N

Probabilistic Relational Reasoning for Differential Privacy

GILLES BARTHE, BORIS KÖPF and FEDERICO OLMEDO, IMDEA Software Institute
SANTIAGO ZANELLA-BÉGUELIN, Microsoft Research

Differential privacy is a notion of confidentiality that allows useful computations on sensible data while protecting the privacy of individuals. Proving differential privacy is a difficult and error-prone task that calls for principled approaches and tool support. Approaches based on linear types and static analysis have recently emerged; however, an increasing number of programs achieve privacy using techniques that fall out of their scope. Examples include programs that aim for weaker, approximate differential privacy guarantees, and programs that achieve differential privacy without using any standard mechanisms. Providing support for reasoning about the privacy of such programs has been an open problem.

We report on CertiPriv, a machine-checked framework for reasoning about differential privacy built on top of the Coq proof assistant. The central component of CertiPriv is a quantitative extension of probabilistic relational Hoare logic that enables one to derive differential privacy guarantees for programs from first principles. We demonstrate the applicability of CertiPriv on a number of examples whose formal analysis is out of the reach of previous techniques. In particular, we provide the first machine-checked proofs of correctness of the Laplacian, Gaussian and Exponential mechanisms and of the privacy of randomized and streaming algorithms from the literature.

Categories and Subject Descriptors: D.3.1 [**Programming Languages**]: Formal Definitions and Theory; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*Program analysis*

General Terms: Languages, Security, Theory, Verification

Additional Key Words and Phrases: Coq proof assistant, differential privacy, relational Hoare logic

1. INTRODUCTION

When dealing with collections of private data one is faced with conflicting requirements: on the one hand, it is fundamental to protect the privacy of the individual contributors; on the other hand, it is desirable to maximize the utility of the data by mining and releasing partial or aggregate information, e.g. for medical statistics, market research, or targeted advertising. Differential privacy [Dwork et al. 2006b] is a quantitative notion of privacy that achieves an attractive trade-off between these two conflicting requirements: it provides strong confidentiality guarantees, yet it is permissive enough to allow for useful computations on private data. The key advantages of differential privacy over alternative definitions of privacy are its good behavior under

This article extends and generalizes the results presented in [Barthe et al. 2012]. In particular, it contains a novel connection of probabilistic lifting to network flow problems, an asymmetric version of apRHL, a proof of correctness of the Gaussian mechanism, a formal analysis of a privacy-preserving k-Median algorithm, and detailed descriptions of all proofs.

This work is supported by European Projects FP7-256980 NESSoS and FP7-229599 AMAROUT, Spanish project TIN2009-14599 DESAFIOS 10, Madrid Regional project S2009TIC-1465 PROMETIDOS and French project ANR SESUR-012 SCALP.

Author's addresses: G. Barthe, B. Köpf and F. Olmedo, IMDEA Software Institute, Campus de Montegancedo S/N, Madrid, Spain; S. Zanella-Béguelin, Microsoft Research, 21 Station Road, Cambridge CB1 2FB, UK.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

composition and its weak assumptions about the prior knowledge of adversaries. For a discussion of the guarantees provided by differential privacy and their limitations, see [Kasiviswanathan and Smith 2008; Kifer and Machanavajjhala 2011].

As the theoretical foundations of differential privacy become better understood, there is momentum to prove privacy guarantees of real systems. Several authors have recently proposed methods for reasoning about differential privacy on the basis of different languages and models of computation, e.g. SQL-like languages [McSherry 2009], higher-order functional languages [Reed and Pierce 2010], imperative languages [Chaudhuri et al. 2011], the MapReduce model [Roy et al. 2010], and I/O automata [Tschantz et al. 2011]. The unifying basis of these approaches are two key results: The first is the observation that one can achieve privacy by perturbing the output of a deterministic program by a suitable amount of symmetrically distributed noise, giving rise to the so-called Laplacian [Dwork et al. 2006b] and Exponential mechanisms [McSherry and Talwar 2007]. The second result are theorems that establish privacy bounds for the sequential and parallel composition of differentially private programs, see e.g. [McSherry 2009]. In combination, both results form the basis for creating and analyzing programs by composing differentially private building blocks.

While approaches relying on composing building blocks apply to an interesting range of examples, they fall short of covering the expanding frontiers of differentially private mechanisms and algorithms. Examples that cannot be handled by previous approaches include mechanisms that aim for weaker guarantees, such as approximate differential privacy [Dwork et al. 2006a], or randomized algorithms that achieve differential privacy without using any standard mechanism [Gupta et al. 2010]. Dealing with such examples requires fine-grained reasoning about the complex mathematical and probabilistic computations that programs perform on private input data. Such reasoning is particularly intricate and error-prone, and calls for principled approaches and tool support.

In this article we present a novel framework for formal reasoning about a large class of quantitative confidentiality properties, including (approximate) differential privacy and probabilistic non-interference. Our framework, coined CertiPriv, is built on top of the Coq proof assistant [The Coq development team 2010] and goes beyond the state-of-the-art in the following three aspects:

Expressivity: CertiPriv enables reasoning about a general and parametrized notion of confidentiality that encompasses differential privacy, approximate differential privacy, and probabilistic noninterference.

Flexibility: CertiPriv enables reasoning about the outcome of probabilistic computations from first principles. That is, instead of being limited to a fixed set of predefined building blocks one can define and use arbitrary building blocks, or reason about arbitrary computations using sophisticated machinery, without any limitation other than being elaborated from first principles. Proofs in CertiPriv can be verified independently and automatically by the Coq type checker.

Extensibility: CertiPriv inherits the generality of the Coq proof assistant and allows modeling and reasoning using arbitrary domains and datatypes. That is, instead of being confined to a fixed set of datatypes, CertiPriv can be extended on demand (e.g. with types and operators for graphs).

We illustrate the scope of CertiPriv by giving machine-checked proofs of four representative examples, some of which fall out of the scope of previous language-based approaches: (i) we prove the correctness of the Laplacian, Gaussian and Exponential mechanisms within our framework (rather than assuming their correctness as a meta-theorem), (ii) we prove the privacy of a randomized approximation algorithm for the

Minimum Vertex Cover problem [Gupta et al. 2010], (iii) we prove the privacy of a randomized approximation algorithm for the k-median problem [Gupta et al. 2010], and (iv) we prove the privacy of randomized algorithms for continual release of aggregate statistics of data streams [Chan et al. 2010]. Taken together, these examples demonstrate the generality and versatility of our approach.

As the first step in our technical development, we recast and generalize the definition of differential privacy. Informally, a probabilistic computation satisfies differential privacy if, independent of each individual's contribution to the dataset, the output distribution is essentially the same. More formally, a probabilistic program c is (ϵ, δ)-differentially private if and only if, given two initial memories m and m' that are adjacent (typically for a notion of adjacency that captures that m and m' differ in the contribution of one individual), the output distributions generated by c are related up to a multiplicative factor $\exp(\epsilon)$ and an additive term δ. That is, for every event E one requires

$$\Pr\left[c(m) : E\right] \leq \exp(\epsilon) \Pr\left[c(m') : E\right] + \delta$$

where $\Pr\left[c(m) : E\right]$ denotes the probability of event E in the distribution obtained by running c on initial memory m. The case of $\delta = 0$ corresponds to the vanilla definition of differential privacy [Dwork et al. 2006b], whereas cases with $\delta > 0$ correspond to *approximate* differential privacy [Dwork et al. 2006a]. For our development, we generalize (ϵ, δ)-differential privacy in two ways: First, we define (ϵ, δ)-differential privacy with respect to arbitrary relations Ψ on initial memories. The original definition is recovered by specializing Ψ to capture adjacency of memories. Second, we introduce a notion of distance (called α-distance) that generalizes statistical distance with a skew parameter α, and we show that a computation c is (ϵ, δ)-differentially private if and only if δ is an upper bound for the $\exp(\epsilon)$-distance between the output distributions obtained by running c on two memories m and m' satisfying Ψ. This generalization of differential privacy has the following two natural readings:

— The first reading is as an information flow property: if Ψ is an equivalence relation and $\epsilon = \delta = 0$, the definition states that the output distributions obtained by executing c in two related memories m and m' coincide, entailing that an adversary who can only observe the final distributions cannot distinguish between the two executions. Or, equivalently, by observing the output distributions, the adversary can only learn the initial memory up to its Ψ-equivalence class.
— The second reading is as a continuity property: if Ψ models adjacency between initial memories, the definition states that c is a continuous mapping between metric spaces, where α-distance is used as a metric on the set of output distributions.

We leverage on both readings to provide a fresh foundation for reasoning about differentially private computations. For this, we build on the observation that differential privacy can be construed as a quantitative 2-property [Terauchi and Aiken 2005; Clarkson and Schneider 2010]. Using this observation we define an approximate probabilistic Relational Hoare Logic (apRHL), following Benton's seminal use of relational logics to reason about information flow [Benton 2004]. Judgments in apRHL have the form

$$c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$$

Their validity implies that δ is an upper bound for the α-distance of the probability distributions generated by the probabilistic programs c_1 and c_2, modulo relational pre- and post-conditions Ψ and Φ on program states. For the special case where Φ is the equality on states, $c_1 = c_2 = c$, and $\alpha = \exp(\epsilon)$, the above judgment entails that the output distributions obtained by executing c starting from two initial memories related by Ψ are at α-distance at most δ, and hence that c is (ϵ, δ)-differentially private with respect to Ψ. As further detailed in Section 5.2, this intuitive understanding of apRHL judgments extends to the important case where Φ is an equivalence relation. With the view that Φ captures the observational capabilities of an adversary, such judgments simultaneously generalize differential privacy and notions of confidentiality as encountered in information flow analysis.

At the core of CertiPriv is a machine-checked proof system for reasoning about the validity of apRHL judgments, including rules for sequential and parallel composition and bounded loops, as well as rules corresponding to the Laplacian, Gaussian and Exponential mechanisms. The soundness of our proof system relies on the novel notion of (α, δ)-lifting of relations on states to relations on distributions over states, which crisply generalizes existing notions of lifting from probabilistic process algebra [Jonsson et al. 2001; Segala and Turrini 2007; Desharnais et al. 2008] and enjoys good closure properties. Moreover, we establish a connection between (α, δ)-liftings and maximum network flows that yields a means for deciding liftings of finite relations.

As bonus material, we present a variant of apRHL that supports reasoning about an asymmetric version of α-distance. Asymmetric apRHL strictly generalizes apRHL, as any proof in apRHL can be replaced by two (symmetric) proofs in asymmetric apRHL. However, reasoning in the asymmetric logic can lead to increased precision (i.e. better bounds), as we demonstrate in Section 6.4 on an approximation algorithm for the Minimum Vertex Cover problem [Gupta et al. 2010].

The basis of our formalization is CertiCrypt [Barthe et al. 2009], a machine-checked framework to verify cryptographic proofs in the Coq proof assistant. The outstanding difference between the two frameworks is that CertiPriv supports reasoning about a wide range of quantitative relational properties expressible in apRHL, whereas CertiCrypt is confined to baseline information flow properties that can be expressed in the fragment $(\alpha, \delta) = (1, 0)$. We refer to Section 8 for a more detailed comparison.

Summary of contributions. Our contributions are twofold. On the theoretical side, we lay the foundations for reasoning formally about an important and general class of approximate relational properties of probabilistic programs. Specifically, we introduce the notions of α-distance and (α, δ)-lifting, and an approximate probabilistic relational Hoare logic. On the practical side, we demonstrate the applicability of our approach by providing the first machine-checked proofs of differential privacy properties of fundamental mechanisms and complex approximation algorithms from the recent literature.

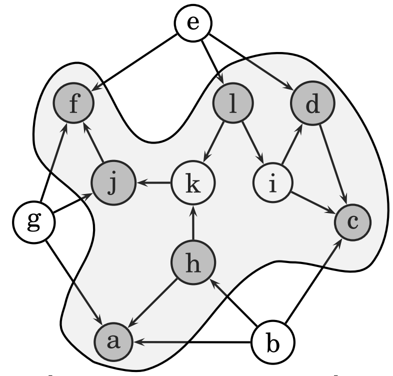
Organization of the article. The remainder of this article is structured as follows. In Section 2 we illustrate the application of our approach to an example algorithm; Section 3 introduces the representation of distributions and basic definitions used in the remainder. Section 4 presents the semantic foundations of apRHL, while Section 5 presents the core proof rules of the logic. Section 6 reports on case studies. Section 7 establishes a connection between the validity of apRHL judgments and network flow problems. We survey prior art and conclude in Sections 8 and 9. The Coq development containing machine-checked proofs of the results and examples in this article can be obtained from

`http://certicrypt.gforge.inria.fr/certipriv/`

Pencil-and-paper proofs of all key results can be found in the appendix.

2. ILLUSTRATIVE EXAMPLE

In this section we illustrate the applicability of our results by analyzing a differentially private approximation algorithm for the Minimum (Unweighted) Vertex Cover problem [Gupta et al. 2010].



$$\pi = [b, g, e, h, l, k, j, i, f, d, c, a]$$

Fig. 1. A minimum vertex cover (vertices in gray) and the cover given by a permutation π of the vertices in the graph (vertices inside the shaded area). The orientation of the edges is determined by π.

A *vertex cover* of an undirected graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that for any edge $(v, w) \in E$ either $v \in S$ or $w \in S$. The Minimum Vertex Cover problem is the problem of finding a vertex cover S of minimal size. In the privacy-preserving version of the problem the goal is to output a good approximation of a minimum cover while concealing the presence or absence of edges in the graph. Contrary to other optimization algorithms where the private data only determines the objective function (i.e. the size of a minimum cover), in the case of the Minimum Vertex Cover problem the edges in the graph determine the feasible solutions. This means that no privacy-preserving algorithm can explicitly output a vertex cover of size less than $n - 1$ for a graph with n vertices, for otherwise any pair of vertices absent from the output reveals the absence of an edge connecting them. To overcome this limitation, the algorithm that we analyze outputs an implicit representation of a cover as a permutation of the vertices in the graph. This output permutation determines an orientation of the edges in the graph by considering each edge as pointing towards the endpoint appearing last in the permutation. A vertex cover can then be recovered by taking for each edge the vertex it points to (Fig. 1). Alternatively, this implicit representation may be regarded as a privacy-preserving recipe for constructing a vertex cover in a distributed manner: the orientation of edges indicates how to reach a vertex in the cover from any given vertex in the graph.

The algorithm shown in Figure 2 is based on a randomized, albeit not privacy-preserving, approximation algorithm from [Pitt 1985] that achieves a constant approximation factor of 2 (i.e. the size of the computed cover is at most twice the size of a minimum vertex cover). The idea behind this algorithm is to iteratively pick a random uncovered edge and add one of its endpoints to the cover set, both the edge and the endpoint being chosen with uniform probability. Equivalently, this iterative process can be seen as selecting a vertex at random with probability proportional to its uncovered degree. This base algorithm can be transformed into a privacy-preserving algorithm by perturbing the distribution according to which vertices are sampled by a carefully calibrated weight factor that grows as more vertices are appended to the output permutation. This idea is implemented in the algorithm shown in Fig. 2, where at each iteration the instruction $v \overset{\$}{\leftarrow} \mathsf{choose}(V, E, \epsilon, n, i)$ chooses a vertex v from V with probability proportional to $d_E(v) + w_i$, where $d_E(v)$ denotes the degree of v in E and

$$w_i = \frac{4}{\epsilon}\sqrt{\frac{n}{n-i}}$$

```
function VertexCover(V, E, ε)
n ← |V|; π ← nil; i ← 0;
while i < n do
   v ←$ choose(V, E, ε, n, i);
   π ← v :: π;
   V ← V \ {v}; E ← E \ ({v} × V);
   i ← i + 1
end
```

Fig. 2. A differentially private approximation algorithm for the Minimum Vertex Cover problem

Put otherwise, the expression $\mathsf{choose}(V, E, \epsilon, n, i)$ denotes the discrete distribution over V whose probability mass function at v is

$$\frac{d_E(v) + w_i}{\displaystyle\sum_{x \in V} d_E(x) + w_i}$$

Consider two graphs $G_1 = (V, E)$ and $G_2 = (V, E \cup \{(t, u)\})$ with the same set of vertices but differing in exactly one edge. To prove that the above algorithm is ϵ-differentially private we must show that the probability of obtaining a permutation π of the vertices in the graph when the input is G_1 differs at most by a multiplicative factor $\exp(\epsilon)$ from the probability of obtaining π when the input is G_2, and vice versa. We show this using the approximate relational Hoare logic that we present in Section 5. We highlight here the key steps in the proof; a more detailed account appears in Section 6.4.

To establish the ϵ-differential privacy of algorithm VertexCover it suffices to prove the validity of the following judgment:

$$\models \textsc{VertexCover}(V, E, \epsilon) \sim_{\exp(\epsilon),0} \textsc{VertexCover}(V, E, \epsilon) : \Psi \Rightarrow \Phi \tag{1}$$

where

$$\Psi \stackrel{\text{def}}{=} V\langle 1\rangle = V\langle 2\rangle \wedge E\langle 2\rangle = E\langle 1\rangle \cup \{(t, u)\} \qquad \Phi \stackrel{\text{def}}{=} \pi\langle 1\rangle = \pi\langle 2\rangle$$

Assertions appearing in apRHL judgments, like Ψ and Φ above, are binary relations on program memories. We usually define assertions using predicate logic formulae over tagged program expressions. When defining an assertion $m_1 \; \Theta \; m_2$, we denote by $e\langle 1\rangle$ (resp. $e\langle 2\rangle$) the value that the expression e takes in memory m_1 (resp. m_2). For example, the post-condition Φ above denotes the relation $\{(m_1, m_2) \; : \; m_1(\pi) = m_2(\pi)\}$.

To prove the judgment above, we show privacy bounds for each iteration of the loop in the algorithm. Proving a bound for the i-th iteration boils down to proving a bound for the ratio between the probability of choosing a particular vertex in the left-hand side program and the right-hand side program, and its reciprocal. We distinguish three different cases, and use the fact that for a graph (V, E), $\sum_{x \in V} d_E(x) = 2|E|$ and the inequality $1 + x \leq \exp(x)$ to derive upper bounds in each case:

(a) the chosen vertex is not one of t, u and neither t nor u are in π.

$$\begin{aligned}
\frac{\Pr[v\langle 1\rangle = x]}{\Pr[v\langle 2\rangle = x]} &= \frac{(d_{E\langle 1\rangle}(x) + w_i) \sum_{y \in V}(d_{E\langle 2\rangle}(y) + w_i)}{(d_{E\langle 2\rangle}(x) + w_i) \sum_{y \in V}(d_{E\langle 1\rangle}(y) + w_i)} \\
&= \frac{(d_{E\langle 1\rangle}(x) + w_i)(2|E\langle 1\rangle| + (n-i)w_i + 2)}{(d_{E\langle 1\rangle}(x) + w_i)(2|E\langle 1\rangle| + (n-i)w_i)} \\
&\leq 1 + \frac{2}{(n-i)w_i} \leq \exp\left(\frac{2}{(n-i)w_i}\right)
\end{aligned}$$

$$\frac{\mathsf{Pr}[v\langle 2\rangle = x]}{\mathsf{Pr}[v\langle 1\rangle = x]} \leq 1$$

(b) the vertex v chosen in the iteration is one of t, u. We analyze the case where $v = t$, the other case is similar.

$$\frac{\mathsf{Pr}[v\langle 1\rangle = t]}{\mathsf{Pr}[v\langle 2\rangle = t]} \leq 1$$

$$\begin{aligned}\frac{\mathsf{Pr}[v\langle 2\rangle = t]}{\mathsf{Pr}[v\langle 1\rangle = t]} &= \frac{(w_i + d_{E\langle 1\rangle}(t) + 1)(2|E\langle 1\rangle| + (n-i)w_i)}{(w_i + d_{E\langle 1\rangle}(t))(2|E\langle 1\rangle| + (n-i)w_i + 2)} \\ &\leq 1 + w_i^{-1} \leq 1 + w_0^{-1} \leq \exp(\epsilon/4)\end{aligned}$$

(c) either t or u is already in π, in which case both executions are observationally equivalent and do not add to the privacy bound.

$$\frac{\mathsf{Pr}[v\langle 1\rangle = x]}{\mathsf{Pr}[v\langle 2\rangle = x]} = \frac{\mathsf{Pr}[v\langle 2\rangle = x]}{\mathsf{Pr}[v\langle 1\rangle = x]} = 1$$

Case (a) can occur at most $(n-2)$ times, while case (b) occurs exactly once. Thus, multiplying the bounds over all n iterations and recalling inequality $\sum_{i=1}^{n} 1/\sqrt{i} \leq 2\sqrt{n}$, one gets

$$\frac{\Pr\left[\textsc{VertexCover}(G_1, \epsilon) : \pi = \vec{v}\right]}{\Pr\left[\textsc{VertexCover}(G_2, \epsilon) : \pi = \vec{v}\right]} \leq \exp\left(\sum_{i=0}^{n-3} \frac{2}{(n-i)w_i}\right) \leq \exp(\epsilon)$$

$$\frac{\Pr\left[\textsc{VertexCover}(G_2, \epsilon) : \pi = \vec{v}\right]}{\Pr\left[\textsc{VertexCover}(G_1, \epsilon) : \pi = \vec{v}\right]} \leq \exp\left(\epsilon/4\right) \leq \exp(\epsilon)$$

The above informal reasoning is captured by a proof rule for loops parametrized by an invariant and a stable property of the product state of both executions (i.e. a relation that once established remains true); this rule is described in Section 5.3. We use the following loop invariant (note that if pre-condition Ψ above holds, the invariant is established by the initialization code appearing before the loop):

$$\begin{array}{l}(t \in \pi\langle 1\rangle \vee u \in \pi\langle 1\rangle \implies E\langle 1\rangle = E\langle 2\rangle) \wedge \\ (t \notin \pi\langle 1\rangle \wedge u \notin \pi\langle 1\rangle \implies E\langle 2\rangle = E\langle 1\rangle \cup \{(t,u)\}) \wedge \\ V\langle 1\rangle = V\langle 2\rangle \wedge \pi\langle 1\rangle = \pi\langle 2\rangle\end{array}$$

and the following stable property:

$$t \in \pi\langle 1\rangle \vee u \in \pi\langle 1\rangle$$

The application of this proof rule requires to prove three judgments as premises, corresponding to each one of the cases detailed above; we detail them in Section 6.4.

3. PRELIMINARIES

3.1. Probabilities and Reals

In the course of our Coq formalization, we have found it convenient to reason about probabilities using the axiomatization of the unit interval $[0, 1]$ provided by the ALEA library of Audebaud and Paulin [Audebaud and Paulin-Mohring 2009]. Their formalization supports as primitive operations addition, inversion, multiplication, and division, and proves that the unit interval $[0, 1]$ can be given the structure of a ω-cpo by taking as order the usual $\leq$ relation and by defining an operator $\sup$ that computes the least upper bound of monotonic $[0, 1]$-valued sequences.

In order to manage the interplay between the formalizations of the unit interval and of the reals, we have axiomatized an embedding/retraction pair between them and built an extensive library of results about the relationship between arithmetic operations in the two domains, e.g.:

Addition. $x +_{[0,1]} y = \min_{\mathbb{R}}(x +_{\mathbb{R}} y, 1)$;
Inversion. $-_{[0,1]}x = 1 -_{\mathbb{R}} x$;
Multiplication. $x \times_{[0,1]} y = x \times_{\mathbb{R}} y$;
Division. If $y \neq 0$, then $x/_{[0,1]}y = \min_{\mathbb{R}}(x/_{\mathbb{R}}y, 1)$.

3.2. Distributions

We view a distribution μ over a set A as a function of type

$$(A \to [0,1]) \to [0,1]$$

that maps a unit-valued random variable (a function in $A \to [0,1]$) to its expected value [Ramsey and Pfeffer 2002; Audebaud and Paulin-Mohring 2009]. When applied to an event $E \subseteq A$ represented by its characteristic function $\mathbb{1}_E \colon A \to [0,1]$, $\mu\ \mathbb{1}_E$ corresponds to the probability of E. When applied to singleton events $E = \{a\}$, $\mu\ \mathbb{1}_{\{a\}}$ corresponds to the probability mass of μ at a, and we denote it using the shorthand $\mu(a)$. When applied to arbitrary functions $f : A \to [0,1]$, $\mu\ f$ gives the expectation of f w.r.t. μ. For discrete distributions μ, the connection between density and expectation is given by the following equation:

$$\mu\ f = \sum_{a \in A} \mu(a)\ f(a)$$

In this formalism, the Bernoulli distribution over $\mathbb{B}$ with success probability p, for instance, is represented as $\lambda f.\ p\ f(\mathsf{true}) + (1-p)\ f(\mathsf{false})$; the distribution over $\mathbb{N}$ that assigns probability $(1/2)^i$ to number i is given by $\lambda f.\ \sum_{i \in \mathbb{N}} (1/2)^i\ f(i)$.

