistic programs based on the idea of privacy regions. A key design goal is to allow us to leverage the framework of abstract interpretation [2]. Recall that abstract interpretation is a framework for semantics-based analysis which works by interpreting programs over an abstract domain in place of the concrete (standard) semantic domain, where each abstract value a denotes a concrete property γa. Each program construct is then given an abstract interpretation which soundly approximates its standard semantics: if a is mapped to a', the standard semantics transforms the property γa to some property $P \subseteq \gamma a'$. The abstract semantics of a program is then computed by a fixed-point iteration. The abstract domain is typically required to be a complete lattice and the design of the framework ensures that it is always sound to over-approximate by computing abstract values higher in the lattice. This freedom may be used (at the cost of reduced precision) to force the fixed-point iteration to converge within an acceptable time limit. Here, we give a lightweight, simplified account of the idea of privacy regions and explain why they have the appropriate structure to serve as an abstract domain.

To motivate the definitions we need to recall the generalisation of differential privacy known as *approximate* or (ϵ, δ)-differential privacy. In this variant (equivalent to ϵ-DP when $\delta = 0$) one weakens the requirement so that the differential privacy property may fail with some probability (typically very small) given by δ. This weakening can be used to obtain much better accuracy when composing analyses.

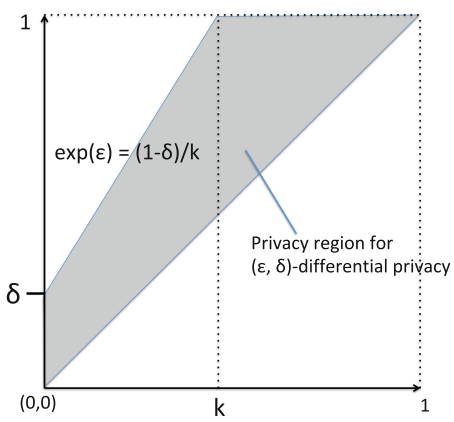

Fig. 7. (ϵ, δ) privacy region

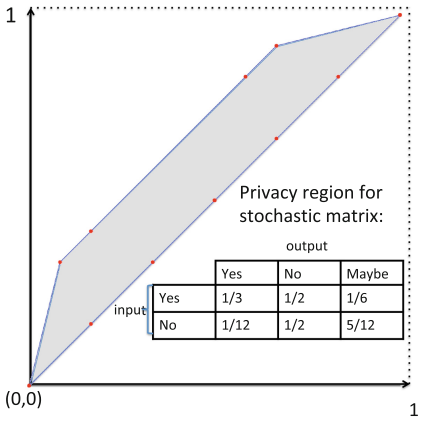

Fig. 8. Example privacy region

A key observation of [18] is that the (ϵ, δ)-differential privacy properties enjoyed by a mechanism can be characterised geometrically. For each choice of (ϵ, δ) define its *privacy region*[1] to be the closed region of the unit square above the line $x = y$ and below the line $y = e^{\epsilon}x + \delta$ as pictured in Fig. 7. Suppose that M has domain A and range B (ie M maps each value in A to a distribution over B). Then M is (ϵ, δ)-DP iff the following set is contained in the (ϵ, δ) privacy region:

$$\{(x, y) \mid a, a' \in A, S \subseteq B, x = \Pr[M(a) \in S], y = \Pr[M(a') \in S], y \geq x\} \quad (*)$$

(Note that any such set is symmetrical about $y = 1 - x$, because $\Pr[M(a) \in S] = 1 - \Pr[M(a) \in \overline{S}]$).

