UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA

POLITÉCNICA

Approximate Relational Reasoning for Probabilistic Programs

PhD Thesis

Federico Olmedo

Departamento de Lenguajes,
Sistemas Informáticos e Ingeniería del Software
FACULTAD DE INFORMÁTICA

Approximate Relational Reasoning for Probabilistic Programs

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Software and Systems

Author: **Federico Olmedo**
Advisor: **Gilles Barthe**

Abstract

The *verified security* methodology is an emerging approach to build high assurance proofs about security properties of computer systems. Computer systems are modeled as probabilistic programs and one relies on rigorous program semantics techniques to prove that they comply with a given security goal. In particular, it advocates the use of interactive theorem provers or automated provers to build fully formal machine-checked versions of these security proofs.

The verified security methodology has proved successful in modeling and reasoning about several standard security notions in the area of cryptography. However, it has fallen short of covering an important class of approximate, quantitative security notions. The distinguishing characteristic of this class of security notions is that they are stated as a "similarity" condition between the output distributions of two probabilistic programs, and this similarity is quantified using some notion of distance between probability distributions.

This class comprises prominent security notions from multiple areas such as private data analysis, information flow analysis and cryptography. These include, for instance, indifferentiability, which enables securely replacing an idealized component of system with a concrete implementation, and differential privacy, a notion of privacy-preserving data mining that has received a great deal of attention in the last few years. The lack of rigorous techniques for verifying these properties is thus an important problem that needs to be addressed.

In this dissertation we introduce several quantitative program logics to reason about this class of security notions. Our main theoretical contribution is, in particular, a quantitative variant of a full-fledged relational Hoare logic for probabilistic programs. The soundness of these logics is fully formalized in the Coq proof-assistant and tool support is also available through an extension of CertiCrypt, a framework to verify cryptographic proofs in Coq.

We validate the applicability of our approach by building fully machine-checked proofs for several systems that were out of the reach of the verified security methodology. These comprise, among others, a construction to build "safe" hash functions into elliptic curves and differentially private algorithms for several combinatorial optimization problems from the recent literature.

Resumen

La *seguridad verificada* es una metodología para demostrar propiedades de seguridad de los sistemas informáticos que se destaca por las altas garantías de corrección que provee. Los sistemas informáticos se modelan como programas probabilísticos y para probar que verifican una determinada propiedad de seguridad se utilizan técnicas rigurosas basadas en modelos matemáticos de los programas. En particular, la seguridad verificada promueve el uso de demostradores de teoremas interactivos o automáticos para construir demostraciones completamente formales cuya corrección es certificada mecánicamente (por ordenador).

La seguridad verificada demostró ser una técnica muy efectiva para razonar sobre diversas nociones de seguridad en el área de criptografía. Sin embargo, no ha podido cubrir un importante conjunto de nociones de seguridad "aproximada". La característica distintiva de estas nociones de seguridad es que se expresan como una condición de "similitud" entre las distribuciones de salida de dos programas probabilísticos y esta similitud se cuantifica usando alguna noción de distancia entre distribuciones de probabilidad.

Este conjunto incluye destacadas nociones de seguridad de diversas áreas como la minería de datos privados, el análisis de flujo de información y la criptografía. Ejemplos representativos de estas nociones de seguridad son la indiferenciabilidad, que permite reemplazar un componente idealizado de un sistema por una implementación concreta (sin alterar significativamente sus propiedades de seguridad), o la privacidad diferencial, una noción de privacidad que ha recibido mucha atención en los últimos años y tiene como objetivo evitar la publicación datos confidenciales en la minería de datos. La falta de técnicas rigurosas que permitan verificar formalmente este tipo de propiedades constituye un notable problema abierto que tiene que ser abordado.

En esta tesis introducimos varias lógicas de programa quantitativas para razonar sobre esta clase de propiedades de seguridad. Nuestra principal contribución teórica es una versión quantitativa de una lógica de Hoare relacional para programas probabilísticos. Las pruebas de correción de estas lógicas son completamente formalizadas en el asistente de pruebas Coq. Desarrollamos, además, una herramienta para razonar sobre propiedades de programas a través de estas lógicas extendiendo CertiCrypt, un framework para verificar pruebas de criptografía en Coq.

Confirmamos la efectividad y aplicabilidad de nuestra metodología construyendo pruebas certificadas por ordendor de varios sistemas cuyo análisis estaba fuera del alcance de la seguridad verificada. Esto incluye, entre otros, una meta-construcción para diseñar funciones de hash "seguras" sobre curvas elípticas y algoritmos diferencialmente privados para varios problemas de optimización combinatoria de la literatura reciente.