Formally, a distribution over A is a function μ of type $(A \to [0,1]) \to [0,1]$ together with proofs of the following properties:

Monotonicity. $f \leq g \implies \mu\ f \leq \mu\ g$;
Compatibility with inverse. $\mu\ (\mathbb{1} - f) \leq 1 - \mu\ f$, where $\mathbb{1}$ is the constant function 1;
Additive linearity. $f \leq \mathbb{1} - g \implies \mu\ (f+g) = \mu\ f + \mu\ g$;
Multiplicative linearity. $\mu\ (k \times f) = k \times \mu\ f$;
Continuity. If $F : \mathbb{N} \to (A \to [0,1])$ is monotonic, then $\mu\ (\sup\ F) \leq \sup\ (\mu \circ F)$

In the statement of the above properties, arithmetic is performed in the interval $[0,1]$ (underflows and overflows are mapped to 0 and 1, respectively). For the sake of readability, we drop the $[0,1]$ sub-index of operators. Functions f and g are universally quantified over the space of functions $A \to [0,1]$ and constant k is universally quantified over the interval $[0,1]$.

Note that we do not require that $\mu\ \mathbb{1} = 1$, and thus, strictly speaking, our definition corresponds to sub-probability distributions. This provides an elegant means of giving semantics to runtime assertions and programs that do not terminate with probability one. We let $\mathcal{D}(A)$ denote the set of sub-probability distributions over A and μ_0 denote the null distribution.

Distributions can be given the structure of a monad; this monadic view eliminates the need for cluttered definitions and proofs involving summations, and allows to give a continuation-passing style semantics to probabilistic programs. Formally, we define the unit and bind operators as follows:

$$\begin{array}{rcl} \mathsf{unit} & : & A \to \mathcal{D}(A) \\ & \stackrel{\text{def}}{=} & \lambda x.\ \lambda f.\ f\ x \\ \mathsf{bind} & : & \mathcal{D}(A) \to (A \to \mathcal{D}(B)) \to \mathcal{D}(B) \\ & \stackrel{\text{def}}{=} & \lambda \mu.\ \lambda M.\ \lambda f.\ \mu\ (\lambda\ x.\ M\ x\ f) \end{array}$$

The unit operator maps $x \in A$ to the Dirac measure δ_x at point x; in the discrete case unit x is the degenerate probability distribution that has all its mass concentrated at x. The bind operator takes a distribution on A and a conditional distribution on B given A, and returns the corresponding marginal distribution on B.

In the remainder we use the following operations and relations:

$$\begin{array}{rcl} \mathsf{range}\ P\ \mu & \stackrel{\text{def}}{=} & \forall f.\ (\forall a.\ P\ a \implies f\ a = 0) \implies \mu\ f = 0 \\ \pi_1(\mu) & \stackrel{\text{def}}{=} & \mathsf{bind}\ \mu\ (\lambda(x,y).\ \mathsf{unit}\ x) \\ \pi_2(\mu) & \stackrel{\text{def}}{=} & \mathsf{bind}\ \mu\ (\lambda(x,y).\ \mathsf{unit}\ y) \\ \mu \le \mu' & \stackrel{\text{def}}{=} & \forall f.\ \mu\ f \le \mu'\ f \end{array}$$

The formula range $P\ \mu$ implies that elements of A with a non-null probability w.r.t. μ satisfy predicate P (we prove this as Lemma A.2 in the Appendix). To see why, consider the contrapositive claim which says that elements of A satisfying $\neg P$ have null probability; the formula range $P\ \mu$ readily gives $\mu\ \mathbb{1}_{\neg P} = 0$. For a distribution μ over a product type $A \times B$, $\pi_1(\mu)$ (resp. $\pi_2(\mu)$) defines its projection on the first (resp. second) component. Finally, $\le$ defines a pointwise partial order on $\mathcal{D}(A)$.

4. FIRST PRINCIPLES

4.1. Skewed Distance between Distributions

In this section we define the notion of α-distance, a parametrized distance between distributions. We show how this notion can be used to express ϵ-differential privacy, (ϵ, δ)-differential privacy, and statistical distance.

We begin by augmenting the Euclidean distance between reals a and b ($|a - b| = \max\{a - b, b - a\}$) with a skew parameter $\alpha \ge 1$, which will later play the role of the factor $\exp(\epsilon)$ in the definition of differential privacy. Namely, we define the α-distance $\Delta_\alpha(a, b)$ between a and b as

$$\Delta_\alpha(a, b) \stackrel{\text{def}}{=} \max\{a - \alpha b, b - \alpha a, 0\}$$

Note that Δ_α is non-negative by definition and that Δ_1 coincides with the standard distance between reals. We extend Δ_α to a distance between distributions as follows.

Definition 4.1 (*α-distance*). For $\alpha \ge 1$, the α-distance $\Delta_\alpha(\mu_1, \mu_2)$ between two distributions μ_1 and μ_2 is defined as:

$$\Delta_\alpha(\mu_1, \mu_2) \stackrel{\text{def}}{=} \max_{f : A \to [0,1]} \Delta_\alpha(\mu_1\ f, \mu_2\ f)$$

The condition $\alpha \ge 1$ is natural when one thinks of differential privacy, and is required to have e.g. $\Delta_\alpha(\mu, \mu) = 0$.

The definition of α-distance considers all unit-valued functions. The next lemma shows that for discrete distributions this definition is equivalent to an alternative definition that considers only Boolean-valued functions, i.e. those corresponding to characteristic functions of events.

LEMMA 4.2. *For all distributions μ_1 and μ_2 over a discrete set A,*

$$\Delta_\alpha(\mu_1, \mu_2) = \max_{E \subseteq A} \Delta_\alpha(\mu_1\ \mathbb{1}_E, \mu_2\ \mathbb{1}_E)$$

An immediate consequence of Lemma 4.2 is that Δ_1 coincides with the standard notion of statistical (i.e. total variation) distance:

$$\Delta_1(\mu_1, \mu_2) = \max_{E \subseteq A} |\mu_1\ \mathbb{1}_E - \mu_2\ \mathbb{1}_E|$$

We state some basic properties of α-distance; these properties are the keystone for reasoning about approximate liftings and for proving the soundness of our logic. All properties are implicitly universally quantified.

LEMMA 4.3 (PROPERTIES OF α-DISTANCE).

(*1*) $0 \leq \Delta_\alpha(\mu_1, \mu_2) \leq 1$
(*2*) $\Delta_\alpha(\mu, \mu) = 0$
(*3*) $\Delta_\alpha(\mu_1, \mu_2) = \Delta_\alpha(\mu_2, \mu_1)$
(*4*) $\Delta_{\alpha\alpha'}(\mu_1, \mu_3) \leq \max(\alpha' \Delta_\alpha(\mu_1, \mu_2) + \Delta_{\alpha'}(\mu_2, \mu_3), \Delta_\alpha(\mu_1, \mu_2) + \alpha\ \Delta_{\alpha'}(\mu_2, \mu_3))$
(*5*) $\alpha \leq \alpha' \implies \Delta_{\alpha'}(\mu_1, \mu_2) \leq \Delta_\alpha(\mu_1, \mu_2)$
(*6*) $\Delta_\alpha(\mathsf{bind}\ \mu_1\ M, \mathsf{bind}\ \mu_2\ M) \leq \Delta_\alpha(\mu_1, \mu_2)$

Most of the above properties are self-explanatory; we briefly highlight the most important ones. Property (4) generalizes the triangle inequality with appropriate skew factors; (5) states that α-distance is anti-monotonic with respect to α; (6) states that probabilistic computation does not increase the distance (which is a well-known fact for statistical distance); Lemma A.5 in the appendix further generalizes this result.

4.2. Differential Privacy

Differential privacy is a condition on the distance between the output distributions produced by a randomized algorithm. Namely, for a given metric on the input space, differential privacy requires that, for any pair of inputs at distance at most 1, the probability that an algorithm outputs a value in an arbitrary set differs at most by a multiplicative factor of $\exp(\epsilon)$. *Approximate* differential privacy relaxes this requirement by additionally allowing for an additive slack δ. The following definition captures these requirements in terms of α-distance; Lemma 4.2 establishes the equivalence to the original definition [Dwork et al. 2006a] for algorithms with discrete output.

Definition 4.4 (*Approximate differential privacy*). Let d be a metric on A. A randomized algorithm $M : A \to \mathcal{D}(B)$ is (ϵ, δ)-differentially private (w.r.t. d) iff

$$\forall a, a' \in A.\ d(a, a') \leq 1 \implies \Delta_{\exp(\epsilon)}(M\ a, M\ a') \leq \delta$$

For algorithms that terminate with probability 1 (i.e. when $(M\,a)\,\mathbb{1}_B = 1$ for all $a \in A$), the above definition corresponds to standard approximate differential privacy [Dwork et al. 2006a], which assumes that an adversary can only observe the result of a query. In particular, $(\epsilon, 0)$-differential privacy corresponds to ϵ-differential privacy.

As the following example shows, Definition 4.4 does not imply termination-sensitive differential privacy. Let $A = \{a, a'\}$, $B = \{b\}$ and $d(a, a') \leq 1$ and consider the algorithm $M : A \to \mathcal{D}(B)$ such that $M\ a$ returns b with probability 1, and $M\ a'$ returns b with probability $1/2$, but loops with probability $1/2$. Algorithm M satisfies Definition 4.4 for $\delta = 0$ and $\epsilon \geq \ln(2)$. However, for any ϵ,

$$\frac{1}{2} = 1 - (M\ a')\ \mathbb{1}_B > \exp(\epsilon)\ (1 - (M\ a)\ \mathbb{1}_B) = 0$$

which would violate privacy when an adversary can observe non-termination.

A termination-sensitive definition of differential privacy can be obtained by considering in Definition 4.4 the extension $M_\perp$ of M to $B_\perp = B \cup \{\perp\}$, letting $(M_\perp\ a)\ \mathbb{1}_{\{\perp\}} =$

$1 - (M\ a)\ \mathbb{1}_B$. As the following lemma shows, we can account for differences in termination on adjacent inputs by shifting these differences to the additive slack.

LEMMA 4.5. *Let M be an (ϵ, δ)-differentially private algorithm. Then, $M_\perp$ is $(\epsilon, \delta + \delta')$-differentially private, where*

$$\delta' = \max_{\{a,a' \mid d(a,a') \le 1\}} |(M\ a)\ \mathbb{1}_B - (M\ a')\ \mathbb{1}_B|$$

PROOF. Let $\alpha = \exp(\epsilon)$; a direct calculation shows that for $E \subseteq B_\perp$ we have

$$\begin{aligned}
&\Delta_\alpha((M_\perp\ a)\ \mathbb{1}_E, (M_\perp\ a')\ \mathbb{1}_E) \\
&\quad \le \Delta_\alpha((M_\perp\ a)\ \mathbb{1}_{E\setminus\{\perp\}}, (M_\perp\ a')\ \mathbb{1}_{E\setminus\{\perp\}}) + \Delta_\alpha((M_\perp a)\ \mathbb{1}_{\{\perp\}}, (M_\perp\ a')\ \mathbb{1}_{\{\perp\}}) \\
&\quad = \Delta_\alpha((M\ a)\ \mathbb{1}_{E\setminus\{\perp\}}, (M\ a')\ \mathbb{1}_{E\setminus\{\perp\}}) + \Delta_\alpha(1 - (M\ a)\ \mathbb{1}_B, 1 - (M\ a')\ \mathbb{1}_B) \le \delta + \delta' \qquad \square
\end{aligned}$$

Timing channels are as problematic as termination channels. They could be taken into account by defining a cost model for programs and treating the cost of executing a program as an observable output. CertiCrypt provides a cost-instrumented semantics (used for capturing *probabilistic polynomial-time* complexity) that can be readily used to capture privacy leaks through timing channels. Although, as we showed, richer models may be used to account for information leaked through side-channels, these are best mitigated by means of independent countermeasures (see [Haeberlen et al. 2011] for an excellent analysis of the space of possible solutions).

Finally, it is folklore that for discrete domains the definition of differential privacy is equivalent to its pointwise variant where one quantifies over characteristic functions of singleton sets rather than those of arbitrary sets; however, this equivalence breaks when considering approximate differential privacy [Dwork et al. 2006a]. The following lemma provides a way to establish bounds for α-distance (and hence for approximate differential privacy) in terms of characteristic functions of singleton sets. Note that the inequality is strict in general.

LEMMA 4.6. *For all distributions μ_1 and μ_2 over a discrete set A,*

$$\Delta_\alpha(\mu_1, \mu_2) \le \sum_{a \in A} \Delta_\alpha(\mu_1(a), \mu_2(a))$$

4.3. Approximate Lifting of Relations to Distributions

In Section 5 we present apRHL, a relational logic for reasoning about probabilistic programs that elaborates on Benton's relational Hoare logic [Benton 2004]. Judgments in Benton's logic are of the form $c_1 \sim c_2 : \Psi \Rightarrow \Phi$, where c_1, c_2 are deterministic programs, and assertions Ψ, Φ are binary relations over program memories. The validity of such a judgment requires that terminating executions of programs c_1 and c_2 in initial memories related by Ψ result in final memories related by Φ. In the logic we consider in next section, judgments have the same shape: assertions are still binary relations over program memories, but programs are probabilistic. Since in this setting a program execution results in a distribution over memories rather than a single final memory, in order to extend Benton's logic to probabilistic programs, we need a means of lifting the post-condition Φ to distributions.

In this section we introduce a notion of *approximate* lifting of binary relations over sets to distributions over those sets, which is the cornerstone for defining validity of apRHL judgments.

Given $\alpha \in \mathbb{R}^{\ge 1}$ and $\delta \in [0, 1]$, the (α, δ)-lifting of $R \subseteq A \times B$ is a relation between $\mathcal{D}(A)$ and $\mathcal{D}(B)$. Two distributions $\mu_1 \in \mathcal{D}(A)$ and $\mu_2 \in \mathcal{D}(B)$ are related by the (α, δ)-lifting of R, whenever there exists a distribution over $A \times B$ whose support is contained in R and whose first and second projections are at most at α-distance δ of μ_1 and μ_2, respectively.

Definition 4.7 (*Lifting*). Let $\alpha \in \mathbb{R}^{\geq 1}$ and $\delta \in [0,1]$. The (α,δ)-lifting of a relation $R \subseteq A \times B$ is the relation $\sim_R^{\alpha,\delta} \subseteq \mathcal{D}(A) \times \mathcal{D}(B)$ such that $\mu_1 \sim_R^{\alpha,\delta} \mu_2$ iff there exists $\mu \in \mathcal{D}(A \times B)$ satisfying

(1) $\mathsf{range}\ R\ \mu$,
(2) $\pi_1\ \mu \leq \mu_1\ \wedge\ \pi_2\ \mu \leq \mu_2$, and
(3) $\Delta_\alpha(\pi_1\ \mu, \mu_1) \leq \delta\ \wedge\ \Delta_\alpha(\pi_2\ \mu, \mu_2) \leq \delta$

We say that a distribution μ satisfying the above conditions is a *witness* for the lifting.

The notion of (α,δ)-lifting generalizes previous notions of lifting, such as that of [Jonsson et al. 2001], which is obtained by taking $\alpha = 1$ and $\delta = 0$, and δ-lifting [Segala and Turrini 2007; Desharnais et al. 2008], obtained by taking $\alpha = 1$.

In the case of equivalence relations, the notion of (α,δ)-lifting admits a more intuitive characterization. Specifically, if R is an equivalence relation over A, then μ_1 and μ_2 are related by the (α,δ)-lifting of R iff the pair of distributions that μ_1 and μ_2 induce on the quotient set A/R are at α-distance at most δ. Formally, we define the distribution induced by $\mu \in \mathcal{D}(A)$ on the quotient set A/R as $(\mu/R)([a]) \stackrel{\text{def}}{=} \mu([a])$.

LEMMA 4.8. *Let R be an equivalence relation over a discrete set A and let $\mu_1, \mu_2 \in \mathcal{D}(A)$. Then,*

$$\mu_1 \sim_R^{\alpha,\delta} \mu_2 \iff \Delta_\alpha(\mu_1/R, \mu_2/R) \leq \delta$$

Jonsson et al. also show that for equivalence relations, their definition of lifting coincides with the more intuitive notion that requires related distributions assign equal probabilities to all equivalence classes [Jonsson et al. 2001]. This result can be recovered from Lemma 4.8 by taking $(\alpha,\delta) = (1,0)$.

The next lemma shows that (α,δ)-lifting is monotonic w.r.t. the slack δ, the skew factor α, and the relation R. An immediate consequence is that for $\alpha > 1$, (α,δ)-lifting is more permissive than the previously proposed notions of lifting.

LEMMA 4.9. *For all $1 \leq \alpha \leq \alpha'$ and $\delta \leq \delta'$, and relations $R \subseteq S$,*

$$\mu_1 \sim_R^{\alpha,\delta} \mu_2 \implies \mu_1 \sim_S^{\alpha',\delta'} \mu_2$$

We next present a fundamental property of (α,δ)-lifting, which is central to the applicability of apRHL to reason about α-distance (and hence differential privacy). Namely, two distributions related by the (α,δ)-lifting of R yield probabilities that are within α-distance of δ when applied to R-equivalent functions. Given $R \subseteq A \times B$ we say that two functions $f : A \to [0,1]$ and $g : B \to [0,1]$ are *R-equivalent*, and write $f =_R g$, iff for every $a \in A$ and $b \in B$, $R\ a\ b$ implies $f\ a = g\ b$. In what follows we use $\equiv$ to denote the identity relation over arbitrary sets.

THEOREM 4.10 (FUNDAMENTAL PROPERTY OF LIFTING). *Let $R \subseteq A \times B$, $\mu_1 \in \mathcal{D}(A)$ and $\mu_2 \in \mathcal{D}(B)$. Then, for any two functions $f_1 : A \to [0,1]$ and $f_2 : B \to [0,1]$,*

$$\mu_1 \sim_R^{\alpha,\delta} \mu_2 \wedge f_1 =_R f_2 \implies \Delta_\alpha(\mu_1\ f_1, \mu_2\ f_2) \leq \delta$$

In particular, when $A = B$ and R is the identity relation,

$$\mu_1 \sim_\equiv^{\alpha,\delta} \mu_2 \implies \Delta_\alpha(\mu_1, \mu_2) \leq \delta$$

Theorem 4.10 provides an interpretation of (α,δ)-lifting in terms of α-distance. Next we present two results that enable us to actually construct witnesses for such liftings.

The first result is the converse of Theorem 4.10 for the special case of R being the identity relation: we prove that two distributions are related by the (α,δ)-lifting of the identity relation if their α-distance is smaller than δ. This result is used to prove the soundness of the logic rule for random assignments given in the next section.

THEOREM 4.11. *Let μ_1 and μ_2 be distributions over a discrete set A. Then*

$$\Delta_\alpha(\mu_1, \mu_2) \leq \delta \implies \mu_1 \sim_{\equiv}^{\alpha,\delta} \mu_2$$

The proof is immediate by considering as a witness for the lifting the distribution with the following probability mass function:

$$\mu(a, a') = \begin{cases} \min(\mu_1(a), \mu_2(a)) & \text{if } a = a' \\ 0 & \text{if } a \neq a' \end{cases}$$

As a side remark, observe that the equivalence $\Delta_\alpha(\mu_1, \mu_2) \leq \delta \iff \mu_1 \sim_{\equiv}^{\alpha,\delta} \mu_2$ is immediate from Lemma 4.8. However, we prefer to keep separate statements and proofs for each direction (Theorems 4.10 and 4.11), because these correspond to theorems in our Coq formalization, while we only give a pencil-and-paper proof of Lemma 4.8 in the Appendix.

The second result shows that (α, δ)-liftings compose. This enables one to derive a judgment relating two programs c_1 and c_2 by introducing an intermediate program c and proving the validity of judgments relating c_1 and c on one hand, and c and c_2 on the other hand. This result is used in the examples of Section 6.2, and more extensively in cryptographic proofs, see e.g. [Barthe et al. 2009].

THEOREM 4.12. *Let μ_1, μ_2 and μ_3 be distributions over discrete sets A, B, and C, respectively. Let $R \subseteq A \times B$ and $S \subseteq B \times C$. For all $\alpha, \alpha' \in \mathbb{R}^{\geq 1}$ and $\delta, \delta' \in [0, 1]$,*

$$\mu_1 \sim_R^{\alpha,\delta} \mu_2 \wedge \mu_2 \sim_S^{\alpha',\delta'} \mu_3 \implies \mu_1 \sim_{R \circ S}^{\alpha\alpha',\delta''} \mu_3$$

where $\delta'' \stackrel{\text{def}}{=} \max(\delta + \alpha\, \delta', \delta' + \alpha'\, \delta)$ and $\circ$ denotes relation composition.

For the proof, let μ_R and μ_S be witnesses for the liftings on the left-hand side of the implication. Then, the distribution μ with the following probability mass function is a witness for the lifting on the right-hand side:

$$\mu(a, c) = \sum_{b \in B \,|\, 0 < \mu_2(b)} \frac{\mu_R(a, b)\ \mu_S(b, c)}{\mu_2(b)}$$

We conclude this section with a result that shows the compatibility of the bind operator with (α, δ)-liftings. This result allows deriving the soundness of the rule for sequential composition presented in the next section. In the following, we say that a relation $R \subseteq A \times B$ is *full* iff for every $a \in A$ there exists $b \in B$ such that $a\ R\ b$, and symmetrically, for every $b \in B$ there exists $a \in A$ such that $a\ R\ b$.

LEMMA 4.13. *Let A, A', B and B' be discrete sets and let $R \subseteq A \times B$ and $R' \subseteq A' \times B'$. Then for any $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$, $M_1 : A \to \mathcal{D}(A')$ and $M_2 : B \to \mathcal{D}(B')$ that satisfy*

$$\mu_1 \sim_R^{\alpha,\delta} \mu_2 \quad \textit{and} \quad \forall a, b.\ a\, R\, b \implies (M_1\ a) \sim_{R'}^{\alpha',\delta'} (M_2\ b)$$

we have

$$(\mathsf{bind}\ \mu_1\ M_1) \sim_{R'}^{\alpha\alpha',\delta+\delta'} (\mathsf{bind}\ \mu_2\ M_2)$$

whenever R is full or $(M_1\ a)\ \mathbb{1}_{A'} = (M_2\ b)\ \mathbb{1}_{B'} = 1$ for every $a \in A$ and $b \in B$.

5. APPROXIMATE RELATIONAL HOARE LOGIC

This section introduces the central component of CertiPriv, namely an approximate probabilistic relational Hoare logic that is used to establish privacy guarantees of programs. We first present the programming language and its semantics. We then define relational judgments and show that they generalize differential privacy. Finally, we define a proof system for deriving valid judgments and an asymmetric variant of the logic.

5.1. Programming Language

CertiPriv supports reasoning about programs that are written in the typed, procedural, probabilistic imperative language pWHILE. Formally, the set of commands is defined inductively by the following clauses:

$$
\begin{array}{rcll}
\mathcal{I} & ::= & \mathcal{V} \leftarrow \mathcal{E} & \text{assignment} \\
& | & \mathcal{V} \stackrel{\$}{\leftarrow} \mathcal{DE} & \text{random sampling} \\
& | & \mathsf{if}\ \mathcal{E}\ \mathsf{then}\ \mathcal{C}\ \mathsf{else}\ \mathcal{C} & \text{conditional} \\
& | & \mathsf{while}\ \mathcal{E}\ \mathsf{do}\ \mathcal{C} & \text{while loop} \\
& | & \mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \dots, \mathcal{E}) & \text{procedure call} \\
& | & \mathsf{assert}\ \ \mathcal{E} & \text{runtime assertion} \\
\mathcal{C} & ::= & \mathsf{skip} & \text{nop} \\
& | & \mathcal{I};\ \mathcal{C} & \text{sequence}
\end{array}
$$

Here, $\mathcal{V}$ is a set of variable identifiers, $\mathcal{P}$ is a set of procedure names[1], $\mathcal{E}$ is a set of expressions, and $\mathcal{DE}$ is a set of distribution expressions. The base language includes expressions over Booleans, integers, lists, option and sum types, but can be extended by the user. The significant novelty of CertiPriv (compared to CertiCrypt), besides the addition of runtime assertions, is that distribution expressions may depend on the program state. This allows to express programs that sample from dynamically evolving probability distributions and will be required for some case studies in Section 6.