The privacy region for M, denoted $\mathcal{R}(M)$, is then defined as the intersection of *all* the (ϵ, δ) privacy regions such that M is (ϵ, δ)-DP, ie all the (ϵ, δ) privacy regions which contain the set (*). Equivalently, and more constructively, we can define $\mathcal{R}(M)$ to be the convex closure of (*): an example of a privacy region for a mechanism (specified as a stochastic matrix) is given in Fig. 8, where the points generated by this construction are marked. Note that each upper edge of a privacy region $\mathcal{R}(M)$ witnesses a distinct (ϵ, δ)-DP property, where ϵ is the log of the gradient and δ is the y-intercept; all of these properties hold simultaneously for M (and, indeed, for any mechanism whose privacy region is contained in $\mathcal{R}(M)$). In this example, the three upper edges witness the properties $(\epsilon = \ln 4, \delta = 0)$, $(\epsilon = 0, \delta = 1/4)$, and $(\epsilon = \ln(1/4), \delta = 3/4)$. (The two vertices of the middle edge – $(1/12, 4/12)$ and $(8/12, 11/12)$ – are generated by the output events $\{\text{Yes}\}$ and $\{\text{No}, \text{Maybe}\}$, respectively.)

[1] For convenience, our definition is a rotation by 90° in the unit square of the region defined by [18] and we restrict to the region above $y = x$ (their definition, after rotation, is symmetric about $y = x$).

Our key proposal is that the set of privacy regions forms a suitable abstract domain for static program analysis based on the principles of abstract interpretation (as outlined at the start of this section):

1. Privacy regions form a *complete lattice* ordered by subset inclusion, with the bottom element represented by the line $x = y$ ($(0,0)$-differential privacy) and top element being the upper left half of the unit square (no differential privacy), and the meet and join given by intersection and convex closure of the union, respectively.
2. The semantic content of a privacy region is all the (ϵ, δ)-regions in which it is contained, so any $R \supseteq \mathcal{R}(M)$ which can be inferred by analysis is a safe approximation for M.
3. Privacy regions subsume the notion of distance between distributions (as used in pure ϵ-DP): the "leading edge" of a privacy region (rising from $(0,0)$) defines an (ϵ, δ)-DP property where $\delta = 0$ and, by convexity, where ϵ is maximal.
4. Even when ϵ is the only property of interest, the extra information carried by privacy regions provides a better abstraction, yielding a more precise bound on ϵ than can be obtained using distance alone.
5. [18] provides a variety of useful results that characterise privacy regions, in particular describing the privacy region of a mechanism built from the composition of multiple mechanisms – very natural operations in a static program analysis.
6. We have been able to construct novel abstractions of function composition and pairing, which are essential ingredients of a static analysis.
7. There are many natural ways to safely coarsen a privacy region (to make it bigger, and thus less precise) – something which is a prerequisite to approximating fixed-point computations necessary to abstract the behaviour of iterative or recursive computation.

5 Conclusions

We have identified two key steps in our goal of constructing a static analysis of local differential privacy: the ability to abstract disjunctive dependency properties, and the use of privacy regions as an abstract domain for privacy cost. We have a promising approach for the semantics of disjunctive dependency, and believe this could be of independent interest. It remains, of course, to show that this can be combined with privacy regions to perform a useful static analysis.

Acknowledgements. This work was partly funded by the Swedish Foundation for Strategic Research (SSF) under the projects WebSec and by the Swedish Research Council (VR).

References

1. Apple Press Release: Apple previews iOS 10, the biggest iOS release ever (2016). https://www.apple.com/newsroom/2016/06/apple-previews-ios-10-biggest-ios-release-ever. Accessed 22 July 2017

2. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings 4th Annual ACM Symposium on Principles of Programming Languages, pp. 238–252 (1977)
3. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. POPL 1979, pp. 269–282. ACM, New York (1979). https://doi.org/10.1145/567752.567778
4. Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy and statistical minimax rates. In: 2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 1592–1592, October 2013. https://doi.org/10.1109/Allerton.2013.6736718
5. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006). https://doi.org/10.1007/11787006_1
6. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_14
7. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theoret. Comput. Sci. **9**, 211–407 (2014). https://doi.org/10.1561/0400000042
8. Ebadi, H.: Dynamic Enforcement of Differential Privacy. Ph.D. thesis, Chalmers University of Technology, March 2018
9. Ebadi, H.: The PreTPost Framework (2018). https://github.com/ebadi/preTpost
10. Ebadi, H., Sands, D.: PreTPost: a transparent, user verifiable, local differential privacy framework (2018). https://github.com/ebadi/preTpost. Also appears in [8]
11. Ebadi, H., Sands, D., Schneider, G.: Differential privacy: now it's getting personal. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL 2015, pp. 69–81. ACM (2015). https://doi.org/10.1145/2676726.2677005
12. Erlingsson, Ú., Pihur, V., Korolova, A.: RAPPOR: randomized aggregatable privacy-preserving ordinal response. In: CCS. ACM (2014)
13. Gaboardi, M., Haeberlen, A., Hsu, J., Narayan, A., Pierce, B.C.: Linear dependent types for differential privacy. In: Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL 2013, pp. 357–370. ACM, New York (2013). https://doi.org/10.1145/2429069.2429113
14. Giacobazzi, R., Ranzato, F.: Optimal domains for disjunctive abstract interpretation. Sci. Comput. Program. **32**(1), 177–210 (1998). https://doi.org/10.1016/S0167-6423(97)00034-8,http://www.sciencedirect.com/science/article/pii/S0167642397000348. 6th European Symposium on Programming
15. Haeberlen, A., Pierce, B.C., Narayan, A.: Differential privacy under fire. In: Proceedings of the 20th USENIX Conference on Security. SEC 2011, pp. 33–33. USENIX Association, Berkeley (2011). http://dl.acm.org/citation.cfm?id=2028067.2028100
16. Hunt, S.: Abstract interpretation of functional languages: from theory to practice. Ph.D. thesis, Imperial College London, UK (1991)
17. Hunt, S., Sands, D.: Binding time analysis: a new perspective. In: Proceedings of the ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation (PEPM 1991), pp. 154–164. ACM Press (1991)

18. Kairouz, P., Oh, S., Viswanath, P.: The composition theorem for differential privacy. IEEE Trans. Inf. Theory **63**(6), 4037–4049 (2017)
19. Kairouz, P., Oh, S., Viswanath, P.: Extremal mechanisms for local differential privacy. J. Mach. Learn. Res. **17**(17), 1–51 (2016). http://jmlr.org/papers/v17/15-135.html
20. Landauer, J., Redmond, T.: A lattice of information. In: CSFW (1993)
21. Malacaria, P.: Algebraic foundations for information theoretical, probabilistic an guessability measures of information flow. CoRR abs/1101.3453 (2011). http://arxiv.org/abs/1101.3453
22. McSherry, F.: Privacy integrated queries. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD). Association for Computing Machinery, Inc., June 2009
23. Proserpio, D., Goldberg, S., McSherry, F.: Calibrating data to sensitivity in private data analysis: a platform for differentially-private analysis of weighted datasets. Proc. VLDB Endow. **7**(8), 637–648 (2014). https://doi.org/10.14778/2732296.2732300
24. Tang, J., Korolova, A., Bai, X., Wang, X., Wang, X.: Privacy loss in Apple's implementation of differential privacy on MacOS 10.12. CoRR abs/1709.02753 (2017). http://arxiv.org/abs/1709.02753
25. Zhang, D., McKenna, R., Kotsogiannis, I., Hay, M., Machanavajjhala, A., Miklau, G.: EKTELO: a framework for defining differentially-private computations. In: Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, 10–15 June 2018, pp. 115–130 (2018). https://doi.org/10.1145/3183713.3196921
26. Zhang, D., Kifer, D.: LightDP: towards automating differential privacy proofs. In: POPL (2017)

Author Queries

Chapter 10

Query Refs.	Details Required	Author's response
AQ1	This is to inform you that corresponding author and email address has been identified as per the information available in the Copyright form.	
AQ2	Per Springer style, both city and country names must be present in the affiliations. Accordingly, we have inserted the city names in all affiliations. Please check and confirm if the inserted city names are correct. If not, please provide us with the correct city names.	