Disclaimer

This dissertation builds on several published works that I have co-authored.

Conference articles:

- *Verified Indifferentiable Hashing into Elliptic Curves.*
  With Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin.
  In *1st Conference on Principles of Security and Trust - POST 2012*.

- *Probabilistic Relational Reasoning for Differential Privacy.*
  With Gilles Barthe, Boris Köpf, and Santiago Zanella Béguelin.
  In *39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL 2012*.

- *Beyond Differential Privacy: Composition Theorems and Relational Logic for f-divergences between Probabilistic Programs.*
  With Gilles Barthe.
  In *40th International Colloquium on Automata, Languages and Programming - ICALP 2013*.

Journal articles:

- *Probabilistic Relational Reasoning for Differential Privacy.*
  With Gilles Barthe, Boris Köpf, and Santiago Zanella Béguelin.
  In *ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 35, Issue 3*, ACM, 2013.

- *Verified Indifferentiable Hashing into Elliptic Curves.*
  With Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin.
  To appear in *Journal of Computer Security (JCS)*, IOS Press, 2013.

I contributed in the elaboration of all of them, as well as in the development of the supporting machine-checked proofs.

A Coq development accompanying this dissertation is available upon request.

Contents

List of Figures

1 Introduction

Over the past few decades the society dependence on computer systems has grown to an unexpected extent. Nowadays every facet of life involves a computer system on some level. It is thus of paramount importance to recognize the risks that one incurs by employing such systems and to provide means to counter them.

Roughly speaking we can say that computer systems admit two kind of users. On the one hand, a set of intended users for which systems are originally designed. On the other hand, a set of unintended users, usually refereed to as *adversaries*, which perform some kind of undesirable or malicious actions to systems. The aim of computer security is to design defense mechanisms that prevent these malicious actions or mitigate their effects.

The process of developing such defense mechanisms involves two stages: one definitional and one constructive. In the definitional stage one sets a precise *system*[1] *model*, one specifies the *threat model* or adversary's abilities, e.g. the set of actions he can perform or the kind of access to the system he has, and finally one defines a *security property*. The goal of the constructive stage is then to design systems that are secure according to a given system model, threat model and security property.

1.1 Problem Overview

Once a system design is proposed, proving that it actually meets the expected security property is, certainly, a complex and error-prone task. To justify this claim, let us consider the case of cryptography, one of the most mature fields of computer security. We can find a large body of examples [Galindo, 2005; Shoup, 2001], where cryptographic systems were proposed together with a security-compliance proof and shortly afterwards substantial flaws were discovered in the security arguments. This suggests that the development of secure systems calls for a more rigorous methodology that delivers high-assurance security proofs.

[1] From now on we will use the terms "computer system" and "system" interchangeably.

A promising approach to circumvent this problem in the design of computer systems is to borrow techniques from the formal methods community such as the use of interactive theorem provers or automated provers. This approach, which we call *verified security* [Barthe *et al.*, 2011b, 2009b], has been successfully applied to prominent cryptographic constructions such as ElGamal and OAEP encryption schemes, Boneh-Franklin identity-based encryption scheme and the Full Domain Hash (FDH) signature scheme [Barthe *et al.*, 2009a, 2011c,d; Zanella Béguelin *et al.*, 2009].

We argue, however, that it falls short of providing the necessary tools to reason about an important class of approximate, quantitative security notions (we elaborate on these security notions shortly afterwards). To see why, let us review the general structure and standard reasoning patterns behind cryptographic proofs within this approach.

Since Goldwasser & Micali [1984]'s seminal work about provable security, cryptographic systems are designed on the basis of some computational hard problem. Security arguments then proceed as a reduction (in the complexity theory sense) from the security of the system to the hardness assumption of the underlying problem. (Said otherwise, one shows that if the underlying problem is hard, then so is breaking the system). Following the code-based game-playing technique [Bellare & Rogaway, 2006; Shoup, 2004], both hardness assumptions and security goals are represented as experiments where an adversary interacts with a challenger; these experiments are called *games* and are modeled as probabilistic programs. Proofs are structured as a sequence (in general, a tree) of transitions $\mathsf{G}, A \to \mathsf{G}', A'$ between pairs of games and events, such that the probability of event A in game G is bounded by a function of the probability of event A' in game G'. A central property of the code-centric view of games is that it enables justifying game transitions by means of semantic arguments. In particular, the verified security approach relies on a probabilistic relational Hoare logic (pRHL) that manipulates judgments of the form