The semantics of programs is defined in two steps. First, we give an interpretation $[\![T]\!]$ to all object types T—these are types that are declared in CertiPriv programs—and we define the set $\mathcal{M}$ of memories as the set of mappings from variables to values. Then, the semantics of an expression e of type T, a distribution expression μ of type T, and a command c, respectively, are given by functions of the following types:

$$[\![e]\!] : \mathcal{M} \to [\![T]\!] \qquad [\![\mu]\!] : \mathcal{M} \to \mathcal{D}([\![T]\!]) \qquad [\![c]\!] : \mathcal{M} \to \mathcal{D}(\mathcal{M})$$

Informally, the semantics of an expression e maps a memory to a value in $[\![T]\!]$, the semantics of a distribution expression μ maps a memory to a distribution over $[\![T]\!]$, and the semantics of a program c maps an initial memory to a distribution over final memories. The semantics of programs complies with the expected equations; Figure 3 provides an excerpt. In the remainder, we only consider programs that sample values from discrete distributions, and so their output distributions are also discrete. Moreover, we say that a program c is *lossless* iff $[\![c]\!]\ m = 1$ for any initial memory m.

5.2. Validity and Privacy

apRHL is an approximate probabilistic relational Hoare logic that supports reasoning about differentially private computations. Judgments in apRHL are of the form

$$c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$$

[1]For the sake of readability, we omit procedure calls from most of the exposition; we keep them in the description of the language because we use them to describe the algorithm SMARTSUM in Fig. 9 and modularize its analysis.

$$
\begin{array}{ll}
[\![\mathsf{skip}]\!]\ m & = \mathsf{unit}\ m \\
[\![i;\ c]\!]\ m & = \mathsf{bind}\ ([\![i]\!]\ m)\ [\![c]\!] \\
[\![x \leftarrow e]\!]\ m & = \mathsf{unit}\ (m\,\{[\![e]\!]\ m/x\}) \\
[\![\mathsf{assert}\ e]\!]\ m & = \text{if } [\![e]\!]\ m = \mathsf{true} \text{ then } (\mathsf{unit}\ m) \text{ else } \mu_0 \\
[\![x \xleftarrow{\$} \mu]\!]\ m & = \mathsf{bind}\ ([\![\mu]\!]\ m)\ (\lambda v.\ \mathsf{unit}\ (m\,\{v/x\})) \\
[\![\mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2]\!]\ m & = \begin{cases} [\![c_1]\!]\ m & \text{if } [\![e]\!]\ m = \mathsf{true} \\ [\![c_2]\!]\ m & \text{if } [\![e]\!]\ m = \mathsf{false} \end{cases} \\
[\![\mathsf{while}\ e\ \mathsf{do}\ c]\!]\ m & = \lambda f.\sup\ (\lambda n.\ [\![[\mathsf{while}\ e\ \mathsf{do}\ c]_n]\!]\ m\ f)
\end{array}
$$

$$
\text{where} \quad \begin{array}{lcl} [\mathsf{while}\ e\ \mathsf{do}\ c]_0 & = & \mathsf{assert}\ \neg e \\ [\mathsf{while}\ e\ \mathsf{do}\ c]_{n+1} & = & \mathsf{if}\ e\ \mathsf{then}\ c;\ [\mathsf{while}\ e\ \mathsf{do}\ c]_n \end{array}
$$

Fig. 3. Semantics of pWHILE programs

where c_1 and c_2 are programs, assertions Ψ and Φ are relations over memories, $\alpha \in \mathbb{R}^{\geq 1}$ is called the skew, and $\delta \in [0, 1]$ is called the slack. In our formalization we use a shallow embedding for logical assertions, allowing us to inherit the expressiveness of the Coq language when writing pre- and post-conditions. In this article, we usually specify an assertion $m_1\ \Theta\ m_2$ as a formula over expressions tagged with either $\langle 1\rangle$ or $\langle 2\rangle$, to indicate whether they should be evaluated in m_1 or m_2, respectively. For instance, the assertion $e_1\langle 1\rangle < e_2\langle 2\rangle$ denotes the relation $\{(m_1, m_2) \mid [\![e_1]\!]\ m_1 < [\![e_2]\!]\ m_2\}$.

An apRHL judgment is valid if, for every pair of initial memories related by the pre-condition Ψ, the corresponding pair of output distributions is related by the (α, δ)-lifting of the post-condition Φ.

Definition 5.1 (*Validity in* apRHL). A judgment $c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$ is valid, written $\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$, iff

$$\forall m_1\ m_2.\ m_1\ \Psi\ m_2 \implies ([\![c_1]\!]\ m_1) \sim_{\Phi}^{\alpha,\delta} ([\![c_2]\!]\ m_2)$$

The following lemma is a direct consequence of the fundamental property of lifting (Theorem 4.10) applied to Definition 5.1. It shows that statements about programs derived using apRHL imply bounds on the α-distance of their output distributions.

LEMMA 5.2. *If* $\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$*, then for all memories* m_1, m_2 *and unit-valued functions* $f_1, f_2 : \mathcal{M} \to [0, 1]$,

$$m_1\ \Psi\ m_2 \wedge f_1 =_{\Phi} f_2 \implies \Delta_\alpha([\![c_1]\!]\ m_1\ f_1, [\![c_2]\!]\ m_2\ f_2) \leq \delta$$

The statement of Lemma 5.2 can be specialized to a statement about the differential privacy of programs.

COROLLARY 5.3. *Let* d *be a metric on* $\mathcal{M}$ *and* Ψ *an assertion expressing that* $d(m_1, m_2) \leq 1$. *If* $\models c \sim_{\exp(\epsilon),\delta} c : \Psi \Rightarrow \equiv$*, then* c *satisfies* (ϵ, δ)*-differential privacy.*

Corollary 5.3 is the central result for deriving differential privacy guarantees in apRHL. Using Theorem 4.11, one can prove the converse to Corollary 5.3, yielding a characterization of approximate differential privacy.

The logic apRHL can also be used to reason about more traditional information-flow properties, such as probabilistic non-interference. To see this, let Ψ be an arbitrary equivalence relation on initial states and let $\equiv$ be the identity relation on final states. A judgment $\models c \sim_{1,0} c : \Psi \Rightarrow \equiv$ entails that two initial states induce the same distribution of final states whenever they are related by Ψ. In particular, this implies that an adversary who can observe (or even repeatedly sample) the output of c will only be able to determine the initial state up to its Ψ-equivalence class. In this way, Ψ can be used for expressing fine-grained notions of confidentiality, including probabilistic noninterference [Sabelfeld and Sands 2000]. Our interpretation of apRHL judgments generalizes to arbitrary equivalence relations as post-conditions. In this way, one can capture adversaries that have only partial views on the system, as required for distributed differential privacy [Beimel et al. 2008].

We finally show how apRHL can also be used for deriving continuity properties of probabilistic programs. We begin by recalling the notion of Lipschitz continuity. Given two sets A and B equipped with metrics Δ_A and Δ_B, we say that a function $f : A \to B$ is *Lipschitz continuous* iff there exists a constant $K \geq 0$ such that $\Delta_B(f(a_1), f(a_2)) \leq K\Delta_A(a_1, a_2)$ for all $a_1, a_2 \in A$. To study the Lipschitz continuity of programs we define a uniform semantic function for programs, mapping distributions to distributions: $[\![c]\!]^\star\ \mu \stackrel{\text{def}}{=} \mathsf{bind}\ \mu\ [\![c]\!]$, and we use α-distance as a metric for both inputs and outputs. The following is a consequence of Theorems 4.10 and 4.11, and Lemma 4.13.

LEMMA 5.4. *If* $\models c_1 \sim_{\alpha,\delta} c_2 : \equiv \Rightarrow \equiv$*, then*

$$\Delta_{\alpha'}(\mu_1, \mu_2) \leq \delta' \implies \Delta_{\alpha\alpha'}([\![c_1]\!]^\star\ \mu_1, [\![c_2]\!]^\star\ \mu_2) \leq \delta + \delta'$$

Taking $c_1 = c_2 = c$ and $\delta = 0$, the conclusion of the above lemma is equivalent to saying that $[\![c]\!]^\star$ is a *metric map*, i.e., that it is continuous with Lipschitz constant $K = 1$.

5.3. Logic

This section introduces a set of proof rules to support reasoning about the validity of apRHL judgments. In order to maximize flexibility and to allow the application of proof rules to be interleaved with other forms of reasoning, the soundness of each proof rule is proved individually as a Coq lemma. Nevertheless, we retain the usual presentation of the rules as a proof system.

We present the core apRHL rules in Figure 4; all rules generalize their counterparts in pRHL [Barthe et al. 2009], which can be recovered by setting $\alpha = 1$ and $\delta = 0$. (Any valid pRHL derivation admits an immediate translation into apRHL.) We begin by describing the rules corresponding to language constructs.

The [skip], [assert] and [assn] rules are direct transpositions of the corresponding pRHL rules. Rule [rand] states that for any two distribution expressions μ_1 and μ_2 of type A, the random assignments $x_1 \stackrel{\$}{\leftarrow} \mu_1$ and $x_2 \stackrel{\$}{\leftarrow} \mu_2$ are (α, δ)-related w.r.t. pre-condition Ψ and post-condition $x_1\langle 1\rangle = x_2\langle 2\rangle$, provided the α-distance between the distributions $[\![\mu_1]\!]\ m_1$ and $[\![\mu_2]\!]\ m_2$ is smaller than δ for any m_1 and m_2 related by Ψ.

Rule [seq] encodes the sequential composition theorem of approximate differential privacy, further elaborated in § 5.5. Since its soundness follows from Lemma 4.13, it requires that either relation Φ' is full, or that both c_1' and c_2' are lossless. If c_1' and c_2' contain no runtime assertions, this requirement can be easily discharged. Indeed, a key property of the proof system of Figures 4 and 5 is that for assertion-free programs c_1, c_2 that sample only from proper probability distributions, if a judgment of the form $\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$ is derivable, then c_1 and c_2 are lossless.

Rule [cond] states that branching statements are (α, δ)-related w.r.t. pre-condition Ψ and post-condition Φ, provided that the pre-condition Ψ ensures that the guards of both statements are equivalent, and that the true and false branches are (α, δ)-related w.r.t. pre-conditions $\Psi \wedge b\langle 1\rangle$ and $\Psi \wedge \neg b\langle 1\rangle$, respectively.

Rule [case] allows one to reason by case analysis on the pre-condition of a judgment. The weakening rule [weak] generalizes the rule of consequence of (relational) Hoare logic by allowing to increase the skew and slack. The composition rule [comp] permits

$$\frac{\forall m_1\ m_2.\ m_1\ \Psi\ m_2 \implies (m_1\{\llbracket e_1\rrbracket\ m_1/x_1\})\ \Phi\ (m_2\{\llbracket e_2\rrbracket\ m_2/x_2\})}{\models x_1 \leftarrow e_1 \sim_{1,0} x_2 \leftarrow e_2 : \Psi \Rightarrow \Phi}[\mathsf{assn}]$$

$$\frac{\forall m_1\ m_2.\ m_1\ \Psi\ m_2 \implies \Delta_\alpha(\llbracket \mu_1\rrbracket\ m_1, \llbracket \mu_2\rrbracket\ m_2) \leq \delta}{\models x_1 \xleftarrow{\$} \mu_1 \sim_{\alpha,\delta} x_2 \xleftarrow{\$} \mu_2 : \Psi \Rightarrow x_1\langle 1\rangle = x_2\langle 2\rangle}[\mathsf{rand}]$$

$$\frac{}{\models \mathsf{skip} \sim_{1,0} \mathsf{skip} : \Psi \Rightarrow \Psi}[\mathsf{skip}] \qquad \frac{\Psi \implies b\langle 1\rangle \equiv b'\langle 2\rangle}{\models \mathsf{assert}\ b \sim_{1,0} \mathsf{assert}\ b' : \Psi \Rightarrow \Psi \wedge b\langle 1\rangle}[\mathsf{assert}]$$

$$\frac{\models c_1 \sim_{\alpha,\delta} c_1' : \Psi \wedge b\langle 1\rangle \Rightarrow \Phi \qquad \models c_2 \sim_{\alpha,\delta} c_2' : \Psi \wedge \neg b\langle 1\rangle \Rightarrow \Phi \qquad \Psi \implies b\langle 1\rangle \equiv b'\langle 2\rangle}{\models \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \sim_{\alpha,\delta} \mathsf{if}\ b'\ \mathsf{then}\ c_1'\ \mathsf{else}\ c_2' : \Psi \Rightarrow \Phi}[\mathsf{cond}]$$

$$\frac{\begin{array}{c}\models c_1 \sim_{\alpha,\delta} c_2 : \Theta \wedge b_1\langle 1\rangle \wedge k = e \Rightarrow \Theta \wedge k < e \\ \Theta \wedge n \leq e \implies \neg b\langle 1\rangle \qquad \Theta \implies b_1\langle 1\rangle = b_2\langle 2\rangle\end{array}}{\models \mathsf{while}\ b_1\ \mathsf{do}\ c_1 \sim_{\alpha^n,n\delta} \mathsf{while}\ b_2\ \mathsf{do}\ c_2 : \Theta \wedge 0 \leq e \Rightarrow \Theta \wedge \neg b_1\langle 1\rangle}[\mathsf{while}]$$

$$\frac{\Phi' \text{ is full } \vee\ c_1', c_2' \text{ are lossless} \qquad \models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi' \qquad \models c_1' \sim_{\alpha',\delta'} c_2' : \Phi' \Rightarrow \Phi}{\models c_1; c_1' \sim_{\alpha\alpha',\delta+\delta'} c_2; c_2' : \Psi \Rightarrow \Phi}[\mathsf{seq}]$$

$$\frac{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \wedge \Theta \Rightarrow \Phi \qquad \models c_1 \sim_{\alpha,\delta} c_2 : \Psi \wedge \neg\Theta \Rightarrow \Phi}{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi}[\mathsf{case}]$$

$$\frac{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi \qquad \models c_2 \sim_{\alpha',\delta'} c_3 : \Psi' \Rightarrow \Phi'}{\models c_1 \sim_{\alpha\alpha',\max(\delta+\alpha\ \delta',\delta'+\alpha'\ \delta)} c_3 : \Psi \circ \Psi' \Rightarrow \Phi \circ \Phi'}[\mathsf{comp}]$$

$$\frac{\begin{array}{l}\models c_1 \sim_{\alpha',\delta'} c_2 : \Psi' \Rightarrow \Phi' \\ \Psi \Rightarrow \Psi' \quad \Phi' \Rightarrow \Phi \quad \alpha' \leq \alpha \quad \delta' \leq \delta\end{array}}{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi}[\mathsf{weak}] \qquad \frac{\models c_2 \sim_{\alpha,\delta} c_1 : \Psi^{-1} \Rightarrow \Phi^{-1}}{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi}[\mathsf{transp}]$$

$$\frac{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi \qquad \forall m_1\ m_2.\ m_1\ \Theta\ m_2 \implies \mathsf{range}\ \ \Theta\ (\llbracket c_1\rrbracket\ m_1 \times \llbracket c_2\rrbracket\ m_2)}{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \wedge \Theta \Rightarrow \Phi \wedge \Theta}[\mathsf{frame}]$$

Fig. 4. Core proof rules of the approximate relational Hoare logic

structuring proofs by introducing intermediate programs (as in the game-playing technique for cryptographic proofs [Barthe et al. 2009]). It yields a rule for the case when Ψ and Φ are partial equivalence relations which, specialized to $\alpha = \alpha' = 1$, reads:

$$\frac{\models c_1 \sim_{1,\delta} c_2 : \Psi \Rightarrow \Phi \qquad \models c_2 \sim_{1,\delta'} c_3 : \Psi \Rightarrow \Phi}{\models c_1 \sim_{1,\delta+\delta'} c_3 : \Psi \Rightarrow \Phi}$$

Rule [frame] rule allows one to strengthen the pre- and post-condition with an assertion Θ whose validity is preserved by executing the commands in the judgment. (In the figure, the notation $\times$ is used to denote the product of two distributions.)

Finally, rule [while] can be used to relate two loops that execute in lockstep and terminate after at most n iterations. The loop invariant Θ ensures that the loops progress in lockstep; to guarantee that both loops terminate within n iterations, the rule re-

$$\frac{\begin{array}{l}\Theta \implies b_1\langle 1\rangle \equiv b_2\langle 2\rangle \wedge P\langle 1\rangle \equiv P\langle 2\rangle \wedge i\langle 1\rangle = i\langle 2\rangle \qquad \Theta \wedge n \leq i\langle 1\rangle \implies \neg b_1\langle 1\rangle \\ \models c_1;\ \mathsf{assert}\ \neg P \sim_{\alpha_1(j),0} c_2;\ \mathsf{assert}\ \neg P : \Theta \wedge (b_1 \wedge i = j \wedge \neg P)\langle 1\rangle \Rightarrow \Theta \wedge i\langle 1\rangle = j{+}1 \\ \models c_1;\ \mathsf{assert}\ P \sim_{\alpha_2,0} c_2;\ \mathsf{assert}\ P : \Theta \wedge (b_1 \wedge i = j \wedge \neg P)\langle 1\rangle \Rightarrow \Theta \wedge i\langle 1\rangle = j{+}1 \\ \models c_1 \sim_{1,0} c_2 : \Theta \wedge (b_1 \wedge i = j \wedge P)\langle 1\rangle \Rightarrow \Theta \wedge (i = j{+}1 \wedge P)\langle 1\rangle\end{array}}{\models \mathsf{while}\ b_1\ \mathsf{do}\ c_1 \sim_{\alpha_2 \prod_{i=0}^{n-1} \alpha_1(i),0} \mathsf{while}\ b_2\ \mathsf{do}\ c_2 : \Theta \wedge i\langle 1\rangle = 0 \Rightarrow \Theta \wedge \neg b_1\langle 1\rangle}\ [\mathsf{gwhile}]$$

Fig. 5. Generalized rule for loops

quires exhibiting a strictly increasing loop variant e. Rule [while] essentially states that the loops are $(n \ln(\alpha), n\delta)$-differentially private when each iteration is $(\ln(\alpha), \delta)$-differentially private. This rule is sufficient for programs like the k-Median algorithm studied in Section 6.3, where the skew factor α and the slack δ are the same for every iteration. Other programs, such as the Minimum Vertex Cover algorithm studied in Section 2, require applying more sophisticated rules in which the skew and the slack may vary across iterations. For instance, the rule [gwhile] shown in Figure 5 allows for a finer-grained case analysis depending on a predicate P whose validity is preserved across iterations. Assume that when P does not hold, the j-th iteration of each loop can be related with skew $\alpha_1(j)$ when P does not hold after their execution, and with skew α_2 when it does. Furthermore, assume that once P holds, the remaining iterations are observationally equivalent. Then, the two loops are related with skew $\alpha_2 \prod_{i=0}^{n-1} \alpha_1(i)$. Intuitively, as long as P does not hold, the j-th iteration is $\ln(\alpha_1(j))$-differentially private, while the single iteration where the validity of P may be established (this occurs necessarily at the same time in both executions) incurs an $\ln(\alpha_2)$ privacy penalty; the remaining iterations preserve P and do not add to the privacy bound.

The proofs of soundness of apRHL rules in Coq rely on properties of approximate lifting. For instance, the soundness of rules [weak] and [comp] follows directly from Lemma 4.9 and Theorem 4.12, respectively. To illustrate the kind of reasoning such proofs involve, we sketch the proof of soundness of [rand]. To establish the validity of judgment

$$x_1 \xleftarrow{\$} \mu_1 \sim_{\alpha,\delta} x_2 \xleftarrow{\$} \mu_2 : \Psi \Rightarrow \Phi$$

where $\Phi \stackrel{\text{def}}{=} x_1\langle 1\rangle = x_2\langle 2\rangle$, we have to show that for every pair of Ψ-related memories m_1 and m_2,

$$(\mathsf{bind}\ (\llbracket \mu_1 \rrbracket\ m_1)\ (\lambda v.\ \mathsf{unit}\ (m_1\ \{v/x_1\}))) \sim_{\Phi}^{\alpha,\delta} (\mathsf{bind}\ (\llbracket \mu_1 \rrbracket\ m_1)\ (\lambda v.\ \mathsf{unit}\ (m_2\ \{v/x_2\})))$$

We prove this by applying Lemma 4.13 with $(\alpha', \delta') = (1, 0)$, and R the identity relation. The hypotheses of the lemma simplify to

$$(\llbracket \mu_1 \rrbracket\ m_1) \sim_{\equiv}^{\alpha,\delta} (\llbracket \mu_2 \rrbracket\ m_2) \qquad \text{and} \qquad \mathsf{unit}\ (m_1\ \{v/x_1\}) \sim_{\Phi}^{1,0} \mathsf{unit}\ (m_2\ \{v/x_2\})$$

The first follows from Theorem 4.11 and the premise of the rule, whereas the second follows from Lemma A.7.

5.4. An Asymmetric Variant of apRHL

The judgments of apRHL can be used to relate the output distributions of programs. More precisely, if $\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$, Lemma 5.2 entails inequalities

$$\llbracket c_1 \rrbracket\ m_1\ f_1 \leq \alpha\ (\llbracket c_2 \rrbracket\ m_2\ f_2) + \delta \qquad \text{and} \qquad \llbracket c_2 \rrbracket\ m_2\ f_2 \leq \alpha\ (\llbracket c_1 \rrbracket\ m_1\ f_1) + \delta$$

for every pair of Ψ-related memories $m_1, m_2 \in \mathcal{M}$ and every pair of Φ-equivalent functions $f_1, f_2 : \mathcal{M} \to [0, 1]$. It is sometimes convenient to reason independently about each of the above inequalities: in this way one can choose different values of the parameters α and δ in the left and right formula, which can lead to stronger privacy guarantees.

We next introduce apRHL*, an asymmetric variant of apRHL that allows to conclude only one of the above inequalities, and thus allows an independent and finer-grained choice of the skew α and the slack δ. apRHL* judgments have the same form

$$c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$$

as the original version of the logic and their validity is defined in a similar way by considering asymmetric versions of the α-distance and (α, δ)-lifting presented in Section 4. Most rules in apRHL remain valid in apRHL* (rule [transp] constitutes the only exception).

We next give formal definitions of the asymmetric counterparts of the notions studied in Sections 4.1 and 4.3 and briefly discuss how their properties translate to the asymmetric setting. We present only the *left* variant of the logic, the *right* variant is analogous. We first define an asymmetric variant of α-distance:

$$\Delta^\star_\alpha(\mu_1, \mu_2) \stackrel{\text{def}}{=} \max_{f:A\to[0,1]} \Delta^\star_\alpha(\mu_1\ f, \mu_2\ f)$$

where $\Delta^\star_\alpha(a, b) = \max\{a - \alpha b, 0\}$. Given $\alpha \in \mathbb{R}^{\geq 1}$, $\delta \in [0, 1]$ and $R \subseteq A \times B$, we define the asymmetric lifting of R as the relation $\overset{\star}{\sim}{}_R^{\alpha,\delta}$ such that $\mu_1 \overset{\star}{\sim}{}_R^{\alpha,\delta} \mu_2$ iff there exists $\mu \in \mathcal{D}(A \times B)$ satisfying

(1) $\mathsf{range}\ R\ \mu$,
(2) $\pi_1\ \mu \leq \mu_1 \ \wedge\ \pi_2\ \mu \leq \mu_2$, and
(3) $\Delta^\star_\alpha(\pi_1\ \mu, \mu_1) \leq \delta$

The distance $\Delta^\star_\alpha(\cdot, \cdot)$ enjoys all properties of Lemma 4.3, except symmetry; the generalized triangle inequality (4) can be strengthened to

$$\Delta^\star_{\alpha\alpha'}(\mu_1, \mu_3) \leq \Delta^\star_\alpha(\mu_1, \mu_2) + \alpha\ \Delta^\star_{\alpha'}(\mu_2, \mu_3)$$

Lemma A.3 can be reformulated as $\Delta^\star_\alpha(\mu_1, \mu_2) = \mu_1(A_0) - \alpha\mu_2(A_0)$ where μ_1 and μ_2 are discrete distributions over A and $A_0 = \{a \in A \mid \mu_1(a) \geq \alpha\mu_2(a)\}$. This relates both variants of α-distance by $\Delta_\alpha(\mu_1, \mu_2) = \max\{\Delta^\star_\alpha(\mu_1, \mu_2), \Delta^\star_\alpha(\mu_2, \mu_1)\}$. Finally, for every pair of distributions μ_1 and μ_2 over a discrete set A one can upper-bound $\Delta^\star_\alpha(\mu_1, \mu_2)$ by $\sum_{a\in A} \Delta^\star_\alpha(\mu_1(a), \mu_2(a))$ as Lemma 4.6 does for standard α-distance.

The new notion of lifting satisfies both the monotonicity condition of Lemma 4.9 and an analogue of Theorem 4.11. Theorem 4.12 can also be strengthened in accordance with the triangle inequality condition of $\Delta^\star_\alpha$ to yield

$$\mu_1 \overset{\star}{\sim}{}_R^{\alpha,\delta} \mu_2 \wedge \mu_2 \overset{\star}{\sim}{}_S^{\alpha',\delta'} \mu_3 \implies \mu_1 \overset{\star}{\sim}{}_{R\circ S}^{\alpha\alpha',\delta+\alpha\delta'} \mu_3$$

The fundamental property of lifting can also be transposed to the asymmetric setting. Given $f : A \to [0, 1]$, $g : B \to [0, 1]$ and $R \subseteq A \times B$, we say that f is *R-dominated* by g, and write it $f \leq_R g$, iff for every $a \in A$ and $b \in B$, $a\ R\ b$ implies $f\ a \leq g\ b$. Theorem 4.10 is reformulated as follows:

$$\mu_1 \overset{\star}{\sim}{}_R^{\alpha,\delta} \mu_2 \wedge f_1 \leq_R f_2 \implies \Delta^\star_\alpha(\mu_1\ f_1, \mu_2\ f_2) \leq \delta$$

We next define validity in apRHL* and show how the asymmetric logic can be used to relate the distributions generated by probabilistic programs.

Definition 5.5 (*Validity in* apRHL*). We say that a judgment $c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$ is *valid* in apRHL*, written $\models^\star c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$, iff

$$\forall m_1\ m_2.\ m_1\ \Psi\ m_2 \implies (\llbracket c_1 \rrbracket\ m_1) \overset{\star}{\sim}{}_\Phi^{\alpha,\delta} (\llbracket c_2 \rrbracket\ m_2)$$

LEMMA 5.6. *If* $\models^{\star} c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$*, then for all memories* m_1, m_2 *and unit-valued functions* $f_1, f_2 : \mathcal{M} \to [0,1]$*,*

$$m_1 \ \Psi \ m_2 \wedge f_1 \leq_{\Phi} f_2 \implies \Delta^{\star}_{\alpha}([\![c_1]\!] \ m_1 \ f_1, [\![c_2]\!] \ m_2 \ f_2) \leq \delta$$

It is not hard to see that Corollary 5.3 and its converse remain valid if the validity of judgment $c \sim_{\exp(\epsilon),\delta} c : \Psi \Rightarrow \equiv$ is taken in apRHL* instead of apRHL. (This is true for any symmetric precondition Ψ). Therefore, approximate differential privacy can also be cast in terms of apRHL*. We immediately obtain a proof system for reasoning about the validity of apRHL* judgments. Except for [transp], all apRHL rules in Figures 4 and 5 can be transposed to apRHL*. For consistency, we keep the names of the original rules and decorate them with a $\star$. E.g., the rule for random assignments reads

$$\frac{\forall m_1 \ m_2 .\ m_1 \ \Psi \ m_2 \implies \Delta^{\star}_{\alpha}([\![\mu_1]\!] \ m_1, [\![\mu_2]\!] \ m_2) \leq \delta}{\models^{\star} x_1 \xleftarrow{\$} \mu_1 \sim_{\alpha,\delta} x_2 \xleftarrow{\$} \mu_2 : \Psi \Rightarrow x_1\langle 1\rangle = x_2\langle 2\rangle}[\mathsf{rand}^{*}]$$

and rule [comp] can be strengthened to

$$\frac{\models^{\star} c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi \qquad \models^{\star} c_2 \sim_{\alpha',\delta'} c_3 : \Psi' \Rightarrow \Phi'}{\models^{\star} c_1 \sim_{\alpha\alpha',\delta+\alpha\ \delta'} c_3 : \Psi \circ \Psi' \Rightarrow \Phi \circ \Phi'}[\mathsf{comp}^{\star}]$$

In Section 6.4 we demonstrate the benefits of apRHL* over apRHL. Concretely, we show how apRHL* can be used to prove a differential privacy bound for an approximation algorithm for the Minimum Vertex Cover problem that improves over the bound that can be proved using apRHL.

5.5. Sequential and Parallel Composition Theorems

Composition theorems play an important role in the construction and analysis of differentially private mechanisms. There are two main forms of composition, namely sequential and parallel. We briefly explain each of them, and establish their connections with reasoning principles in apRHL.

The sequential composition theorem states that the composition of an (ϵ,δ)-differentially private computation with an (ϵ',δ')-differentially private computation yields an $(\epsilon+\epsilon',\delta+\delta')$-differentially private computation [Dwork et al. 2006a; McSherry 2009]. The apRHL rule for sequential composition [seq] provides a counterpart to this first theorem. One can curb the linear growth in ϵ by shifting some of the privacy loss to δ [Dwork et al. 2010], a result which is established using an information-theoretic analogue of the dense model theorem. Proving the soundness of this alternative bound is a significant challenge, which we leave for future work.

The parallel composition theorem states that the composition of an (ϵ,δ)-differentially private computation with another (ϵ',δ')-differentially private computation that operates on a disjoint part of the dataset yields a $(\max(\epsilon,\epsilon'),\max(\delta,\delta'))$-differentially private computation [McSherry 2009]. This theorem has a natural counterpart in apRHL. To make this claim precise, we introduce the parallel composition of two commands, as a construct taking two commands that operate on disjoint parts of the memory. Formally, the construction $c\ {}_X\|_Y\ c'$ is only well defined when X and Y are disjoint sets of variables, with c reading and writing variables from X, and c' reading and writing variables from Y. The semantics of $c\ {}_X\|_Y\ c'$ coincides with the semantics of $c; c'$:

$$[\![c\ {}_X\|_Y\ c']\!] \stackrel{\text{def}}{=} [\![c; c']\!]$$

Now assume that $c\ {}_X\|_Y\ c'$ is well-defined. Let Ψ and Ψ' be relational formulae that depend only on variables in X and Y, respectively. We establish the following rule [par]

$$\frac{\models c \sim_{\alpha,\delta} c : \Psi \Rightarrow \equiv \qquad \models c' \sim_{\alpha',\delta'} c' : \Psi' \Rightarrow \equiv}{\models c\ {}_X\|_Y\ c' \sim_{\max(\alpha,\alpha'),\max(\delta,\delta')} c\ {}_X\|_Y\ c' : \Psi \vee \Psi' \Rightarrow \equiv}\,[\mathsf{par}]$$

whose proof follows from the observation that for every command c_0

$$\models c_0 \sim_{1,0} c_0 : \equiv \Rightarrow \equiv$$

and uses the sequential composition rule to derive

$$\models c\ {}_X\|_Y\ c' \sim_{\alpha,\delta} c\ {}_X\|_Y\ c' : \Psi \Rightarrow \equiv \qquad \models c\ {}_X\|_Y\ c' \sim_{\alpha',\delta'} c\ {}_X\|_Y\ c' : \Psi' \Rightarrow \equiv$$

The validity of [par] then follows from the rules of weakening and case analysis.

To see why [par] captures parallel composition of computations as described above, instantiate Ψ to express that memories coincide on variables in X and differ in the value of at most one variable in Y. Symmetrically, instantiate Ψ' to express that memories coincide on Y and differ in at most one variable in X. The disjunction $\Psi \vee \Psi'$ captures the fact that the initial memories differ in the value of at most one variable in $X \cup Y$, i.e. that they are adjacent in the sense of the standard definition of differential privacy.

6. CASE STUDIES

We illustrate the versatility of our framework by proving from first principles the correctness of the Laplacian, Gaussian and Exponential mechanisms. We then apply these mechanisms to prove differential privacy for an algorithm solving the k-Median problem, several streaming algorithms, and an approximation algorithm for the Minimum Vertex Cover problem.

6.1. Laplacian, Gaussian and Exponential Mechanisms

Many algorithms for computing statistics and data mining are numeric, meaning that they return (approximations of) real numbers. The Laplacian and Gaussian mechanisms of Dwork et al. [Dwork et al. 2006b; Dwork et al. 2006a] are fundamental tools for making such computations differentially private. This is achieved by perturbing the algorithm's true output with symmetric noise calibrated according to its sensitivity.

In the reminder, we use $\mathcal{L}(r,\sigma)$ and $\mathcal{N}(r,\sigma)$ to denote, respectively, the Laplace and Gaussian distribution with mean r and scale factor σ. Their density functions at x satisfy

$$\mathcal{L}(r,\sigma)(x) \propto \exp\left(-\frac{|x-r|}{\sigma}\right) \qquad \text{and} \qquad \mathcal{N}(r,\sigma)(x) \propto \exp\left(-\frac{|x-r|^2}{\sigma}\right)$$

To transform a deterministic computation $f \colon A \to \mathbb{R}$ into a differentially private computation, one needs to set r to the true output of the computation and choose σ (i.e. the amount of noise) according to the *sensitivity* of f. Informally, the sensitivity of f measures how far apart it maps nearby inputs. Formally, the sensitivity $\mathbf{S}_f$ is defined relative to a metric d on A as

$$\mathbf{S}_f \stackrel{\text{def}}{=} \max_{a,a' \mid d(a,a') \leq 1} |f\ a - f\ a'|$$

The justification for the Laplacian mechanism is a result that states that for a function $f : A \to \mathbb{R}$, the randomized algorithm that on input a returns a value sampled from distribution $\mathcal{L}(f(a), \mathbf{S}_f/\epsilon)$ is ϵ-differentially private [Dwork et al. 2006b].

While the Laplacian mechanism transforms numerical algorithms into computations that satisfy standard differential privacy, the Gaussian mechanism achieves only approximate differential privacy. The randomized algorithm that on input a returns a value drawn from distribution $\mathcal{N}(f(a), \sigma)$ is (ϵ, δ)-differentially private provided σ is chosen so that the tail of $\mathcal{N}(0, \sigma)$ satisfies a particular bound involving ϵ and δ. We elaborate on such constrain later.

One limitation of the Laplacian and Gaussian mechanisms is that they are confined to numerical algorithms. The Exponential mechanism [McSherry and Talwar 2007] is a general mechanism for building differentially private algorithms with arbitrary (but discrete) output domains. The Exponential mechanism takes as parameters a base distribution μ on a set B, and a scoring function $s : A \times B \to \mathbb{R}^{\geq 0}$; intuitively, values b maximizing $s(a, b)$ are the most appealing output for an input a. The Exponential mechanism is a randomized algorithm that takes a value $a \in A$ and returns a value $b \in B$ that approximately maximizes the score $s(a, b)$, where the quality of the approximation is determined by a parameter $\epsilon > 0$. Formally, the Exponential mechanism $\mathcal{E}^{\epsilon}_{s,\mu}$ maps every element in A to a distribution in B whose probability mass at b is:

$$\mathcal{E}^{\epsilon}_{s,\mu}(a)\ b = \frac{\exp(\epsilon\ s(a, b))\ (\mu\ b)}{\displaystyle\sum_{b' \in B} \exp(\epsilon\ s(a, b'))\ (\mu\ b')}$$

The definition implicitly assumes that the sum in the denominator is bounded for all $a \in A$. McSherry and Talwar [McSherry and Talwar 2007] show that $\mathcal{E}^{\epsilon}_{s,\mu}$ is $2\epsilon\mathbf{S}_s$-differentially private, where $\mathbf{S}_s$ is the maximum sensitivity of s w.r.t. a, for all b.

We define the three mechanisms we consider as instances of a general construction $(\cdot)^{\sharp}$ that takes as input a function $f : A \to B \to \mathbb{R}^{\geq 0}$ and returns another function $f^{\sharp} : A \to \mathcal{D}(B)$ such that for every $a \in A$ the probability mass of $f^{\sharp}\ a$ at b is given by:

$$f^{\sharp}\ a\ b = \frac{f\ a\ b}{\displaystyle\sum_{b' \in B} f\ a\ b'}$$

Using this construction, the Exponential mechanism for a scoring function s, base distribution μ and scale factor ϵ is defined as

$$\mathcal{E}^{\epsilon}_{s,\mu} \stackrel{\text{def}}{=} (\lambda a\ b.\ \exp(\epsilon\ s(a, b))\ (\mu\ b))^{\sharp}$$

whereas the Laplacian and Gaussian mechanisms with mean value r and scale factor σ are defined, respectively, as

$$\mathcal{L}(r, \sigma) \stackrel{\text{def}}{=} \left(\lambda a\ b.\ \exp\left(-\frac{|b - a|}{\sigma}\right)\right)^{\sharp} r \qquad \mathcal{N}(r, \sigma) \stackrel{\text{def}}{=} \left(\lambda a\ b.\ \exp\left(-\frac{|b - a|^2}{\sigma}\right)\right)^{\sharp} r$$

Rigorously speaking, we consider discrete versions of the Laplacian and Gaussian mechanisms over integers. (When instantiating the operator $(\cdot)^{\sharp}$ in the definition of both mechanisms, we take $A = B = \mathbb{Z}$.)

We derive the correctness of Gaussian mechanism as a consequence of the following lemma.

LEMMA 6.1. *Let B be a discrete set and consider $f : A \to B \to \mathbb{R}^{\geq 0}$ such that $f^{\sharp}$ is well defined. Moreover, let a_1, a_2 in A and $\alpha \geq 1$ be such that $\sum_{b \in B} f\ a_1\ b \leq \alpha \sum_{b \in B} f\ a_2\ b$ and $\sum_{b \in B} f\ a_2\ b \leq \alpha \sum_{b \in B} f\ a_1\ b$. Then for every $\alpha' \geq 1$,*

$$\Delta_{\alpha\alpha'}(f^{\sharp}\ a_1, f^{\sharp}\ a_2) \leq \max\{f^{\sharp}\ a_1\ \mathbb{1}_{S_1}, f^{\sharp}\ a_2\ \mathbb{1}_{S_2}\}$$

where $S_1 = \{b \in B \mid f\ a_1\ b > \alpha'\ f\ a_2\ b\}$ and $S_2 = \{b \in B \mid f\ a_2\ b > \alpha'\ f\ a_1\ b\}$.

$$\frac{m_1\ \Psi\ m_2 \implies |[\![r]\!]\ m_1 - [\![r]\!]\ m_2| \leq k \qquad \exp(\epsilon) \leq \alpha}{\models x \xleftarrow{\$} \mathcal{L}(r, \frac{k}{\epsilon}) \sim_{\alpha,0} y \xleftarrow{\$} \mathcal{L}(r, \frac{k}{\epsilon}) : \Psi \Rightarrow x\langle 1\rangle = y\langle 2\rangle}[\mathsf{lap}]$$

$$\frac{m_1\ \Psi\ m_2 \implies |[\![r]\!]\ m_1 - [\![r]\!]\ m_2| \leq k \qquad \exp(\epsilon) \leq \alpha \qquad \mathcal{B}\left(\sigma, \frac{\sigma\epsilon - k^2}{2k}\right) \leq \delta}{\models x \xleftarrow{\$} \mathcal{N}(r, \sigma) \sim_{\alpha,\delta} y \xleftarrow{\$} \mathcal{N}(r, \sigma) : \Psi \Rightarrow x\langle 1\rangle = y\langle 2\rangle}[\mathsf{norm}]$$

$$\frac{m_1\ \Psi\ m_2 \implies d([\![a]\!]\ m_1, [\![a]\!]\ m_2) \leq k \qquad \exp(2k\mathbf{S}_s\epsilon) \leq \alpha}{\models x \xleftarrow{\$} \mathcal{E}^{\epsilon}_{s,\mu}(a) \sim_{\alpha,0} y \xleftarrow{\$} \mathcal{E}^{\epsilon}_{s,\mu}(a) : \Psi \Rightarrow x\langle 1\rangle = y\langle 2\rangle}[\mathsf{exp}]$$

Fig. 6. Rules for the Laplacian, Gaussian and Exponential mechanisms

The correctness of the Laplacian and Exponential mechanisms is derived from the following corollary.

COROLLARY 6.2. *Let B be a discrete set and consider $f : A \to B \to \mathbb{R}^{\geq 0}$ such that $f^\sharp$ is well defined. Moreover, let a_1, a_2 in A and $\alpha \geq 1$ be such that for all b, $f\ a_1\ b \leq \alpha\ f\ a_2\ b$ and $f\ a_2\ b \leq \alpha\ f\ a_1 b$. Then,*

$$\Delta_{\alpha^2}(f^\sharp\ a_1, f^\sharp\ a_2) = 0$$

If moreover $\sum_{b\in B} f\ a\ b = \sum_{b \in B} f\ a'\ b$, then

$$\Delta_{\alpha}(f^\sharp\ a_1, f^\sharp\ a_2) = 0$$

The privacy guarantees for the Laplacian, Gaussian and Exponential mechanisms are stated as rules [lap], [norm] and [exp] in Figure 6. The premise of rule [lap] requires to prove that the values around which the mechanism is centered are within distance k. This is the case when these values are computed by a k-sensitive function starting from adjacent inputs, which corresponds to the usual interpretation of the guarantees provided by the Laplacian mechanism [Dwork et al. 2006b]. In the premise of rule [norm], $\mathcal{B}(\sigma, x)$ denotes the probability that the normal distribution $\mathcal{N}(0, \sigma)$ takes values greater than x. The rule can be simplified by considering particular (upper) bounds of $\mathcal{B}(\sigma, x)$. For instance, the Gaussian mechanism of [Dwork et al. 2006a] is recovered from rule [norm] by adopting the bound $\mathcal{B}(\sigma, x) \leq \frac{\sigma e^{-x^2/\sigma}}{2x\sqrt{\pi}}$, while that of [Nikolov et al. 2012] by considering a Chernoff bound. For the sake of generality, we present rule [norm] in a generic way and assume no particular bound for $\mathcal{B}(\sigma, x)$.

As a further illustration of the expressive power of CertiPriv, we have also defined a Laplacian mechanism $\mathcal{L}^n$ for lists; given $\sigma \in \mathbb{R}^+$ and a vector $a \in \mathbb{Z}^n$, the mechanism $\mathcal{L}^n$ outputs a vector in $\mathbb{Z}^n$ whose i-th component is drawn from distribution $\mathcal{L}(a[i], \sigma)$. More formally, we have proved the soundness of the following rule

$$\frac{m_1\ \Psi\ m_2 \implies \sum_{1\leq i\leq n} |[\![a[i]]\!]\ m_1 - [\![a[i]]\!]\ m_2| \leq k}{\models x \xleftarrow{\$} \mathcal{L}^n(a, \frac{k}{\epsilon}) \sim_{\exp(\epsilon),0} y \xleftarrow{\$} \mathcal{L}^n(a, \frac{k}{\epsilon}) : \Psi \Rightarrow x\langle 1\rangle = y\langle 2\rangle}\ [\mathsf{lap}^n]$$

6.2. Statistics over Streams

In this section we present analyses of algorithms for computing private and continual statistics in data streams [Chan et al. 2010]. As in [Chan et al. 2010], we focus on algorithms for private summing and counting. More sophisticated algorithms, e.g. computing heavy hitters in a data stream, can be built using sums and counters as primitive operations and inherit their privacy and utility guarantees.

We consider streams of elements in a bounded subset $D \subseteq \mathbb{Z}$, i.e. with $|x - y| \leq b$ for all $x, y \in D$. This setting is slightly more general than the one considered by Chan et al. [Chan et al. 2010], where only streams over $\{0,1\}$ are considered. On the algorithmic side, the generalization to bounded domains is immediate; for the privacy analysis, however, one needs to take the bound b into account because it conditions the sensitivity of computations. This requires a careful definition of metrics and propagation of bounds, which is supported by CertiPriv.

Although in our implementation we formalize streams as finite lists, we use array-notation in the exposition for the sake of readability. Given an array a of n elements in D, the goal is to release, for every point $0 \le j < n$ the aggregate sum $c[j] = \sum_{i=0}^{j} a[i]$ in a privacy-preserving manner. As observed in [Chan et al. 2010], there are two immediate solutions to the problem. The first is to maintain an exact aggregate sum $c[j]$ and output at each iteration a curated version $\overline{c}[j] \xleftarrow{\$} \mathcal{L}(c[j], b/\epsilon)$ of that sum. The second solution is to maintain and output a noisy aggregate sum $\tilde{c}[j]$, which is updated at iteration $j+1$ according to

$$\overline{a}[j+1] \xleftarrow{\$} \mathcal{L}(a[j+1], b/\epsilon);\ \tilde{c}[j+1] \leftarrow \tilde{c}[j] + \overline{a}[j+1]$$

The stream $\overline{c}[0]\cdots\overline{c}[n-1]$ offers weak, $n\epsilon$-differential privacy, because every element of a may appear in n different elements of $\overline{c}$, each with independent noise. However, each $\overline{c}[j]$ offers good accuracy because noise is added only once. In contrast, the stream $\tilde{c}[0]\cdots\tilde{c}[n-1]$ offers improved, ϵ-differential privacy, because each element of a appears only in one ϵ-differentially private query. However, as shown in [Chan et al. 2010], the sum $\tilde{c}[j]$ yields poor accuracy because noise is added j times during its computation.

One solution proposed by Chan et al. [Chan et al. 2010] is a combination of both basic methods of releasing partial sums that achieves a good compromise between privacy and accuracy. The idea is to split the stream a into chunks of length q, where the less accurate (but more private) method is used to compute the sum within the current chunk, and the more accurate (but less private) method is used to compute summaries of previous chunks. Formally, let $s_t = \sum_{i=0}^{q-1} a[t\ q + i]$ be the sum over the t-th chunk of a and let $\overline{s}_t \xleftarrow{\$} \mathcal{L}(s_t, b/\epsilon)$ be the corresponding noisy version. Then, for each $j = qr + k$, with $k < q$, we compute

$$\hat{c}[j] = \sum_{t=0}^{r-1} \overline{s}_t + \sum_{i=0}^{k} \overline{a}[qr+i]$$

The sequence $\hat{c}[0]\cdots\hat{c}[n-1]$ offers 2ϵ-differential privacy, intuitively because each element of a is accessed twice during computation. Moreover, $\hat{c}[j]$ also offers improved accuracy over $\tilde{c}[j]$ because noise is added only $r+k$ times rather than $j = qr+k$ times.

We will now turn the above informal security analysis into a formal analysis of program code. The code for computing $\overline{s}_t$ is given as the function PARTIALSUM in Figure 7, the code for computing $\overline{c}$ is given as the function PARTIALSUM' in Figure 8, and the code for computing $\hat{c}$ is given as the function SMARTSUM in Figure 9. We next sketch the key steps in our proofs of differential privacy bounds for each of these algorithms. For all of our examples, we use the pre-condition

$$\begin{aligned}\Psi \stackrel{\text{def}}{=}\ & \mathsf{length}(a\langle 1\rangle) = \mathsf{length}(a\langle 2\rangle) \wedge a\langle 1\rangle \doteq a\langle 2\rangle \wedge \\ & \forall i.\ 0 \le i < \mathsf{length}(a\langle 1\rangle) \implies |a[i]\langle 1\rangle - a[i]\langle 2\rangle| \le b\end{aligned}$$

which relates two lists $a\langle 1\rangle$ and $a\langle 2\rangle$ whenever they have the same length, differ in at most one element, and the distance between the elements at the same position at each array is upper-bounded by b.

PARTIALSUM. The proof of differential privacy of PARTIALSUM proceeds in two key steps. First, we prove (using the pRHL fragment of apRHL) that

$$\models c_{1-5} \sim_{1,0} c_{1-5} : \Psi \Rightarrow |s\langle 1\rangle - s\langle 2\rangle| \le b$$

```
function PartialSum(a)
1 s ← 0; i ← 0;
2 while i < length(a) do
3     s ← s + a[i];
4     i ← i + 1;
5 end;
6 s ← ℒ(s, b/ϵ)
```

Fig. 7. A simple ϵ-differentially private algorithm for sums over streams

```
function PartialSum'(a)
1 ā ←$ ℒ^n(a, b/ϵ);
2 s[0] ← ā[0]; i ← 1;
3 while i < length(a) do
4     s[i] ← s[i − 1] + ā[i];
5     i ← i + 1;
6 end
```

Fig. 8. An ϵ-differentially private algorithm for partial sums over streams

where c_{1-5} corresponds to the code in lines 1-5 in Figure 7, i.e. the initialization and the loop. We apply the rule [lap] that gives a bound for the privacy guarantee achieved by the Laplacian mechanism (see Figure 6) to $c_6 = s \xleftarrow{\$} \mathcal{L}(s, b/\epsilon)$ (the instruction in line 6) and derive

$$\models c_6 \sim_{\exp(\epsilon),0} c_6 : |s\langle 1\rangle - s\langle 2\rangle| \leq b \Rightarrow s\langle 1\rangle = s\langle 2\rangle$$

Using rule [seq], applied to c_{1-5} and c_6, we derive the following statement about PartialSum, which implies that its output s is ϵ-differentially private.

$$\models \textsc{PartialSum}(a) \sim_{\exp(\epsilon),0} \textsc{PartialSum}(a) : \Psi \Rightarrow s\langle 1\rangle = s\langle 2\rangle$$

PartialSum'. Our implementation of PartialSum' in Figure 8 differs slightly from the description given above in that we first add noise to the entire stream (line 1), before computing the partial sums of the noisy stream (lines 2-6). This modification allows us to take advantage of the proof rule for the Laplacian mechanism on lists. By merging the addition of noise into the loop, our two-pass implementation can be turned into an observationally equivalent one-pass implementation suitable for processing streams of data.

The proof of privacy for PartialSum' proceeds in the following basic steps. First, we apply the rule [lap^n] to the random assignment in line 1 (noted as c_1) of PartialSum'. We obtain

$$\models c_1 \sim_{\exp(\epsilon),0} c_1 : \Psi \Rightarrow \overline{a}\langle 1\rangle = \overline{a}\langle 2\rangle$$

i.e. the output $\overline{a}$ is ϵ-differentially private at this point. For lines 2-6 (denoted by c_{2-6}), we prove (using the pRHL fragment of apRHL) that

$$\models c_{2-6} \sim_{1,0} c_{2-6} : \overline{a}\langle 1\rangle = \overline{a}\langle 2\rangle \Rightarrow s\langle 1\rangle = s\langle 2\rangle$$

This is straightforward because of the equality appearing in the pre-condition; this result can be derived using apRHL rules, but is also an immediate consequence of the preservation of α-distance by probabilistic computations (see Lemma 4.3).

Finally, we apply the rule for sequential composition to c_1 and c_{2-6} and obtain

$$\models \textsc{PartialSum'}(a) \sim_{\exp(\epsilon),0} \textsc{PartialSum'}(a) : \Psi \Rightarrow s\langle 1\rangle = s\langle 2\rangle$$

which implies that the output s of PartialSum' is ϵ-differentially private.

```
function SmartSum(a, q)
1 i ← 0; c ← 0;
2 while i < length(a)/q do
3    b ← PartialSum(a[i q..i(q + 1) − 1]);
4    x ← PartialSum'(a[i q..i(q + 1) − 1]);
5    s ← OffsetCopy(s, x, c, iq, q);
6    c ← c + b;
7    i ← i + 1;
8 end
```

Fig. 9. A 2ϵ-differentially private algorithm for partial sums over streams ($a[i..j]$ denotes the sub-array of a at entries i through j).

SmartSum. Our implementation of the smart private sum algorithm in Figure 9 makes use of PartialSum and PartialSum' as building blocks, which enables us to reuse the above proofs. In addition, we use of a procedure OffsetCopy that given two lists s and x, a constant c and non-negative integers i, q, returns a list which is identical to s, but where the entries $s[i] \cdots s[i + (q - 1)]$ are replaced by the first q elements of x, plus a constant offset c, i.e. $s[i + j] = x[j] + c$ for $0 \leq i < q$. We obtain

$$\models s \leftarrow \text{OffsetCopy}(s, x, c, i, q) \sim_{1,0} s \leftarrow \text{OffsetCopy}(s, x, c, i, q) : =_{\{s,x,c,i,q\}} \Rightarrow s\langle 1\rangle = s\langle 2\rangle$$

We combine this result with the judgments derived for PartialSum and PartialSum' using the rule for sequential composition, obtaining

$$\models c_{4-7} \sim_{\exp(2\epsilon),0} c_{4-7} : \Psi \Rightarrow s\langle 1\rangle = s\langle 2\rangle$$

where c_{4-7} denotes the body of the loop in lines 4-7. To conclude, we apply the rule for while loops in Fig. 5 with $\alpha_1(i) = 1$ and $\alpha_2 = \exp(2\epsilon)$. This instantiation of the rule states that a loop that is non-interfering in all but one iteration is 2ϵ-differentially private, if the interfering loop iteration is 2ϵ-differentially private. More technically, the existence of a single interfering iteration is built into the rule using a stable predicate of the state of the program. In our case, the critical iteration corresponds to the one in which the chunk processed contains the entry in which the two streams differ.

6.3. k-Median

We discuss next a private version of the k-Median problem [Gupta et al. 2010]. This problem constitutes an instance of the so called *facility location problems*, whose goal is to find an optimal placement for a set of facilities intended to serve a given set of clients. To model this family of problems we assume the existence of a finite set of points V and a quasimetric $d : V \times V \to \mathbb{R}^{\geq 0}$ on this set. (A quasimetric is metric without the symmetry requirement.) Facilities and clients are represented by the points in V, whereas d measures the cost of matching a client to a facility. Given an integer k and a set $C \subseteq V$ of *clients*, the aim of the k-Median problem is to select a set $F \subseteq V$ of *facilities* of size k that minimizes the sum of the distance of each client to the nearest facility. Formally, this corresponds to minimizing the objective function

$$cost_C(F) \stackrel{\text{def}}{=} \sum_{c \in C} d(c, F) \qquad \text{where } d(c, F) \stackrel{\text{def}}{=} \min_{f \in F} d(c, f)$$

As finding the optimal solution is hard in general, in practice, one has to resort to heuristic techniques. In particular, one can perform a time-bounded local search to find an approximation of the optimal solution. *Local search* is a general-purpose heuristic aimed to find a solution within a search space that maximizes (or minimizes) the value of some objective function. Given a neighborhood relation on the search space and an initial candidate solution, the local search heuristic proceeds by iteratively replacing

```
function kMedian(C, ε, S_0)
1 i ← 0; S[0] ← S_0;
2 while i < T do
3     (x, y) ←$ pick_swap(ε, C, S[i], S[i] × (V \ S[i]));
4     S[i + 1] ← (S[i] \ {x}) ∪ {y};
5     i ← i + 1
6 end;
7 j ←$ pick_solution(ε, C, T, S)
```

Fig. 10. A $2\epsilon\Delta(T+1)$-differentially private algorithm for computing the k-Median

the current solution with one within its neighborhood, until some time bound or some "good" sub-optimal solution is reached. The simplest way to implement the local search technique for the k-Median problem is by considering two sets of facilities to be neighbors iff they differ in exactly one point and halting upon a predefined number of iterations. More precisely, the implementation we consider begins with an initial solution S_0 and in the i-th iteration, finds, if possible, a pair of points $(x, y) \in S_i \times (V \setminus S_i)$ such that the solution obtained from S_i by swapping x for y outperforms S_i; if this is the case, it sets the new solution S_{i+1} to $(S_i \setminus \{x\}) \cup \{y\}$.

Observe that the aforementioned heuristic might leak some information about the set of clients C. Gupta et al. [Gupta et al. 2010] showed how to turn this algorithm into a differentially private algorithm that conceals the presence or absence of any client in C. The crux is to rely on the Exponential mechanism to choose the pair of points $(x, y) \in S_i \times (V \setminus S_i)$ in a differentially private way. The description of the algorithm is given in Figure 10. We assume that the quasi-metric space (V, d) is fixed. Moreover, the algorithm is parametrized by an integer T, which determines the number of solution updates the local search will perform. The integer k is implicitly determined by the size of the initial solution S_0. Lines $1-6$ iteratively refine S_0 and store all the intermediate solutions in S (we use array-notation to refer to these solutions, in our Coq formalization we use lists). Line 7 picks the (index of the) solution to be output by the algorithm.

In each iteration of the loop, the algorithm updates the current solution $S[i]$ by substituting one of its points. That is, it chooses a point x in $S[i]$ and a point y not belonging to $S[i]$ and swaps them. In order to do so in a differentially private way the algorithm uses (a variant of) the Exponential mechanism. Specifically, the pair of points (x, y) is drawn from the parametrized distribution pick_swap. Given $C, F \subseteq V$, $R \subseteq V \times V$ and $\epsilon > 0$, distribution $\mathsf{pick\_swap}(\epsilon, C, F, R)$ assigns to each pair (x, y) in R a probability proportional to $\exp\left(-\epsilon\, cost_C((F \setminus \{x\}) \cup \{y\})\right)$. Technically, this mechanism is defined as an instance of the construction $(\cdot)^\sharp$ introduced in Section 6.1:

$$\mathsf{pick\_swap}(\epsilon, C, F, R) \stackrel{\text{def}}{=} g_{\epsilon,R}{}^\sharp\ (C, F)$$

where $g_{\epsilon,R}$ has type $\mathcal{P}(V)^2 \to R \to \mathbb{R}^{\geq 0}$ and is defined as

$$g\ (C, F)\ (x, y) \stackrel{\text{def}}{=} \exp\left(-\epsilon\, cost_C((F \setminus \{x\}) \cup \{y\})\right)$$

During a solution update, pairs of vertices with lower resulting cost are more likely to be chosen. However, swapping such pairs might deliver increased values of the cost function (for instance, when dealing with a local minimum). This raises the need to choose one of the computed solutions, in accordance with the value assigned to them by the objective function. Likewise, this choice should not leak any information about the clients in C. This is accomplished by distribution pick_solution in line 7, which is defined in the same spirit as pick_swap by equation:

$$\mathsf{pick\_solution}(\epsilon, C, T, S) \stackrel{\text{def}}{=} h_{\epsilon,T,S}{}^\sharp\ C$$

where $T \in \mathbb{N}$, S is an array of T sets of points from V, and $h_{\epsilon,T,S}$ has type $\mathcal{P}(V) \to \{0,\ldots,T-1\} \to \mathbb{R}^{\geq 0}$ and is defined as

$$h_{\epsilon,T,S}(C,j) \stackrel{\text{def}}{=} \exp\left(-\epsilon\; cost_C(S[j])\right)$$

The original proof [Gupta et al. 2010] shows that the algorithm in Figure 10 is $2\epsilon\Delta(T+1)$-differentially private, where $\Delta = \max_{v_1,v_2\in V} d(v_1,v_2)$ is the diameter of the quasi-metric space. The key steps in the proof are as follows. First show that for every $F \subseteq V$, the function $cost_{(\cdot)}(F)$ has sensitivity Δ. Let $C_1 = \{c_0, c_1, \ldots, c_m\}$ and $C_2 = \{c'_0, c_1, \ldots, c_m\}$ be two subsets of V differing in at most one point. Then,

$$|cost_{C_1}(F) - cost_{C_2}(F)| = \left|\min_{f\in F} d(c_0,f) - \min_{f\in F} d(c'_0,f)\right| \leq \Delta$$

where the last inequality holds because both terms $\min_{f\in F} d(c_0,f)$ and $\min_{f\in F} d(c'_0,f)$ are non-negative and upper-bounded by Δ. Now observe that the mechanisms used to choose the pair of points (x,y) (line 3) and to pick the output solution (line 7) can be viewed as instances of the Exponential mechanism with uniform base distributions and score functions $\lambda C\; F\; (x,y).\, -cost_C((F\setminus\{x\})\cup\{y\})$ and $\lambda C\; j.\, -cost_C(S[j])$ respectively, having each of them sensitivity Δ. Therefore each of them is $2\epsilon\Delta$-differentially private. Since privacy composes additively and step 3 is run T times one concludes that the algorithm is $2\epsilon\Delta(T+1)$-differentially private.

Next we present a language-based analysis of this security result using apRHL. The privacy statement is formalized by the judgment

$$\models \text{KMEDIAN}(C,\epsilon,S_0) \sim_{\exp(2\epsilon\Delta(T+1)),0} \text{KMEDIAN}(C,\epsilon,S_0) : \Psi \Rightarrow S[j]\langle 1\rangle = S[j]\langle 2\rangle \quad (2)$$

where $\Psi \stackrel{\text{def}}{=} S_0\langle 1\rangle = S_0\langle 2\rangle \wedge C\langle 1\rangle \doteq C\langle 2\rangle$. We let $c = \text{KMEDIAN}(C,\epsilon,S_0)$ and use the same convention as in Section 6.2 to denote program fragments by indicating the initial and final lines in subscript.

We begin by applying the rule for sequential composition [seq], which enables to derive the privacy condition (2) from judgments

$$\models c_{1-6} \sim_{\exp(2\epsilon\Delta T),0} c_{1-6} : \Psi \Rightarrow I$$

and

$$\models c_7 \sim_{\exp(2\epsilon\Delta),0} c_7 : I \Rightarrow S[j]\langle 1\rangle = S[j]\langle 2\rangle$$

where $I \stackrel{\text{def}}{=} i\langle 1\rangle = i\langle 2\rangle \wedge S\langle 1\rangle = S\langle 2\rangle \wedge C\langle 1\rangle \doteq C\langle 2\rangle$.

The former is derived with an application of rule [while] with $n = T$, $\Theta = I$, $\alpha = \exp(2\Delta\epsilon)$, and $\delta = 0$. Rule [while] is enough because α and δ are constant across iterations. To prove the premise

$$\models c_{3-5} \sim_{\exp(2\epsilon\Delta),0} c_{3-5} : I \wedge (i<T)\langle 1\rangle \wedge (i<T)\langle 2\rangle \Rightarrow I \wedge \neg(i<T)\langle 1\rangle \wedge \neg(i<T)\langle 2\rangle$$

we use rule [assn] to deal with lines 4 and 5, rule [rand] to deal with the random assignment in line 3, and rule [frame] to prepare for this application. By setting $\mu = \mathsf{pick\_swap}(\epsilon, C, S[i], S[i] \times (V \setminus S[i]))$, the premise

$$\forall m_1\; m_2.\; m_1\; I\; m_2 \implies \Delta_{\exp(2\epsilon\Delta)}(\llbracket\mu\rrbracket\; m_1, \llbracket\mu\rrbracket\; m_2)$$

of rule [rand] can be discharged by Lemma 6.2, which requires showing that for all C_1, C_2, F, x and y satisfying $C_1 \doteq C_2 \;\wedge\; x \in F \;\wedge\; y \notin F$,

$$\exp(-\epsilon\; cost_{C_1}((F\setminus\{x\})\cup\{y\})) \leq \exp(\epsilon\Delta)\; \exp(-\epsilon\; cost_{C_2}((F\setminus\{x\})\cup\{y\}))$$

This inequality is a direct consequence of the sensitivity property of the function $cost$ stated above.

We are left to verify the second premise of the [seq] rule. We follow a similar reasoning to the one above, that is, we rely on rule [rand] and on rule [frame] to prepare for its application. The reasoning boils down to showing

$$\forall m_1\ m_2.\ m_1\ S = m_2\ S \wedge m_1\ C \doteq m_2\ C \implies \Delta_{\exp(2\epsilon\Delta)}([\![\mu']\!]\ m_1, [\![\mu']\!]\ m_2)$$

where $\mu' = \mathsf{pick\_solution}(\epsilon, C, T, S)$. Similarly, this requires proving that for all C_1, C_2, S and j satisfying $C_1 \doteq C_2\ \wedge\ 0 \leq j < T$,

$$\exp(-\epsilon\ cost_{C_1}(S[j])) \leq \exp(\epsilon\Delta)\ \exp(-\epsilon\ cost_{C_2}(S[j]))$$

which follows from the sensitivity of function $cost$.

6.4. Minimum Vertex Cover

We conclude this section with a more detailed account of the proof of differential privacy of the Minimum Vertex Cover approximation algorithm of Section 2. To obtain sharper privacy bounds, we recast the privacy statement using the asymmetric logic apRHL*. Rather than a single pRHL judgment, we need to prove the following pair of apRHL* judgments:

$$\models^\star \textsc{VertexCover}(V, E, \epsilon) \sim_{\exp(\epsilon),0} \textsc{VertexCover}(V, E, \epsilon) : \Psi_1 \Rightarrow \Phi \tag{3}$$

$$\models^\star \textsc{VertexCover}(V, E, \epsilon) \sim_{\exp(\epsilon),0} \textsc{VertexCover}(V, E, \epsilon) : \Psi_2 \Rightarrow \Phi \tag{4}$$

where

$$\begin{array}{rcl} \Psi_1 & \stackrel{\text{def}}{=} & V\langle 1\rangle = V\langle 2\rangle \wedge E\langle 2\rangle = E\langle 1\rangle \cup \{(t,u)\} \\ \Psi_2 & \stackrel{\text{def}}{=} & V\langle 1\rangle = V\langle 2\rangle \wedge E\langle 1\rangle = E\langle 2\rangle \cup \{(t,u)\} \\ \Phi & \stackrel{\text{def}}{=} & \pi\langle 1\rangle = \pi\langle 2\rangle \end{array}$$

Let us focus first on (3). We prove the validity of this judgment using an asymmetric variant of the generalized rule for while loops given in Figure 5. This rule, which we call [gwhile*], has the same shape as [gwhile], but judgments in the premises and conclusion are interpreted in apRHL*. We apply the rule with parameters

$$\alpha_1(i) = \exp\left(\frac{2}{(n-i)w_i}\right) \qquad \alpha_2 = 1,$$

the loop invariant

$$\Theta = \begin{array}{l} (t \in \pi\langle 1\rangle \vee u \in \pi\langle 1\rangle \implies E\langle 1\rangle = E\langle 2\rangle) \wedge \\ (t \notin \pi\langle 1\rangle \wedge u \notin \pi\langle 1\rangle \implies E\langle 2\rangle = E\langle 1\rangle \cup \{(t,u)\}) \wedge \\ V\langle 1\rangle = V\langle 2\rangle \wedge \pi\langle 1\rangle = \pi\langle 2\rangle \wedge i\langle 1\rangle = i\langle 2\rangle, \end{array}$$

and the stable property

$$P = t \in \pi \vee u \in \pi$$

The first and second judgments appearing in the premises of the rule are of the form

$$\models^\star c; \mathsf{assert}\ P \sim_{\alpha,0} c; \mathsf{assert}\ P : \Psi' \Rightarrow \Phi' \quad \text{and} \quad \models^\star c; \mathsf{assert}\ \neg P \sim_{\alpha,0} c; \mathsf{assert}\ \neg P : \Psi' \Rightarrow \Phi'$$

where c is the body of the loop. For each of these premises, we first hoist the assertion immediately after the random assignment that chooses the vertex v in c. As a result, the expression in the assertion becomes $(t, u \notin (v :: \pi))$ in the case of the first premise, and $(t \in (v :: \pi) \vee u \in (v :: \pi))$ in the case of the second. We then compute the weakest pre-condition of the assignments that now follow the assertions. The resulting judgments simplify, after applying the [weak*] and [frame*] rules, to judgments of the form

$$\models^\star c' \sim_{\alpha,0} c' : \Psi'' \Rightarrow v\langle 1\rangle = v\langle 2\rangle$$

where

$$\Psi'' \stackrel{\text{def}}{=} E\langle 2\rangle = E\langle 1\rangle \cup \{(t,u)\} \wedge V\langle 1\rangle = V\langle 2\rangle \ \wedge t, u \notin \pi \wedge i\langle 1\rangle = i\langle 2\rangle = j \wedge \pi\langle 1\rangle = \pi\langle 2\rangle$$

For the first premise we have $\alpha = \alpha_1(j)$ and

$$c' = v \xleftarrow{\$} \mathsf{choose}(V, E, \epsilon, n, i);\ \mathsf{assert}\ (t, u \notin (v :: \pi))$$

whereas for the second premise we have $\alpha = \alpha_2$ and

$$c' = v \xleftarrow{\$} \mathsf{choose}(V, E, \epsilon, n, i);\ \mathsf{assert}\ (t \in (v :: \pi) \vee u \in (v :: \pi))$$

To establish the validity of each judgment, we cast the code for c' as a random assignment where v is sampled from the interpretation of $\mathsf{choose}(V, E, \epsilon, n, i)$ restricted to v satisfying the condition in the assertion. For the first premise, the restriction amounts to $v \neq u, t$ whereas for the second it amounts to $v = t \vee v = u$. In either case, we apply the asymmetric rule for random assignments [rand⋆] and are thus left to prove that the distance $\Delta^\star_\alpha$ between the corresponding distributions is 0. In view of the variant of Lemma 4.6 for $\Delta^\star_\alpha$ this in turn amounts to verifying that for each vertex x, the ratio between the probability of choosing x in each distribution is bounded by α, which directly translates into the inequalities presented in the initial analysis of the algorithm. Technically, these inequalities are proved by appealing to a variant of Corollary 6.2.

To prove the remaining judgment (4) we follow a similar reasoning; we apply rule [gwhile⋆] with parameters

$$\alpha_1(i) = 1 \qquad \alpha_2 = \exp(\epsilon/4),$$

the loop invariant

$$\Theta = \begin{array}{l} (t \in \pi\langle 1\rangle \vee u \in \pi\langle 1\rangle \implies E\langle 1\rangle = E\langle 2\rangle) \wedge \\ (t \notin \pi\langle 1\rangle \wedge u \notin \pi\langle 1\rangle \implies E\langle 1\rangle = E\langle 2\rangle \cup \{(t,u)\}) \wedge \\ V\langle 1\rangle = V\langle 2\rangle \wedge \pi\langle 1\rangle = \pi\langle 2\rangle \wedge i\langle 1\rangle = i\langle 2\rangle, \end{array}$$

and the same stable property as before

$$P = t \in \pi \vee u \in \pi$$

As a final remark, we observe that the use of apRHL* is fundamental to prove the privacy bound ϵ from [Gupta et al. 2010], as opposed to [Barthe et al. 2012], where the use of apRHL yields a looser bound of $5\epsilon/4$. This is because the proof in apRHL* allows to prove independently that $\exp(\epsilon)$ is a bound for the ratios

$$\frac{\Pr\left[\textsc{VertexCover}(G_1, \epsilon) : \pi = \vec{v}\right]}{\Pr\left[\textsc{VertexCover}(G_2, \epsilon) : \pi = \vec{v}\right]} \quad \text{and} \quad \frac{\Pr\left[\textsc{VertexCover}(G_2, \epsilon) : \pi = \vec{v}\right]}{\Pr\left[\textsc{VertexCover}(G_1, \epsilon) : \pi = \vec{v}\right]},$$

while a proof in apRHL requires to prove this simultaneously. As a consequence, the proof in apRHL* consists of two independent applications of the asymmetric rule [gwhile⋆]. One application requires to bound for each iteration of the loop the ratio

$$\frac{\mathsf{Pr}[v\langle 2\rangle = x]}{\mathsf{Pr}[v\langle 1\rangle = x]}$$

while the other requires to bound its reciprocal. For each application, one can choose independent—and thus tighter—parameters α_1 and α_2; namely $(\alpha_1(i), \alpha_2) = (\exp\left(2/((n-i)w_i)\right), 1)$ and $(\alpha_1(i), \alpha_2) = (1, \exp(\epsilon/4))$. In contrast, when using the symmetric logic apRHL, one needs to choose a single pair of parameters to bound both ratios simultaneously, namely $(\alpha_1(i), \alpha_2) = (\exp\left(2/((n-i)w_i)\right), \exp(\epsilon/4))$. This translates into a looser privacy bound.

7. LIFTING AS AN OPTIMIZATION PROBLEM

Proving the validity of an apRHL judgment $c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$ boils down to proving that for any pair of memories m_1, m_2 related by the pre-condition Ψ, the distributions $[\![c_1]\!]\, m_1$ and $[\![c_2]\!]\, m_2$ are related by the (α,δ)-lifting of the post-condition Φ. Thus, a first step to automate reasoning in apRHL is to provide a procedure to decide the (α,δ)-lifting of a relation.

In this section, we establish a correspondence between deciding (α,δ)-lifting and finding a minimum loss flow in networks with multiplicative losses and gains. Our result extends and generalizes a known connection between less expressive classes of liftings and network flow problems [Jonsson et al. 2001; Desharnais et al. 2008]. This correspondence allows us to cast the problem of proving that a pair of distributions is in the (α,δ)-lifting of a relation as an optimization problem, and to use any available algorithm to solve the latter (e.g. linear programming methods).

We begin by recalling some basic definitions about flow networks [Lawler 1976; Murty 1992]. A *network* is a tuple $(V, E, \bot, \top, c)$ where $G = (V \cup \{\bot, \top\}, E)$ is a finite directed graph with a distinguished *source* ($\bot$) and *sink* ($\top$), and $c : E \to \mathbb{R}^{\geq 0}$ is a function assigning a non-negative *capacity* $c(e)$ to each edge e in E. A *network with losses and gains* (NLG) is a network in which each edge e is also given a positive *gain* $\gamma(e) \in \mathbb{R}^{>0}$. For such a network we say that a mapping $f : E \to \mathbb{R}^{\geq 0}$ is a *feasible flow* iff it satisfies the following conditions:

$$\begin{array}{lll} 0 \leq f(e) \leq c(e) & \forall e \in E & \text{(capacity constraints)} \\ Ex_f(v) = 0 & \forall v \in V & \text{(flow conservation)} \end{array}$$

where $Ex_f(v) = \sum_{e \in in(v)} \gamma(e) f(e) - \sum_{e \in out(v)} f(e)$ is the *flow excess* at vertex v and $in(v)$ and $out(v)$ represent the sets of incoming and outgoing edges, respectively.

This problem generalizes the standard network flow problem in the sense that one allows flow along an edge not to be conserved: if $f(u,v)$ units of flow enter edge (u,v) then $\gamma(u,v)f(u,v)$ arrive at its head v. Due to these losses and gains, the flow $f_\bot = -Ex_f(\bot)$ leaving the source and the flow $f_\top = Ex_f(\top)$ arriving at the sink might be different. The difference $f_\bot - f_\top$ is called the *loss* of f.

The *minimum flow loss problem* is the problem of finding a feasible flow f^* of minimum loss. This is usually done by fixing the value of $f_\bot$ and maximizing $f_\top$ (or dually, by fixing $f_\top$ and minimizing $f_\bot$). Since the minimum flow loss problem can be formulated as a linear program, it can be solved in polynomial time using, e.g. the ellipsoid method or Karmarkar's algorithm. It can also be solved by polynomial combinatorial algorithms, which exploit the structure of the underlying network [Tardos and Wayne 1998; Goldfarb et al. 1997].

We now show how deciding whether $\mu_1 \sim_R^{\alpha,\delta} \mu_2$ holds can be reduced to finding feasible flows in a suitable NLG. For given distributions $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$, and $R \subseteq A \times B$ we define the NLG $\mathcal{N}(\mu_1, \mu_2, R, \alpha)$ by setting $V = A \cup B$ and $E = (\{\bot\} \times A) \cup R \cup (B \times \{\top\})$, and defining capacity and gain as follows:

$$c(u,v) = \begin{cases} \mu_1(v) & \text{if } u = \bot \text{ and } v \in A \\ \mu_2(u) & \text{if } u \in B \text{ and } v = \top \\ 1 & \text{if } u \in A \text{ and } v \in B \end{cases} \qquad \gamma(u,v) = \begin{cases} \alpha^{-1} & \text{if } u = \bot \text{ and } v \in A \\ 1 & \text{if } u \in B \text{ and } v = \top \\ \alpha & \text{if } u \in A \text{ and } v \in B \end{cases}$$

That is, the vertex adjacency relation corresponds to R, together with edges from the source to A and from B to the sink. Intuitively, the edges exiting the source and the edges entering the sink labelled with their capacities represent distributions μ_1 and μ_2, respectively, while the flow along edges joining R-related elements determines a product distribution over $A \times B$. In Figure 11 we illustrate the construction of such a network.

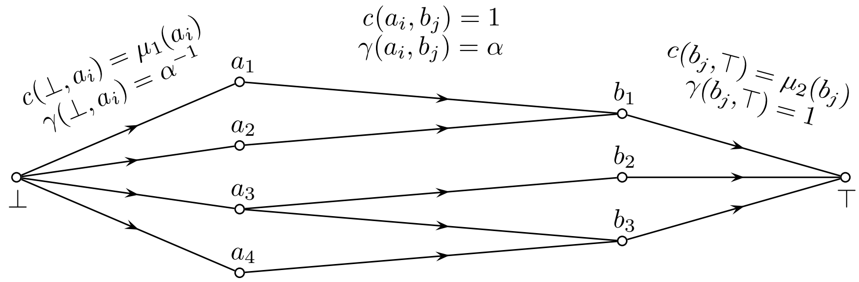


Fig. 11. Network $\mathcal{N}(\mu_1, \mu_2, R, \alpha)$ used to interpret the lifting of relation R as a minimum flow loss problem. Here, $A = \{a_1, \cdots, a_4\}$, $B = \{b_1, \cdots, b_3\}$, and $R = \{(a_1, b_1), (a_2, b_1), (a_3, b_2), (a_3, b_3), (a_4, b_3)\}$.

The following result establishes a correspondence between feasible flows in the network $\mathcal{N}(\mu_1, \mu_2, R, \alpha)$ and witnesses for the lifting $\mu_1 \sim_R^{\alpha,\delta} \mu_2$.

THEOREM 7.1. *Let μ_1 and μ_2 be a pair of distributions over finite sets A and B and let $R \subseteq A \times B$. Then, the statements*

(*1*) *there exists a feasible flow f in $\mathcal{N}(\mu_1, \mu_2, R, \alpha)$ s.t. $f_\bot \geq \mu_1(A) - \delta$ and $f_\top \geq \mu_2(B) - \delta$*
(*2*) *distributions μ_1 and μ_2 are related by the (α, δ)-lifting of R,*

are equivalent when $\alpha = 1$; if $\alpha \geq 1$, then (1) implies (2).

As a corollary, one can efficiently decide whether $\mu_1 \sim_R^{\alpha,\delta} \mu_2$ by solving a minimum flow loss problem in $\mathcal{N}(\mu_1, \mu_2, R, \alpha)$. It suffices to find a feasible flow f^* that maximizes $f_\top$ subject to $f_\bot = \mu_1(A) - \delta$. If $f^*_\top \geq \mu_2(B) - \delta$ then one has $\mu_1 \sim_R^{\alpha,\delta} \mu_2$, in which case a witness distribution μ for the lifting is readily obtained by taking $\mu(a, b) = f^*(a, b)$.

8. RELATED WORK

Our work builds upon program verification techniques, and in particular (probabilistic and relational) program logics, to reason about differential privacy. We briefly review relevant work in these areas.

Differential privacy. There is a vast body of work on differential privacy. We refer to recent overviews, see e.g. [Dwork 2008; Dwork 2011], for an account of some of the latest developments in the field, and focus on language-based approaches to differential privacy. The Privacy Integrated Queries (PINQ) platform [McSherry 2009] supports reasoning about the privacy guarantees of programs in a simple SQL-like language. The reasoning is based on the sensitivity of basic queries such as `Select` and `GroupBy`, the differential privacy of building blocks such as `NoisySum` and `NoisyAvg`, and meta-theorems for their sequential and parallel composition. AIRAVAT [Roy et al. 2010] leverages these building blocks for distributed computations based on MapReduce.

The linear type system of [Reed and Pierce 2010] extends sensitivity analysis to a higher-order functional language. By using a suitable choice of metric and probability monads, the type system also supports reasoning about probabilistic, differentially private computations. As in PINQ, the soundness of the type system makes use of known composition theorems and relies on assumptions about the sensitivity/differential privacy of nontrivial building blocks, such as arithmetic operations, conditional swap operations, or the Laplacian mechanism. While the type system can handle functional data structures, it does not allow for analyzing programs with conditional branching. Work on the automatic derivation of sensitivity properties of imperative programs [Chaudhuri et al. 2011] addresses this problem and can (in conjunction with the Laplacian mechanism) be used to derive differential privacy guarantees of programs with control flow. Although this approach supports reasoning about probabilistic computations, the reasoning is restricted to Lipschitz conditions.

In contrast to [McSherry 2009; Reed and Pierce 2010; Chaudhuri et al. 2011], CertiPriv supports reasoning about differential privacy guarantees from first principles. In particular, CertiPriv enabled us to prove (rather than to assume) the correctness of Laplacian, Gaussian and Exponential mechanisms, and the differential privacy of complex interleavings of (not necessarily differentially private) probabilistic computations. This comes at a price in automation; while the above systems are mostly automated, reasoning in apRHL in general cannot be fully automated.

Tschantz et al. [Tschantz et al. 2011] consider the verification of privacy properties based on I/O-automata. They focus on the verification of the correct use of differentially private sanitization mechanisms in interactive systems, where the effect of a mechanism is soundly abstracted using a single, idealized transition. Our verification-based approach shares many similarities with this method. In particular, their definition of differential privacy is also based on a notion of lifting that closely resembles the one we use to define validity in apRHL, and their unwinding-based verification method can be regarded as an abstract, language-independent, equivalent of apRHL. However, their method is currently limited to reason about ϵ-differential privacy.

An early approach to quantitative confidentiality analysis [Pierro et al. 2004] uses the distance of output distributions to quantify information flow. Their measure is closely related to $(0, \delta)$-approximate differential privacy, which can be reasoned about in CertiPriv. More recent approaches to quantitative information-flow focus on measures of confidentiality based on information-theoretic entropy. Techniques for code-based structural reasoning about these measures are developed in [Clark et al. 2007]. For an overview and a discussion of the relationship between entropy-based measures of confidentiality and differential privacy, see [Barthe and Köpf 2011].

Relational program verification. Program logics have a long tradition and have been used effectively to reason about functional correctness of programs. In contrast, privacy is a 2-safety property [Terauchi and Aiken 2005; Clarkson and Schneider 2010], that is, a (universally quantified) property about two runs of a program. There have been several proposals for applying program logics to 2-safety, but most of these proposals are confined to deterministic programs.

Program products [Zaks and Pnueli 2008; Barthe et al. 2011b] conflate two programs into a single one embedding the semantics of both. Product programs allow reducing the verification of 2-safety properties to the verification of safety properties on the product program, which can be done using standard program verification methods. Self-composition [Barthe et al. 2004] is a specific instance of product program.

Benton [Benton 2004] develops a relational Hoare logic (RHL) for a core imperative language and shows how it can be used to reason about information flow properties and correctness of compiler optimizations. Amtoft et al. [Amtoft and Banerjee 2004; Amtoft et al. 2006] develop specialized relational logics for information flow. Backes et al. [Backes et al. 2009] compute relational weakest preconditions as a basis for quantifying information leaks. Further applications of relational program verification include determinism [Burnim and Sen 2009] and robustness [Chaudhuri et al. 2011].

CertiCrypt [Barthe et al. 2009] is a machine-checked framework to reason about probabilistic computations in the presence of adversarial code. The main components of CertiCrypt are a formalization of pRHL, a relational Hoare logic for probabilistic programs, and a set of certified transformations. Although CertiCrypt has been used to verify several emblematic cryptographic constructions, pRHL does not support reasoning about statistical distance. Using a specialization of aPRHL with $\alpha = 1$, Barthe et al. [Barthe et al. 2012] prove indifferentiability from a random oracle of a hash function into elliptic curves. Moreover, Almeida et al. [Almeida et al. 2012] use the same logic to reason about statistical zero-knowledge.

EasyCrypt [Barthe et al. 2011a] is an automated tool that verifies automatically pRHL judgments using SMT solvers and a verification condition generator. In a follow-up work, we have extended EasyCrypt with support for reasoning about apRHL judgments and probabilistic operators, and with a simple mechanism to infer relational loop invariants. Moreover, we use this extension to build automated proofs of differential privacy for some of the examples reported in this paper, and interactive game-based proofs [Barthe et al. 2009] of computational differential privacy [Mironov et al. 2009] for 2-party computations.

Verification of probabilistic programs. Reif [Reif 1980], Kozen [Kozen 1985], and Feldman and Harel [Feldman and Harel 1984], were among the first to develop axiomatic logics for probabilistic computations. This line of work was further developed by Jones [Jones 1993], Morgan et al. [Morgan et al. 1996], den Hartog [den Hartog 1999], and more recently by Chadha et al. [Chadha et al. 2007]. Although their expressiveness varies, these logics are sufficiently expressive to reason about the probability of events in distributions generated by probabilistic programs. For instance, these logics have been used for proving termination of random walks, and correctness of probabilistic primality tests. As generalizations of Hoare logics, these logics are tailored towards trace properties rather than 2-safety properties like differential privacy. It should be possible to develop relational variants of these logics or to use self-composition for reasoning about differential privacy.

Hurd [Hurd 2003; Hurd et al. 2005] was among the first to develop a machine-checked framework to reason about probabilistic programs. His formalization is based on the standard notion of σ-algebras, and partly follows earlier formalizations in Mizar. Building on Hurd's work, Coble [Coble 2010], and Mhamdi, Hasan and Tahar [Mhamdi et al. 2010; Mhamdi et al. 2011] formalized integration theory in the HOL proof assistant. Coble [Coble 2008] also used the formalization to reason about privacy of solutions to the Dining Cryptographers problem. In contrast, our work is based on the ALEA library [Audebaud and Paulin-Mohring 2009], which follows a monadic approach to discrete probabilities. The library has been used to formally verify several examples of probabilistic termination. More recently, it has been used to reason about the security of watermarking algorithms [Baelde et al. 2012] and (in our own work) about cryptographic constructions [Barthe et al. 2009].

9. CONCLUSIONS

CertiPriv is a machine-checked framework that supports fine-grained reasoning about an expressive class of privacy policies in the Coq proof assistant. In contrast to previous language-based approaches to differential privacy, CertiPriv allows to reason directly about probabilistic computations and to build proofs from first principles. As a result, CertiPriv achieves flexibility, expressiveness, and reliability, and appears as a plausible starting point for formally analyzing new developments in the field of differential privacy.

In a follow-up work we study how to increase automation in differential privacy proofs using SMT solvers to mechanize reasoning in apRHL. It would be interesting to combine reasoning in apRHL with other automated analyses, such as the linear type system of [Reed and Pierce 2010], to achieve a higher degree of automation.

APPENDIX

We present proof sketches of most results in the body of the article. All results presented here and in the body of the paper have been formally verified using the Coq proof assistant, with the only exceptions of Lemma 4.8 and Theorem 7.1 which are not central to our development. We present first some auxiliary lemmas.

A. AUXILIARY LEMMAS

In the remainder, for a distribution $\mu \in \mathcal{D}(A)$ and a set $E \subseteq A$, we note the probability $\mu\ \mathbb{1}_E$ as $\mu(E)$. Moreover, for sets A and B and a relation $R \subseteq A \times B$, we use $\pi_1(R)$ to denote the set $\{a \in A \mid \exists b.\ a\ R\ b\}$ and $R(a)$ to denote the set $\{b \in B \mid a\ R\ b\}$.

LEMMA A.1. *Let $\mu \in \mathcal{D}(A)$ satisfy predicate* $\mathsf{range}\ P\ \mu$*. Then, for any $M : A \to \mathcal{D}(B)$, any predicate Q over B and any pair of functions $f, g : A \to [0,1]$,*

a) $(\forall a.\ P\ a \implies f\ a = g\ a) \implies \mu\ f = \mu\ g$
b) $(\forall a.\ P\ a \implies \mathsf{range}\ Q\ (M\ a)) \implies \mathsf{range}\ Q\ (\mathsf{bind}\ \mu\ M)$

PROOF.
a) To prove that $\mu\ f = \mu\ g$ it suffices to show that $\mu\ (\lambda a.\ |f\ a - g\ a|) = 0$. Since $\mathsf{range}\ P\ \mu$ holds, we can conclude by showing that the function $(\lambda a.\ |f\ a - g\ a|)$ is null at every point satisfying P, which follows from the premise of the implication.
b) Immediate from a). □

LEMMA A.2. *For any distribution μ over a discrete set A,*

$$\mathsf{range}\ P\ \mu \iff (\forall a.\ \mu(a) > 0 \implies P\ a)$$

PROOF. For the left to right direction, consider an element a not satisfying P and show that $\mu(a) = \mu\ \mathbb{1}_a = 0$. It suffices to verify that $\mathbb{1}_a$ is null at every element satisfying P, which is trivial because $\mathbb{1}_a$ is only not null at a. For the right to left direction, consider a function f such that $\forall a.\ P\ a \implies f(a) = 0$ and show that $\mu\ f = 0$ as follows:

$$\mu\ f = \sum_{a \in A} \mu(a) f(a) = \sum_{a \in A\ \mid\ 0 < \mu(a)} \mu(a) f(a) \ \leq \sum_{a \in A\ \mid\ P\ a} \mu(a) f(a) \ = 0 \qquad \square$$

LEMMA A.3. *Let μ_1 and μ_2 be two distributions over a discrete set A. Moreover, let $A_0 \stackrel{def}{=} \{a \in A \mid \mu_1(a) \geq \alpha\mu_2(a)\}$ and $A_1 \stackrel{def}{=} \{a \in A \mid \mu_2(a) \geq \alpha\mu_1(a)\}$. Then,*

$$\Delta_\alpha(\mu_1, \mu_2) = \max\{\mu_1(A_0) - \alpha\mu_2(A_0), \mu_2(A_1) - \alpha\mu_1(A_1)\}$$

PROOF. The inequality

$$\Delta_\alpha(\mu_1, \mu_2) \geq \max\{\mu_1(A_0) - \alpha\mu_2(A_0), \mu_2(A_1) - \alpha\mu_1(A_1)\}$$

follows trivially from the definition of α-distance between distributions. To prove the converse inequality, observe that for any $f : A \to [0,1]$ we have:

$$\begin{aligned}
\mu_1\ f - \alpha\mu_2\ f &= \sum_{a \in A} \mu_1(a) f(a) - \alpha \sum_{a \in A} \mu_2(a) f(a) \\
&= \sum_{a \in A_0} (\mu_1(a) - \alpha\mu_2(a)) f(a) \ + \sum_{a \notin A_0} (\mu_1(a) - \alpha\mu_2(a)) f(a) \\
&\leq \sum_{a \in A_0} \mu_1(a) - \alpha\mu_2(a) = \ \mu_1(A_0) - \alpha\mu_2(A_0)
\end{aligned}$$

In a similar way one can prove that $\mu_2\ f - \alpha\mu_1\ f \leq \mu_2(A_1) - \alpha\mu_1(A_1)$. By combining these two results one gets the desired inequality

$$\Delta_\alpha(\mu_1, \mu_2) \leq \max\{\mu_1(A_0) - \alpha\mu_2(A_0), \mu_2(A_1) - \alpha\mu_1(A_1)\} \qquad \square$$

LEMMA A.4. *For any two distributions* μ_1, μ_2 *over a discrete set* A*,*

$$\mu_2 \leq \mu_1 \implies \Delta_\alpha(\mu_1, \mu_2) = \mu_1(A_0) - \alpha\mu_2(A_0)$$

where $A_0 \stackrel{def}{=} \{a \in A \mid \mu_1(a) \geq \alpha\mu_2(a)\}$.

PROOF. Immediate from Lemma A.3, as condition $\mu_2 \leq \mu_1$ implies that $A_1 = \emptyset$, and thus $\Delta_\alpha(\mu_1, \mu_2) = \max\{\mu_1(A_0) - \alpha\mu_2(A_0), 0\}$. □

LEMMA A.5. *Let* A *and* B *be two discrete sets. Then, for any* $\mu_1, \mu_2 \in \mathcal{D}(A)$ *and* $M_1, M_2 : A \to \mathcal{D}(B)$ *that satisfy*

$$\Delta_\alpha(\mu_1, \mu_2) \leq \delta \qquad \textit{and} \qquad \forall a.\ \Delta_{\alpha'}(M_1\ a, M_2\ a) \leq \delta'$$

we have $\Delta_{\alpha\alpha'}(\mathsf{bind}\ \mu_1\ M_1, \mathsf{bind}\ \mu_2\ M_2) \leq \delta + \delta'$. *If, moreover, we have* $\mathsf{range}\ R\ \mu_1$ *and* $\mathsf{range}\ R\ \mu_2$ *for some predicate* R*, then the second hypothesis can be relaxed to*

$$\forall a.\ R\ a \implies \Delta_{\alpha'}(M_1\ a, M_2\ a) \leq \delta'$$

PROOF. As a first step, observe that

$$\Delta_{\alpha\alpha'}(\mathsf{bind}\ \mu_1\ M_1, \mathsf{bind}\ \mu_2\ M_2) \leq \Delta_{\alpha\alpha'}(\theta_1, \theta_2)$$

where $\theta_1, \theta_2 \in \mathcal{D}(A \times B)$ are defined as

$$\theta_1 \stackrel{\text{def}}{=} \mathsf{bind}\ \mu_1\ (\lambda a.\ \mathsf{unit}\ (a, M_1\ a)) \qquad \theta_2 \stackrel{\text{def}}{=} \mathsf{bind}\ \mu_2\ (\lambda a.\ \mathsf{unit}\ (a, M_2\ a))$$

This follows from Lemma 4.3.6, since $\pi_2\theta_1 = \mathsf{bind}\ \mu_1\ M_1$ and $\pi_2\theta_2 = \mathsf{bind}\ \mu_2\ M_2$.

We now apply Lemma A.3 to bound $\Delta_{\alpha\alpha'}(\theta_1, \theta_2)$. We are left to prove

$$\theta_1(X_0) - \alpha\alpha'\ \theta_2(X_0) \leq \delta + \delta' \qquad \text{and} \qquad \theta_2(X_1) - \alpha\alpha'\ \theta_1(X_1) \leq \delta + \delta'$$

where $X_0 \stackrel{\text{def}}{=} \{(a, b) \mid \theta_1(a, b) \geq \alpha\alpha'\ \theta_2(a, b)\}$ and $X_1 \stackrel{\text{def}}{=} \{(a, b) \mid \theta_2(a, b) \geq \alpha\alpha'\ \theta_1(a, b)\}$. We proceed to prove the first inequality. In what follows we use $\nu_1(a)$ (resp. $\nu_2(a)$) to denote the expression $M_1(a)(X_0(a))$ (resp. $M_2(a)(X_0(a))$).

$$\begin{aligned}
\theta_1(X_0) - \alpha\alpha'\ \theta_2(X_0) &= \sum_{(a,b)\in X_0} \mu_1(a)\ M_1(a)(b) - \alpha\alpha'\ \mu_2(a)\ M_2(a)(b) \\
&\stackrel{(1)}{=} \sum_{a\in\pi_1(X_0)} \sum_{b\in X_0(a)} \mu_1(a)\ M_1(a)(b) - \alpha\alpha'\mu_2(a)\ M_2(a)(b) \\
&= \sum_{a\in\pi_1(X_0)} \mu_1(a)\ \nu_1(a) - \alpha\alpha'\mu_2(a)\ \nu_2(a) \\
&\stackrel{(2)}{\leq} \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)>1}} \mu_1(a) - \alpha\mu_2(a) \ + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1}} \mu_1(a)\big(\alpha'\nu_2(a) + \delta'\big) - \alpha\alpha'\mu_2(a)\ \nu_2(a) \\
&= \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)>1}} \mu_1(a) - \alpha\mu_2(a) + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1}} \mu_1(a)\delta' + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1}} \alpha'\nu_2(a)\big(\mu_1(a) - \alpha\mu_2(a)\big) \\
&\leq \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)>1}} \mu_1(a) - \alpha\mu_2(a) + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1}} \mu_1(a)\delta' + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1\\ \mu_1(a)\geq\alpha\mu_2(a)}} \mu_1(a) - \alpha\mu_2(a)
\end{aligned}$$

Equality (1) holds by a simple reordering of the terms in the sum; to justify (2) we rely on the fact that the expression $\mu_1(a)\ \nu_1(a) - \alpha\alpha'\mu_2(a)\ \nu_2(a)$ can be bounded by $\mu_1(a) - \alpha\mu_2(a)$ when $\alpha'\nu_2(a) > 1$ and on hypothesis $\forall a.\ \Delta_{\alpha'}(M_1\ a, M_2\ a) \leq \delta'$ to bound

$\nu_1(a)$ by $\alpha'\nu_2(a)+\delta'$. (Observe that inequality (2) remains valid in the variant of the lemma where we have the weaker hypothesis $\forall a.\ R\ a \implies \Delta_{\alpha'}(M_1\ a, M_2\ a) \leq \delta'$. This is because for inequality (2) to hold it suffices to bound $\nu_1(a)$ by $\alpha'\nu_2(a)+\delta'$ only when $\mu_1(a) > 0$.) By letting $Y_1 \stackrel{\text{def}}{=} \{a \in \pi_1(X_0) \mid \alpha'\nu_2(a) \leq 1 \implies \mu_1(a) \geq \alpha\mu_2(a)\}$ and $Y_2 \stackrel{\text{def}}{=} \{a \in \pi_1(X_0) \mid \alpha'\nu_2(a) \leq 1\}$ we have

$$\theta_1(X_0) - \alpha\alpha'\ \theta_2(X_0) \leq \mu_1(Y_1) - \alpha\mu_2(Y_1) + \mu_1(Y_2)\delta' \leq \delta + \delta'$$

We follow a similar reasoning to prove $\theta_2(X_1) - \alpha\alpha'\ \theta_1(X_1) \leq \delta + \delta'$ and conclude. □

PROPOSITION A.6. *Let A and B be two (non-empty) discrete sets. Then, for any pair of distributions $\mu_1 \in \mathcal{D}(A)$ and $\mu \in \mathcal{D}(A \times B)$, there exists a distribution $\mu' \in \mathcal{D}(A \times B)$ that satisfies:*

$$\pi_1\mu' = \mu_1 \qquad \textit{and} \qquad \Delta_\alpha(\mu, \mu') \leq \Delta_\alpha(\pi_1\mu, \mu_1)$$

If, moreover, $R \subseteq A \times B$ is a full relation and range $R\ \mu$ *holds, then* range $R\ \mu'$ *also holds.*

PROOF. Let g be some map from A to B (the existence of such a map is guaranteed as B is non-empty). Define μ' as follows:

$$\mu'(a,b) \stackrel{\text{def}}{=} \begin{cases} \dfrac{\mu_1(a)\ \mu(a,b)}{(\pi_1\mu)(a)} & \text{if } 0 < (\pi_1\mu)(a) \\ \mu_1(a)\ \mathbb{1}_{\{g(a)\}}(b) & \text{otherwise} \end{cases}$$

The proof of equality $\pi_1\mu' = \mu_1$ is immediate by doing a case analysis on whether $\pi_1\mu$ is null at a or not. In view of Lemma A.3, in order to prove that $\Delta_\alpha(\mu,\mu') \leq \Delta_\alpha(\pi_1\mu, \mu_1)$ it suffices to show inequalities

$$\mu'(X_0) - \alpha\ \mu(X_0) \leq \Delta_\alpha(\pi_1\mu, \mu_1) \qquad \text{and} \qquad \mu(X_1) - \alpha\ \mu'(X_1) \leq \Delta_\alpha(\pi_1\mu, \mu_1)$$

where $X_0 \stackrel{\text{def}}{=} \{(a,b) \mid \mu'(a,b) \geq \alpha\ \mu(a,b)\}$ and $X_1 \stackrel{\text{def}}{=} \{(a,b) \mid \mu(a,b) \geq \alpha\ \mu'(a,b)\}$. To prove the first inequality we consider the pair of sets $X_0^0 \stackrel{\text{def}}{=} \{(a,b) \in X_0 \mid 0 < (\pi_1\mu)(a)\}$ and $X_0^1 \stackrel{\text{def}}{=} \{(a,b) \in X_0 \mid (\pi_1\mu)(a) = 0\}$ and observe that

$$\begin{aligned}
\mu'(X_0^0) - \alpha\mu(X_0^0) &= \sum_{(a,b)\in X_0^0} \big(\mu_1(a) - \alpha(\pi_1\mu)(a)\big)\ \frac{\mu(a,b)}{(\pi_1\mu)(a)} \\
&= \sum_{a \in \pi_1(X_0^0)} \big(\mu_1(a) - \alpha(\pi_1\mu)(a)\big) \sum_{b \in X_0^0(a)} \frac{\mu(a,b)}{(\pi_1\mu)(a)} \\
&\leq \sum_{\substack{a\in\pi_1(X_0^0) \\ \mu_1(a) \geq \alpha\pi_1\mu(a)}} \mu_1(a) - \alpha(\pi_1\mu)(a)
\end{aligned}$$

$$\begin{aligned}
\mu'(X_0^1) - \alpha\mu(X_0^1) &= \sum_{(a,b)\in X_0^1} \mu_1(a)\ \mathbb{1}_{\{g(a)\}}(b) - \alpha\mu(a,b) \\
&= \sum_{a\in\pi_1(X_0^1)} \mu_1(a) \sum_{b\in X_0^1(a)} \mathbb{1}_{\{g(a)\}}(b) - \sum_{a\in\pi_1(X_0^1)} \alpha \sum_{b \in X_0^1(a)} \mu(a,b) \\
&\stackrel{(1)}{\leq} \sum_{a\in\pi_1(X_0^1)} \mu_1(a) - \alpha(\pi_1\mu)(a)\ \leq \sum_{\substack{a\in\pi_1(X_0^1) \\ \mu_1(a)\geq\alpha(\pi_1\mu)(a)}} \mu_1(a) - \alpha(\pi_1\mu)(a)
\end{aligned}$$

Inequality (1) holds since for every a in $\pi_1(X_0^1)$, we have $\sum_{b\in X_1(a)} \mathbb{1}_{\{g(a)\}}(b) \leq 1$ and $\sum_{b\in X_0^1(a)} \mu(a,b) = (\pi_1\mu)(a) = 0$. Combining the two inequalities above we get:

$$\begin{aligned}\mu'(X_0) - \alpha\,\mu(X_0) &= \mu'(X_0^0) - \alpha\mu(X_0^0) + \mu'(X_0^1) - \alpha\mu(X_0^1)\\ &\leq \sum_{\substack{a\in\pi_1(X_0)\\ \mu_1(a)\geq\alpha\pi_1\mu(a)}} \mu_1(a) - \alpha(\pi_1\mu)(a) \leq \Delta_\alpha(\pi_1\mu,\mu_1)\end{aligned}$$

To prove inequality $\mu(X_1) - \alpha\,\mu(X_1)' \leq \Delta_\alpha(\pi_1\mu,\mu_1)$, we split the set X_1 into X_1^0 and X_1^1 as done with X_0 and show that $\mu(X_1^0) - \alpha\mu'(X_1^0) \leq \Delta_\alpha(\pi_1\mu,\mu_1)$ and $\mu(X_1^1) - \alpha\mu'(X_1^1) = 0$.

Assume that R is full and that we have $\mathsf{range}\ R\ \mu$. When defining μ' let us choose map g such that $a\ R\ g(a)$ for every $a \in A$. We derive $\mathsf{range}\ R\ \mu'$ from Lemma A.2. Assume $\mu'(a,b) > 0$. Then, from the definition of μ' we have either $\mu_1(a)\mu(a,b) > 0$ or $b = g(a)$. In the first case, we have $a\ R\ b$ from hypothesis $\mathsf{range}\ R\ \mu$; in the second case, $a\ R\ b$ follows from the definition of g. Then, $\mathsf{range}\ R\ \mu'$ follows. □

LEMMA A.7. *For any relation $R \subseteq A \times B$, $a\ R\ b \implies (\mathsf{unit}\ a) \sim_R^{1,0} (\mathsf{unit}\ b)$*

PROOF. The proof is immediate by considering $(\mathsf{unit}\ a) \times (\mathsf{unit}\ b)$ as witness distribution for the lifting $(\mathsf{unit}\ a) \sim_R^{1,0} (\mathsf{unit}\ b)$. □

B. PROOFS

PROOF OF LEMMA 4.2. The inequality $\max_{E\subseteq A}\ \Delta_\alpha(\mu_1(E),\mu_2(E)) \leq \Delta_\alpha(\mu_1,\mu_2)$ follows from the definition of α-distance between distributions while its converse is a direct consequence of Lemma A.3. □

PROOF OF THEOREM 4.10. Let μ be a witness for the lifting $\mu_1 \sim_R^{\alpha,\delta} \mu_2$. Then,

$$\mu_1\ f_1 - \alpha\mu_2\ f_2 \overset{(1)}{\leq} \mu_1\ f_1 - \alpha(\pi_2\ \mu)\ f_2 \overset{(2)}{=} \mu_1\ f_1 - \alpha(\pi_1\ \mu)\ f_1 \overset{(3)}{\leq} \delta$$

From the definition of μ we have $\pi_2\ \mu \leq \mu_2$ and $\Delta_\alpha(\mu_1,\pi_1\ \mu) \leq \delta$, which imply (1) and (3), respectively. We also have $\mathsf{range}\ R\ \mu$, which combined with Lemma A.1.a and hypothesis $f_1 =_R f_2$ entails formula $(\pi_2\ \mu)\ f_2 = (\pi_1\ \mu)\ f_1$ and shows equality (2). □

In what follows we will rely extensively on the fact that for any distribution μ over the product of two discrete sets A and B, we can compute the probability mass function of its projection $(\pi_1\mu)(a)$ (resp. $(\pi_2\mu)(b)$) as $\sum_{b\in B}\mu(a,b)$ (resp. $\sum_{a\in A}\mu(a,b)$).

PROOF OF THEOREM 4.12. Recall that the probability mass function of the proposed witness is

$$\mu(a,c) = \sum_{b\in B,\ 0<\mu_2(b)} \frac{\mu_R(a,b)\ \mu_S(b,c)}{\mu_2(b)}$$

where μ_R and μ_S are witnesses for the liftings $\mu_1 \sim_R^{\alpha,\delta} \mu_2$ and $\mu_2 \sim_S^{\alpha,\delta'} \mu_3$, respectively.

FACT 1. $\pi_1\mu \leq \pi_1\mu_R$.

PROOF. Immediate from the reasoning below:

$$\begin{aligned}(\pi_1\mu)(a) &\overset{(1)}{=} \sum_{c\in C}\sum_{b\in B\ \mid\ 0<\mu_2(b)} \frac{\mu_R(a,b)\mu_S(b,c)}{\mu_2(b)} \overset{(2)}{=} \sum_{b\in B\ \mid\ 0<\mu_2(b)} \mu_R(a,b)\frac{\sum_{c\in C}\mu_S(b,c)}{\mu_2(b)}\\ &\overset{(3)}{\leq} \sum_{b\in B\ \mid\ 0<\mu_2(b)} \mu_R(a,b) \leq (\pi_1\ \mu_R)(a)\end{aligned}$$

Equalities (1) and (2) are an unfolding of μ and a reordering of the series respectively, while inequality (3) is a consequence of hypothesis $\pi_1\mu_S \le \mu_2$. $\square$

FACT 2. $\Delta_\beta(\pi_1\mu, \pi_1\mu_R) \le \Delta_\beta(\pi_1\mu_S, \mu_2)$ *for all* $\beta > 1$.

PROOF. In view of Fact 1, we can use Lemma A.4 to compute $\Delta_\beta(\pi_1\mu, \pi_1\mu_R)$. Let $A_0 = \{a \in A \mid (\pi_1\mu)(a) \ge \beta(\pi_1\mu_R)(a)\}$; the proof proceed as follows:

$$\begin{aligned}
\Delta_\beta(\pi_1\mu, \pi_1\mu_R) &= (\pi_1\mu_R)(A_0) - \beta(\pi_1\mu)(A_0) \\
&= \sum_{b\in B} \mu_R(A_0, b) - \beta \sum_{c\in C} \sum_{b\in B \mid 0<\mu_2(b)} \frac{\mu_R(A_0,b)\ \mu_S(b,c)}{\mu_2(b)} \\
&\overset{(1)}{=} \sum_{b\in B} \mu_R(A_0, b) - \beta \sum_{b\in B \mid 0<\mu_2(b)} \frac{\mu_R(A_0,b)\ (\pi_1\mu_S)(b)}{\mu_2(b)} \\
&\overset{(2)}{=} \sum_{b\in B \mid 0<\mu_2(b)} \mu_R(A_0, b) - \beta \sum_{b\in B \mid 0<\mu_2(b)} \frac{\mu_R(A_0,b)\ (\pi_1\mu_S)(b)}{\mu_2(b)} \\
&\le \sum_{b\in B \mid \beta(\pi_1\mu_S)(b)<\mu_2(b)} \frac{\mu_R(A_0,b)}{\mu_2(b)} \big(\mu_2(b) - \beta(\pi_1\mu_S)(b)\big) \\
&\overset{(3)}{\le} \sum_{b\in B \mid \beta(\pi_1\mu_S)(b)<\mu_2(b)} \mu_2(b) - \beta(\pi_1\mu_S)(b) \overset{(4)}{=} \Delta_\beta(\pi_1\mu_S, \mu_2)
\end{aligned}$$

In (1) we perform a series reordering; step (2) is valid as for each $b \in B$, on account of inequality $0 \le \mu_R(A_0, b) \le (\pi_2\mu_R)(b) \le \mu_2(b)$, we have $\mu_2(b) = 0 \implies \mu_R(A_0, b) = 0$; the same argument allows bounding the factors $\mu_R(A_0, b)/\mu_2(b)$ by 1 in (3); finally equality (4) is justified by Lemma A.4 as, by hypothesis, $\pi_1\mu_S \le \mu_2$. $\square$

We now turn to the proof of the main claim. We have to check the three conditions that μ should satisfy to be a witness for the lifting

$$\mu_1 \sim_{R\circ S}^{\alpha\ \alpha', \max(\delta+\alpha\ \delta', \delta'+\alpha'\ \delta)} \mu_3$$

For the sake of brevity we only show how to conclude that $\pi_1\mu \le \mu_1$ and $\Delta_{\alpha\alpha'}(\pi_1\mu, \mu_1) \le \max(\delta + \alpha\ \delta', \delta' + \alpha'\ \delta)$; the proofs of the inequalities $\pi_2\mu \le \mu_3$ and $\Delta_{\alpha\alpha'}(\pi_2\mu, \mu_3) \le \max(\delta + \alpha\ \delta', \delta' + \alpha'\ \delta)$ are analogous. We use Lemma A.2 to derive condition range $(R \circ S)\ \mu$. Let (a, c) be such that $\mu(a, c) > 0$. From the definition of μ it follows that there exists b such that $\mu_R(a, b) > 0$ and $\mu_S(b, c) > 0$. Thus, from the same lemma applied twice (but in the converse direction as before), we derive $a\ (R \circ S)\ c$ and hence, range $(R \circ S)\ \mu$. To prove that $\pi_1\mu$ is dominated by μ_1 we apply transitivity with $\pi_1\ \mu_R$. The inequality $\pi_1\ \mu \le \pi_1\ \mu_R$ holds on account of Fact 1, while inequality $\pi_1\ \mu_R \le \mu_1$ follows from μ_R being a witness for the lifting $\mu_1 \sim_R^{\alpha,\delta} \mu_2$. Finally, the bound on distance $\Delta_{\alpha\alpha'}(\pi_1\ \mu, \mu_1)$ is proved by transitivity with $\pi_1\mu_R$ as follows:

$$\begin{aligned}
\Delta_{\alpha\alpha'}(\pi_1\mu, \mu_1) &\overset{(1)}{\le} \max\begin{pmatrix}\alpha\Delta_{\alpha'}(\pi_1\mu, \pi_1\mu_R) + \Delta_\alpha(\pi_1\mu_R, \mu_1),\\ \alpha'\Delta_\alpha(\pi_1\mu_R, \mu_1) + \Delta_{\alpha'}(\pi_1\mu, \pi_1\mu_R)\end{pmatrix} \\
&\overset{(2)}{\le} \max\begin{pmatrix}\alpha\Delta_{\alpha'}(\pi_1\mu_S, \mu_2) + \Delta_\alpha(\pi_1\mu_R, \mu_1),\\ \alpha'\Delta_\alpha(\pi_1\mu_R, \mu_1) + \Delta_{\alpha'}(\pi_1\mu_S, \mu_2)\end{pmatrix} \overset{(3)}{\le} \max\left(\alpha\delta' + \delta, \alpha'\delta + \delta'\right)
\end{aligned}$$

Here (1) constitutes an instance of Lemma 4.3.4, (2) follows from Fact 2, and (3) follows from μ_R and μ_S being witnesses for liftings $\mu_1 \sim_R^{\alpha,\delta} \mu_2$ and $\mu_2 \sim_S^{\alpha',\delta'} \mu_3$ respectively.

PROOF OF LEMMMA 4.13. We first sketch a proof when $M_1\ a\ \mathbb{1}_{A'} = M_2\ b\ \mathbb{1}_{B'} = 1$ for all a, b. Let $\mu \in \mathcal{D}(A \times B)$ be a witness for $\mu_1 \sim_R^{\alpha,\delta} \mu_2$ and let $M : A \times B \to \mathcal{D}(A' \times B')$ map R-related values a, b to a witness distribution of the lifting $(M_1\ a) \sim_{R'}^{\alpha',\delta'} (M_2\ b)$ and non R-related values a, b to the product distribution $(M_1\ a) \times (M_2\ b)$. Hence,

i) $\mathsf{range}\ R\ \mu$,
ii) $\pi_1\ \mu \leq \mu_1\ \wedge\ \pi_2\ \mu \leq \mu_2$,
iii) $\Delta_\alpha(\pi_1\ \mu, \mu_1) \leq \delta\ \wedge\ \Delta_\alpha(\pi_2\ \mu, \mu_2) \leq \delta$,
iv) $a\ R\ b \implies \mathsf{range}\ R'\ M(a, b)$,
v) $\pi_1\ (M(a, b)) \leq M_1\ a\ \wedge\ \pi_2\ (M(a, b)) \leq M_2\ b$, and
vi) $\Delta_{\alpha'}(\pi_1\ (M(a, b)), M_1\ a) \leq \delta' \wedge \Delta_{\alpha'}(\pi_2\ (M(a, b)), M_2\ b) \leq \delta'$.

Hypothesis $M_1\ a\ \mathbb{1}_{A'} = M_2\ b\ \mathbb{1}_{B'} = 1$ is fundamental to show the validity of vi) when a and b are not related by R. For such values, it guarantees that $\pi_1\ (M(a, b)) = M_1\ a$ and $\pi_2\ (M(a, b)) = M_2\ b$, and therefore

$$\Delta_{\alpha'}(\pi_1\ (M(a, b)), M_1\ a) = \Delta_{\alpha'}(\pi_2\ (M(a, b)), M_2\ b) = 0 \leq \delta'$$

We claim that $\mathsf{bind}\ \mu\ M$ is witness of the lifting $(\mathsf{bind}\ \mu_1\ M_1) \sim_{R'}^{\alpha\alpha',\delta+\delta'} (\mathsf{bind}\ \mu_2\ M_2)$. The condition $\mathsf{range}\ R'\ (\mathsf{bind}\ \mu\ M)$ follows from Lemma A.1.b and properties i) and iv), whereas condition $\pi_1(\mathsf{bind}\ \mu\ M) \leq \mathsf{bind}\ \mu_1\ M_1$ can be shown by applying transitivity with $\mathsf{bind}\ (\pi_1\ \mu)\ M_1$; inequality $\pi_1(\mathsf{bind}\ \mu\ M) \leq \mathsf{bind}\ (\pi_1\ \mu)\ M_1$ follows from property v), whereas inequality $\mathsf{bind}\ (\pi_1\ \mu)\ M_1 \leq \mathsf{bind}\ \mu_1\ M_1$ follows from the monotonicity of the bind operator and property ii). Condition $\pi_2(\mathsf{bind}\ \mu\ M) \leq \mathsf{bind}\ \mu_2\ M_2$ is proved analogously, by applying transitivity with distribution $\mathsf{bind}\ (\pi_2\ \mu)\ M_2$.

Finally, we prove condition $\Delta_{\alpha\alpha'}(\pi_1\ (\mathsf{bind}\ \mu\ M), \mathsf{bind}\ \mu_1\ M_1) \leq \delta + \delta'$ using Proposition A.6. A direct application of this proposition and properties iii) and vi) above, implies there exists a distribution $\mu' \in \mathcal{D}(A \times B)$ and $M' : A \times B \to \mathcal{D}(A' \times B')$ s.t.

vii) $\pi_1\ \mu' = \mu_1$,
viii) $\Delta_\alpha(\mu, \mu') \leq \delta$,
ix) $\pi_1\ (M'(a, b)) = M_1\ a$, and
x) $\Delta_{\alpha'}(M(a, b), M'(a, b)) \leq \delta'$

Hence,

$$\begin{aligned}\Delta_{\alpha\alpha'}(\pi_1\ (\mathsf{bind}\ \mu\ M), \mathsf{bind}\ \mu_1\ M_1) &\overset{(1)}{=} \Delta_{\alpha\alpha'}(\pi_1\ (\mathsf{bind}\ \mu\ M), \pi_1\ (\mathsf{bind}\ \mu'\ M')) \\ &\overset{(2)}{\leq} \Delta_{\alpha\alpha'}(\mathsf{bind}\ \mu\ M, \mathsf{bind}\ \mu'\ M')\ \overset{(3)}{\leq}\ \delta + \delta'\end{aligned}$$

Equality (1) holds because combining vii) and ix) one gets $\mathsf{bind}\ \mu_1\ M_1 = \pi_1\ (\mathsf{bind}\ \mu'\ M')$; inequality (2) is a direct application of Lemma 4.3.6 while inequality (3) can be justified by Lemma A.5 and hypotheses viii) and x). The remaining inequality $\Delta_{\alpha\alpha'}(\pi_2\ (\mathsf{bind}\ \mu\ M), \mathsf{bind}\ \mu_2\ M_2) \leq \delta + \delta'$ is shown analogously.

The proof proceeds similarly when R is full. The major difference between is that now the validity of propositions vi) and x) can be guaranteed only when $a\ R\ b$ holds. This only affects the argument that we use to justify inequality

$$\Delta_{\alpha\alpha'}(\mathsf{bind}\ \mu\ M, \mathsf{bind}\ \mu'\ M') \leq \delta + \delta'$$

To justify it, we use the variant of Lemma A.5 that requires that the inequality $\Delta_{\alpha'}(M(a, b), M'(a, b)) \leq \delta'$ holds only when $a\ R\ b$. This variant has as additional hypotheses $\mathsf{range}\ R\ \mu$ and $\mathsf{range}\ R\ \mu'$. The first follows from i), while the second follows from Proposition A.6. $\square$

PROOF OF LEMMA 4.8. For the "only if" direction we use Lemma A.3 to bound $\Delta_\alpha(\mu_1/R, \mu_2/R)$. We thus introduce sets $A_0 \stackrel{\text{def}}{=} \{S \in A/R \mid (\mu_1/R)(S) \geq \alpha(\mu_2/R)(S)\}$ and $A_1 \stackrel{\text{def}}{=} \{S \in A/R \mid (\mu_2/R)(S) \geq \alpha(\mu_1/R)(S)\}$. We have now to show that δ is an upper bound of both $(\mu_1/R)(A_0) - \alpha(\mu_2/R)(A_0)$ and $(\mu_2/R)(A_1) - \alpha(\mu_1/R)(A_1)$. Let μ be a witness for the lifting $\mu_1 \sim_R^{\alpha,\delta} \mu_2$. Then,

$$
\begin{aligned}
(\mu_1/R)(A_0) - \alpha(\mu_2/R)(A_0) &= \sum_{S \in A_0} \mu_1(S) - \alpha\mu_2(S) \\
&= \sum_{S \in A_0} \mu_1(S) - \alpha(\pi_1\ \mu)(S) + \alpha(\pi_1\ \mu)(S) - \alpha\mu_2(S) \\
&\stackrel{(1)}{=} \sum_{S \in A_0} \mu_1(S) - \alpha(\pi_1\ \mu)(S) + \alpha(\pi_2\ \mu)(S) - \alpha\mu_2(S) \\
&\stackrel{(2)}{\leq} \sum_{S \in A_0} \mu_1(S) - \alpha(\pi_1\ \mu)(S) = \mu_1(A_0) - \alpha(\pi_1\ \mu)(A_0) \leq \delta
\end{aligned}
$$

The validity of (1) amounts to showing that $(\pi_1\mu)(S) = (\pi_2\mu)(S)$ for all $S \in A_0$. This equality can be restated as $\mu\ f_1 = \mu\ f_2$, where $f_1(a_1, a_2) = \mathbb{1}_{a_1 \in S}$ and $f_2(a_1, a_2) = \mathbb{1}_{a_2 \in S}$. By Lemma A.1.a this reduces in turn to verifying that f_1 and f_2 are R-equivalent, which is immediate. Finally, inequality (2) is a direct consequence of condition $\pi_2\ \mu \leq \mu_2$. To show that $(\mu_2/R)(A_1) - \alpha(\mu_1/R)(A_1) \leq \delta$ we follow a similar reasoning.

For the "if" direction, we propose

$$
\mu(a_1, a_2) \stackrel{\text{def}}{=} \begin{cases} \dfrac{\mu_1(a_1)\mu_2(a_2)}{\tilde{\mu}([a_1])} & \text{if } a_1 R\, a_2 \ \wedge\ 0 < \tilde{\mu}([a_1]) \\ 0 & \text{otherwise} \end{cases}
$$

as a witness for the lifting $\mu_1 \sim_R^{\alpha,\delta} \mu_2$, where $\tilde{\mu}([a]) = \max\{\mu_1([a]), \mu_2([a])\}$.

We next verify the three conditions that μ must satisfy. Lemma A.2 readily entails range $R\ \mu$. Computing the first and second projections of μ gives:

$$
(\pi_1\mu)(a) = \begin{cases} \mu_1(a)\dfrac{\mu_2([a])}{\tilde{\mu}([a])} & \text{if } 0 < \tilde{\mu}([a]) \\ 0 & \text{otherwise} \end{cases} \qquad (\pi_2\mu)(a) = \begin{cases} \mu_2(a)\dfrac{\mu_1([a])}{\tilde{\mu}([a])} & \text{if } 0 < \tilde{\mu}([a]) \\ 0 & \text{otherwise} \end{cases}
$$

from which one can observe that $\pi_1\mu \leq \mu_1$ and $\pi_2\mu \leq \mu_2$. We can then use Lemma A.3 to bound $\Delta_\alpha(\pi_1\mu, \mu_1)$. It yields equality $\Delta_\alpha(\pi_1\ \mu, \mu_1) = \mu_1(A_1) - \alpha(\pi_1\ \mu)(A_1)$, where $A_1 \stackrel{\text{def}}{=} \{a \in A \mid \mu_1(a) \geq \alpha(\pi_1\ \mu)(a)\}$. We now have

$$
\begin{aligned}
\Delta_\alpha(\pi_1\mu, \mu_1) = \sum_{a \in A_1} \mu_1(a) - \alpha(\pi_1\mu)(a) &\stackrel{(1)}{=} \sum_{\substack{a \in A_1 \\ 0 < \tilde{\mu}([a])}} \mu_1(a) - \alpha\mu_1(a)\frac{\mu_2([a])}{\tilde{\mu}([a])} \\
&\stackrel{(2)}{=} \sum_{\substack{a \in A_1 \\ 0 < \mu_1([a]) \\ \alpha\mu_2([a]) \leq \mu_1([a])}} \frac{\mu_1(a)}{\mu_1([a])}\big(\mu_1([a]) - \alpha\mu_2([a])\big) \ + \sum_{\substack{a \in A_1 \\ 0 < \tilde{\mu}([a]) \\ \mu_1([a]) < \alpha\mu_2([a])}} \mu_1(a) - \alpha\mu_1(a)\frac{\mu_2([a])}{\tilde{\mu}([a])} \\
&\stackrel{(3)}{\leq} \sum_{\substack{a \in A_1 \\ 0 < \mu_1([a]) \\ \alpha\mu_2([a]) \leq \mu_1([a])}} \frac{\mu_1(a)}{\mu_1([a])}\big(\mu_1([a]) - \alpha\mu_2([a])\big)
\end{aligned}
$$

$$
\begin{aligned}
&\overset{(4)}{=} \sum_{[\cdot]\in A/R} \sum_{\substack{a\in[\cdot]\cap A_1\\ \mu_1([\cdot])\geq\alpha\mu_2([\cdot])}} \frac{\mu_1(a)}{\mu_1([\cdot])}\big(\mu_1([\cdot]) - \alpha\mu_2([\cdot])\big)\\
&= \sum_{\substack{[\cdot]\in A/R\\ \mu_1([\cdot])\geq\alpha\mu_2([\cdot])}} \big(\mu_1([\cdot]) - \alpha\mu_2([\cdot])\big) \sum_{\substack{a\in[\cdot]\cap A_1\\ 0<\mu_1([\cdot])}} \frac{\mu_1(a)}{\mu_1([\cdot])}\\
&\overset{(5)}{\leq} \sum_{\substack{[\cdot]\in A/R\\ \mu_1([\cdot])\geq\alpha\mu_2([\cdot])}} \mu_1([\cdot]) - \alpha\mu_2([\cdot]) = (\mu_1/R)(X) - \alpha(\mu_2/R)(X)\\
&\qquad\qquad\qquad\qquad\text{where } X = \{[\cdot] \in A/R \mid \mu_1([\cdot]) \geq \alpha\mu_2([\cdot])\}\\
&\leq \Delta_\alpha(\mu_1/R, \mu_2/R) \leq \delta
\end{aligned}
$$

In (1) we unfold the above computed $\pi_1\mu$ and use the fact that $\mu_1(a) = 0$ when $\tilde{\mu}([a]) = 0$. In (2) we reorder terms and substitute $\mu_1([a])$ for $\tilde{\mu}([a])$ when $\mu_1([a]) \geq \alpha\mu_2([a])$. Inequality (3) is valid because for each a such that $\mu_1([a]) < \alpha\mu_2([a])$, $\mu_1(a) \leq \alpha\mu_1(a)\frac{\mu_2([a])}{\tilde{\mu}([a])}$. To see this observe that in case $\mu_2([a]) \leq \mu_1([a])$, the term equals $\frac{\mu_1(a)}{\mu_1([a])}\big(\mu_1([a]) - \alpha\mu_2([a])\big)$, whereas if $\mu_2([a]) > \mu_1([a])$, it simplifies to $(1-\alpha)\mu_1(a)$. In (4) we reorder the series in order to group all terms $\frac{\mu_1(a)}{\mu_1([a])}\big(\mu_1([a]) - \alpha\mu_2([a])\big)$ with a in the same equivalence class. Finally, inequality (5) holds because for every $[\cdot] \in A/R$, we have $\sum_{\substack{a\in[\cdot]\cap A_1\\ 0<\mu_1([\cdot])}} \frac{\mu_1(a)}{\mu_1([\cdot])} \leq 1$. The inequality $\Delta_\alpha(\pi_2\ \mu, \mu_2) \leq \delta$ is proved analogously. □

PROOF OF LEMMA 6.1. Let T be a subset of B. The reasoning below shows that $f^\sharp\ a_1\ T - \alpha\alpha' f^\sharp\ a_2\ T \leq f^\sharp\ a_1\ S_1$.

$$
\begin{aligned}
f^\sharp\ a_1\ T &= \frac{\sum_{b\in T\cap\overline{S_1}} f\ a_1\ b}{\sum_{b\in B} f\ a_1\ b} + \frac{\sum_{b\in T\cap S_1} f\ a_1\ b}{\sum_{b\in B} f\ a_1\ b} \leq \alpha' \frac{\sum_{b\in T\cap\overline{S_1}} f\ a_2\ b}{\sum_{b\in B} f\ a_1\ b} + \frac{\sum_{b\in T\cap S_1} f\ a_1\ b}{\sum_{b\in B} f\ a_1\ b}\\
&\leq \alpha\alpha' \frac{\sum_{b\in T\cap\overline{S_1}} f\ a_2\ b}{\sum_{b\in B} f\ a_2\ b} + \frac{\sum_{b\in T\cap S_1} f\ a_1\ b}{\sum_{b\in B} f\ a_1\ b} \leq \alpha\alpha' f^\sharp\ a_2\ T + f^\sharp\ a_1\ S_1
\end{aligned}
$$

Similarly, we can show that $f^\sharp\ a_2\ T - \alpha\alpha' f^\sharp\ a_1\ T \leq f^\sharp\ a_2\ S_2$. The final result follows from Lemma A.3. □

PROOF OF COROLLARY 6.2. From Lemma 6.1 with $\alpha' = \alpha$. Observe that hypothesis

$$\forall b \in B,\ f(a_1, b) \leq \alpha\ f(a_2, b) \wedge f(a_2, b) \leq \alpha\ f(a_1, b)$$

implies $S_1 = S_2 = \emptyset$. Hence, $f^\sharp\ a_1\ \mathbb{1}_{S_1} = f^\sharp\ a_2\ \mathbb{1}_{S_2} = 0$, and thus $\Delta_{\alpha^2}(f^\sharp\ a_1, f^\sharp\ a_2) = 0$. □

PROOF OF SOUNDNESS OF RULE [EXP]. Applying rule [rand], we are left to prove that for any pair of memories m_1, m_2 s.t. $m_1\ \Psi\ m_2$,

$$\Delta_{\exp(k\mathbf{S}_s\epsilon)^2}(\mathcal{E}^\epsilon_{s,\mu}([\![a]\!]\ m_1), \mathcal{E}^\epsilon_{s,\mu}([\![a]\!]\ m_2)) \leq 0 \tag{5}$$

Let $f\ a\ b = \exp(\epsilon\ s(a, b)\ \mu(b))$, $\gamma = \exp(k\mathbf{S}_s\epsilon)$, $a_1 = [\![a]\!]\ m_1$, and $a_2 = [\![a]\!]\ m_2$.

From the first premise of rule [exp] we have $d(a_1, a_2) \leq k$, and hence for all $b \in B$, $s(a_1, b) - s(a_2, b) \leq k\mathbf{S}_s$. Moreover,

$$
\begin{aligned}
\mu(b)k\mathbf{S}_s\epsilon \leq k\mathbf{S}_s\epsilon &\implies \exp(\mu(b)k\mathbf{S}_s\epsilon) \leq \gamma\\
&\implies \exp(\mu(b)(s(a_1, b) - s(a_2, b))\epsilon) \leq \gamma\\
&\implies f(a_1, b) \leq \gamma f(a_2, b)
\end{aligned}
$$

Hence, for all $b \in B$, $f(a_1, b) \leq \gamma f(a_2, b)$, and analogously $f(a_2, b) \leq \gamma f(a_1, b)$. Observe that (5) is equivalent to $\Delta_{\gamma^2}(f^\sharp\ a_1, f^\sharp\ a_2) \leq 0$, which follows from Lemma 6.2. □

PROOF OF SOUNDNESS OF RULE [NORM]. Applying rule [rand], we are left to prove that for every pair of memories m_1, m_2 s.t. $m_1\ \Psi\ m_2$,

$$\Delta_{\exp(\epsilon)}(\mathcal{N}(\llbracket r \rrbracket\ m_1, \sigma), \mathcal{N}(\llbracket r \rrbracket\ m_2, \sigma)) \leq \mathcal{B}\left(\sigma, \frac{1}{2}(\sigma\epsilon - k^2)/k\right) \leq \delta \tag{6}$$

We prove (6) by applying Lemma 6.1 with $A = B = \mathbb{Z}$, $f\ a\ b = \exp\left(-|b-a|^2/\sigma\right)$, $a_1 = \llbracket r \rrbracket\ m_1$, $a_2 = \llbracket r \rrbracket\ m_2$, $\alpha = 1$ and $\alpha' = \exp(\epsilon)$. We conclude by showing inequality

$$\max\{f^\sharp\ a_1\ \mathbb{1}_{S_1}, f^\sharp\ a_2\ \mathbb{1}_{S_2}\} \leq \mathcal{B}\left(\sigma, \frac{1}{2}(\sigma\epsilon - k^2)/k\right) \tag{7}$$

(Here S_1 and S_2 are defined as in the statement of Lemma 6.1.) The reader can verify that (7) is entailed by $|a_1 - a_2| \leq k$, which follows from the first premise of the rule. □

PROOF OF THEOREM 7.1. Direction $(2) \implies (1)$ for $\alpha = 1$ follows from [Desharnais et al. 2008, Theorem 7]. We sketch a proof of the converse. Let f be a feasible flow in $\mathcal{N}(\mu_1, \mu_2, R, \alpha)$ with $f_\perp \geq \mu_1(A) - \delta$ and $f_\top \geq \mu_2(B) - \delta$. Observe first that from the flow conservation constraints at vertices $a \in A$ and $b \in B$, we have

$$f(\perp, a) = \alpha \sum_{b \in R(a)} f(a, b) \qquad\qquad f(b, \top) = \alpha \sum_{a \in R^{-1}(b)} f(a, b) \tag{8}$$

We propose as a witness for the lifting $\mu_1 \sim_R^{\alpha,\delta} \mu_2$, the distribution $\mu \in \mathcal{D}(A \times B)$ with probability mass function

$$\mu(a, b) = \begin{cases} f(a, b) & \text{if } a\ R\ b \\ 0 & \text{otherwise} \end{cases}$$

Clearly $\mu(a, b) \geq 0$ for all $(a, b) \in A \times B$ and a simple computation using (8) yields $\mu(A, B) \leq \alpha^{-1} \leq 1$; thus μ is a proper sub-probability distribution over $A \times B$. We now proceed to verify that μ is a witness for the lifting. Property range $R\ \mu$ is immediate by the definition of μ. Inequality $\pi_1 \mu \leq \mu_1$ follows from (8):

$$(\pi_1\mu)(a) \leq \alpha(\pi_1\mu)(a) = \alpha \sum_{b \in R(a)} \mu(a, b) = \alpha \sum_{b \in R(a)} f(a, b) = f(\perp, a) \leq c(\perp, a) = \mu_1(a)$$

The same reasoning applies for the inequality $\pi_2\mu \leq \mu_2$. We are left to prove that $\Delta_\alpha(\pi_1\ \mu, \mu_1) \leq \delta$ and $\Delta_\alpha(\pi_2\ \mu, \mu_2) \leq \delta$. We focus on the former inequality, the latter can be shown with a similar argument. Lemma A.3 together with condition $\pi_1\mu \leq \mu_1$ implies $\Delta_\alpha(\pi_1\mu, \mu_1) = \mu_1(A_0) - \alpha(\pi_1\mu)(A_0)$ where $A_0 \stackrel{\text{def}}{=} \{a \in A \mid \mu_1(a) \geq \alpha(\pi_1\mu)(a)\}$. Thus,

$$\begin{aligned}
\mu_1(A) - \delta \leq f_\perp = \sum_{a \in A} f(\perp, a) &= \sum_{a \notin A_0} f(\perp, a) + \sum_{a \in A_0} f(\perp, a) \\
&\stackrel{(1)}{\leq} \sum_{a \notin A_0} \mu_1(a) + \sum_{a \in A_0} f(\perp, a) \stackrel{(2)}{\leq} \sum_{a \notin A_0} \mu_1(a) + \alpha \sum_{a \in A_0} \sum_{b \in R(a)} f(a, b) \\
&= \sum_{a \notin A_0} \mu_1(a) + \alpha \sum_{a \in A_0} \sum_{b \in R(a)} d(a, b) = \mu_1(A) - \mu_1(A_0) + \alpha(\pi_1\mu)(A_0)
\end{aligned}$$

Inequality (1) holds since $f(\perp, a) \leq c(\perp, a) = \mu_1(a)$, whereas (2) is immediate from (8). Hence we have $\mu_1(A_0) - \alpha(\pi_1\mu)(A_0) \leq \delta$, which concludes the proof. □

REFERENCES

ALMEIDA, J. B., BARBOSA, M., BANGERTER, E., BARTHE, G., KRENN, S., AND ZANELLA-BÉGUELIN, S. 2012. Full proof cryptography: verifiable compilation of efficient zero-knowledge protocols. In *ACM Conference on Computer and Communications Security*, T. Yu, G. Danezis, and V. D. Gligor, Eds. ACM, 488–500.

AMTOFT, T., BANDHAKAVI, S., AND BANERJEE, A. 2006. A logic for information flow in object-oriented programs. In *33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006*. ACM, New York, 91–102.

AMTOFT, T. AND BANERJEE, A. 2004. Information flow analysis in logical form. In *SAS*, R. Giacobazzi, Ed. Lecture Notes in Computer Science Series, vol. 3148. Springer, 100–115.

AUDEBAUD, P. AND PAULIN-MOHRING, C. 2009. Proofs of randomized algorithms in Coq. *Sci. Comput. Program. 74,* 8, 568–589.

BACKES, M., KÖPF, B., AND RYBALCHENKO, A. 2009. Automatic discovery and quantification of information leaks. In *30th IEEE Symposium on Security and Privacy, S&P 2009*. IEEE Computer Society, 141–153.

BAELDE, D., COURTIEU, P., GROSS-AMBLARD, D., AND PAULIN-MOHRING, C. 2012. Towards provably robust watermarking. In *ITP*, L. Beringer and A. P. Felty, Eds. Lecture Notes in Computer Science Series, vol. 7406. Springer, 201–216.

BARTHE, G., D'ARGENIO, P., AND REZK, T. 2004. Secure information flow by self-composition. In *17th IEEE Workshop on Computer Security Foundations, CSFW 2004*. IEEE Computer Society, Washington, 100–114.

BARTHE, G., GRÉGOIRE, B., HERAUD, S., AND ZANELLA-BÉGUELIN, S. 2011a. Computer-aided security proofs for the working cryptographer. In *Advances in Cryptology – CRYPTO 2011*. Lecture Notes in Computer Science Series, vol. 6841. Springer, Heidelberg, 71–90.

BARTHE, G., GRÉGOIRE, B., AND ZANELLA-BÉGUELIN, S. 2009. Formal certification of code-based cryptographic proofs. In *36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009*. ACM, New York, 90–101.

BARTHE, G. AND KÖPF, B. 2011. Information-theoretic bounds for differentially private mechanisms. In *24rd IEEE Computer Security Foundations Symposium, CSF 2011*. IEEE Computer Society, Los Alamitos, 191–204.

BARTHE, G., KÖPF, B., OLMEDO, F., AND ZANELLA-BÉGUELIN, S. 2012. Probabilistic relational reasoning for differential privacy. In *39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012*. ACM, New York, 97–110.

BARTHE, G., OLMEDO, F., AND ZANELLA-BÉGUELIN, S. 2011b. Verifiable security of Boneh-Franklin identity-based encryption. In *5th International Conference on Provable Security, ProvSec 2011*. Lecture Notes in Computer Science Series, vol. 6980. Springer, Heidelberg, 68–83.

BEIMEL, A., NISSIM, K., AND OMRI, E. 2008. Distributed private data analysis: Simultaneously solving how and what. In *Advances in Cryptology – CRYPTO 2008*. Lecture Notes in Computer Science Series, vol. 5157. Springer, Heidelberg, 451–468.

BENTON, N. 2004. Simple relational correctness proofs for static analyses and program transformations. In *31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004*. ACM, New York, 14–25.

BURNIM, J. AND SEN, K. 2009. Asserting and checking determinism for multithreaded programs. In *ESEC/SIGSOFT FSE*, H. van Vliet and V. Issarny, Eds. ACM, 3–12.

CHADHA, R., CRUZ-FILIPE, L., MATEUS, P., AND SERNADAS, A. 2007. Reasoning about probabilistic sequential programs. *Theoretical Computer Science 379,* 1-2, 142–165.

CHAN, T.-H. H., SHI, E., AND SONG, D. 2010. Private and continual release of statistics. In *37th International colloquium on Automata, Languages and Programming, ICALP 2010*. Lecture Notes in Computer Science Series, vol. 6199. Springer, Heidelberg, 405–417.

CHAUDHURI, S., GULWANI, S., LUBLINERMAN, R., AND NAVIDPOUR, S. 2011. Proving programs robust. In *8th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, ESEC/FSE '11*. ACM.

CLARK, D., HUNT, S., AND MALACARIA, P. 2007. A static analysis for quantifying information flow in a simple imperative language. *Journal of Computer Security 15,* 3, 321–371.

CLARKSON, M. R. AND SCHNEIDER, F. B. 2010. Hyperproperties. *Journal of Computer Security 18,* 6, 1157–1210.

COBLE, A. R. 2008. Formalized information-theoretic proofs of privacy using the hol4 theorem-prover. In *Privacy Enhancing Technologies*, N. Borisov and I. Goldberg, Eds. Lecture Notes in Computer Science Series, vol. 5134. Springer, 77–98.

COBLE, A. R. 2010. Anonymity, information, and machine-assisted proof. Tech. Rep. UCAM-CL-TR-785, University of Cambridge, Computer Laboratory. July.

DEN HARTOG, J. 1999. Verifying probabilistic programs using a hoare like logic. In *ASIAN*, P. S. Thiagarajan and R. H. C. Yap, Eds. Lecture Notes in Computer Science Series, vol. 1742. Springer, 113–125.

DESHARNAIS, J., LAVIOLETTE, F., AND TRACOL, M. 2008. Approximate analysis of probabilistic processes: Logic, simulation and games. In *5th International Conference on Quantitative Evaluation of Systems, QEST 2008*. IEEE Computer Society, 264–273.

DWORK, C. 2008. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*. Lecture Notes in Computer Science Series, vol. 4978. Springer, Heidelberg, 1–19.

DWORK, C. 2011. A firm foundation for private data analysis. *Commun. ACM 54,* 1, 86–95.

DWORK, C., KENTHAPADI, K., MCSHERRY, F., MIRONOV, I., AND NAOR, M. 2006a. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology – EUROCRYPT 2006*. Lecture Notes in Computer Science Series, vol. 4004. Springer, Heidelberg, 486–503.

DWORK, C., MCSHERRY, F., NISSIM, K., AND SMITH, A. 2006b. Calibrating noise to sensitivity in private data analysis. In *3rd Theory of Cryptography Conference, TCC 2006*. Lecture Notes in Computer Science Series, vol. 3876. Springer, Heidelberg, 265–284.

DWORK, C., ROTHBLUM, G. N., AND VADHAN, S. P. 2010. Boosting and differential privacy. In *Symposium on Foundations of Computer Science – FOCS 2010*. IEEE, 51–60.

FELDMAN, Y. A. AND HAREL, D. 1984. A probabilistic dynamic logic. *J. Comput. Syst. Sci. 28,* 2, 193–215.

GOLDFARB, D., JIN, Z., AND ORLIN, J. 1997. Polynomial-time highest-gain augmenting path algorithms for the generalized circulation problem. *Math. Oper. Res. 22,* 4, 793–802.

GUPTA, A., LIGETT, K., MCSHERRY, F., ROTH, A., AND TALWAR, K. 2010. Differentially private combinatorial optimization. In *21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*. SIAM, 1106–1125.

HAEBERLEN, A., PIERCE, B. C., AND NARAYAN, A. 2011. Differential privacy under fire. In *20th USENIX Security Symposium*. USENIX Association.

HURD, J. 2003. Formal verification of probabilistic algorithms. Tech. Rep. UCAM-CL-TR-566, University of Cambridge, Computer Laboratory. May.

HURD, J., MCIVER, A., AND MORGAN, C. 2005. Probabilistic guarded commands mechanized in HOL. *Theor. Comput. Sci. 346,* 1, 96–112.

JONES, C. 1993. Probabilistic non-determinism. Ph.D. thesis, University of Edinburgh.

JONSSON, B., YI, W., AND LARSEN, K. G. 2001. Probabilistic extensions of process algebras. In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. Elsevier, Amsterdam, 685–710.

KASIVISWANATHAN, S. P. AND SMITH, A. 2008. A note on differential privacy: Defining resistance to arbitrary side information. Cryptology ePrint Archive, Report 2008/144.

KIFER, D. AND MACHANAVAJJHALA, A. 2011. No free lunch in data privacy. In *2011 International Conference on Management of Data, SIGMOD '11*. ACM, 193–204.

KOZEN, D. 1985. A probabilistic pdl. *J. Comput. Syst. Sci. 30,* 2, 162–178.

LAWLER, E. 1976. *Combinatorial optimization: networks and matroids*. Holt, Rinehart and Winston.

MCSHERRY, F. AND TALWAR, K. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007*. IEEE Computer Society, Washington, 94–103.

MCSHERRY, F. D. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *35th SIGMOD International Conference on Management of Data, SIGMOD 2009*. ACM, New York, 19–30.

MHAMDI, T., HASAN, O., AND TAHAR, S. 2010. On the formalization of the lebesgue integration theory in hol. In *ITP*, M. Kaufmann and L. C. Paulson, Eds. Lecture Notes in Computer Science Series, vol. 6172. Springer, 387–402.

MHAMDI, T., HASAN, O., AND TAHAR, S. 2011. Formalization of entropy measures in hol. In *ITP*, M. C. J. D. van Eekelen, H. Geuvers, J. Schmaltz, and F. Wiedijk, Eds. Lecture Notes in Computer Science Series, vol. 6898. Springer, 233–248.

MIRONOV, I., PANDEY, O., REINGOLD, O., AND VADHAN, S. 2009. Computational differential privacy. In *Advances in Cryptology – CRYPTO 2009*. Lecture Notes in Computer Science Series, vol. 5677. Springer, Heidelberg, 126–142.

MORGAN, C., MCIVER, A., AND SEIDEL, K. 1996. Probabilistic predicate transformers. *ACM Trans. Program. Lang. Syst. 18,* 3, 325–353.

MURTY, K. 1992. *Network programming*. Prentice Hall.

NIKOLOV, A., TALWAR, K., AND ZHANG, L. 2012. The geometry of differential privacy: the sparse and approximate cases. *CoRR abs/1212.0297*.

PIERRO, A. D., HANKIN, C., AND WIKLICKY, H. 2004. Approximate non-interference. *Journal of Computer Security 12,* 1, 37–82.

PITT, L. 1985. A simple probabilistic approximation algorithm for vertex cover. Tech. Rep. TR-404, Yale University.

RAMSEY, N. AND PFEFFER, A. 2002. Stochastic lambda calculus and monads of probability distributions. In *29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2002*. ACM, New York, 154–165.

REED, J. AND PIERCE, B. C. 2010. Distance makes the types grow stronger: a calculus for differential privacy. In *15th ACM SIGPLAN International Conference on Functional programming, ICFP 2010*. ACM, New York, 157–168.

REIF, J. H. 1980. Logics for probabilistic programming (extended abstract). In *STOC*, R. E. Miller, S. Ginsburg, W. A. Burkhard, and R. J. Lipton, Eds. ACM, 8–13.

ROY, I., SETTY, S. T. V., KILZER, A., SHMATIKOV, V., AND WITCHEL, E. 2010. Airavat: security and privacy for MapReduce. In *7th USENIX Conference on Networked Systems Design and Implementation, NSDI 2010*. USENIX Association, Berkeley, 297–312.

SABELFELD, A. AND SANDS, D. 2000. Probabilistic noninterference for multi-threaded programs. In *13th IEEE Workshop on Computer Security Foundations, CSFW 2000*. IEEE Computer Society, Los Alamitos, 200–215.

SEGALA, R. AND TURRINI, A. 2007. Approximated computationally bounded simulation relations for probabilistic automata. In *20th IEEE Computer Security Foundations Symposium, CSF 2007*. IEEE Computer Society, 140–156.

TARDOS, E. AND WAYNE, K. 1998. Simple generalized maximum flow algorithms. In *Integer Programming and Combinatorial Optimization*. Lecture Notes in Computer Science Series, vol. 1412. Springer Berlin / Heidelberg, 310–324.

TERAUCHI, T. AND AIKEN, A. 2005. Secure information flow as a safety problem. In *12th International Symposium on Static Analysis, SAS 2005*. Lecture Notes in Computer Science Series, vol. 3672. Springer, Heidelberg, 352–367.

THE COQ DEVELOPMENT TEAM. 2010. The Coq Proof Assistant Reference Manual Version 8.3. Online – `http://coq.inria.fr`.

TSCHANTZ, M. C., KAYNAR, D., AND DATTA, A. 2011. Formal verification of differential privacy for interactive systems. *Electronic Notes in Theoretical Computer Science 276*, 61–79.

ZAKS, A. AND PNUELI, A. 2008. Covac: Compiler validation by program analysis of the cross-product. In *FM*, J. Cuéllar, T. S. E. Maibaum, and K. Sere, Eds. Lecture Notes in Computer Science Series, vol. 5014. Springer, 35–51.

Received September 2012; revised March 2013; accepted