$$\models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi,$$

where G_1 and G_2 are games and Ψ and Φ are binary relations over program memories. Barthe *et al.* [2009b] present a machine-checked proof system to reason about pRHL judgments and provide the following rules to derive assertions about the probability of events in games from pRHL judgments:

$$\frac{m_1 \,\Psi\, m_2 \qquad \models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi \qquad \Phi \Longrightarrow (A\langle 1\rangle \iff B\langle 2\rangle)}{\Pr\left[\mathsf{G}_1(m_1) : A\right] = \Pr\left[\mathsf{G}_2(m_2) : B\right]}$$

$$\frac{m_1 \,\Psi\, m_2 \qquad \models \mathsf{G}_1 \sim \mathsf{G}_2 : \Psi \Rightarrow \Phi \qquad \Phi \Longrightarrow (A\langle 1\rangle \Longrightarrow B\langle 2\rangle)}{\Pr\left[\mathsf{G}_1(m_1) : A\right] \leq \Pr\left[\mathsf{G}_2(m_2) : B\right]}$$

Here, $\Pr\left[\mathsf{G}(m) : E\right]$ represents the probability of event E in the distribution obtained from running G on the initial memory m; $E\langle 1\rangle$ (resp. $E\langle 2\rangle$) is the lifting of E to a binary memory relation; the pair (m_1, m_2) belongs to $E\langle 1\rangle$ (resp. $E\langle 2\rangle$) if m_1 (resp. m_2) belongs to E.

The above rules have proved useful to reason about a wide range of security notions, but fall short of capturing another important class. Intuitively, the security properties out of scope are modeled using a pair of probabilistic programs (or a single program run on two different initial memories) and are specified as a "similarity" condition between their output distributions (or some related distributions); technically, one considers the distance between their output distributions according to some metric and establishes an upper bound for its value; this upper bound is provided as a parameter of the security notion. Notice that the above pair of rules allows reasoning about the output of probabilistic programs at the level of individual events and compare their probabilities w.r.t. the (in)equality relation, while the foregoing class of security notions requires reasoning at a distribution-wise level, establishing similarity conditions.

This class of security notions comprises prominent concepts from several disciplines such as private data analysis, information flow analysis and cryptography. Let us review some examples.

Differential Privacy. Differential privacy [Dwork, 2006] is a confidentiality policy that provides strong privacy guarantees in the analysis of sensitive data. Assume that D is a database whose rows contain sensitive data about a set of individuals. Informally, a randomized computation c over D is differentially private if the sensitive data of each individual contributing to D is protected against arbitrary adversaries with query access to $c(D)$. Formally, given $\delta \in [0,1]$ and $\epsilon \in \mathbb{R}^{\geq 0}$ we say that a randomized computation c is (ϵ,δ)-*differentially private* if the distributions it outputs on any two inputs D_1 and D_2 differing at most in one row are (ϵ,δ)-close, i.e. if for every event P on the output domain of c we have

$$\Pr[c(D_1) : P] \leq e^{\epsilon} \Pr[c(D_2) : P] + \delta.$$

Intuitively, a differentially private computation c is insensitive to changes in the contribution of any particular individual. This prevents data leaks through the output of c.

Observe that the we consider a rather unrestrictive thread model: it allows for adversaries of unbound computational power and arbitrary auxiliary information. This turns differential privacy into a very strong privacy guarantee.

Approximate Probabilistic Noninterference. In a nutshell, noninterference [Goguen & Meseguer, 1982] can be viewed as a confidentiality policy that prevents the flow of information from a secret part of a system to a public part of the system. For the sake of simplicity we now discuss the termination-insensitive version of noninterference. Let c be a probabilistic program endowed with a denotational semantics $[\![c]\!] : \mathcal{M} \to \mathcal{D}(\mathcal{M})$ that maps an initial state to a distribution of final states; states (or memories) are mappings from variables to values and each variable is either public or private (public variables contain low sensitive data while private variables contain high sensitive data). We say that c is *noninterferent* if for every pair of initial states m_1 and m_2 that coincide

in their public variables we have

$$\pi_L([\![c]\!]\, m_1) = \pi_L([\![c]\!]\, m_2),$$

where π_L projects a distribution over states to the associated distribution over the set of public variables. As so defined, noninterference is a binary notion: a program is either interferent or noninterferent. Unfortunately, absolute noninterference is too strong and can hardly be achieved in practice. We can obtain a more flexible notion of confidentiality relaxing the above equation as follows:

$$d(\pi_L([\![c]\!]\, m_1),\, \pi_L([\![c]\!]\, m_2)) \leq \epsilon.$$

Here d denotes some notion of distance between probability distributions, e.g. statistical distance or relative entropy, and ϵ quantifies the degree of noninterference of c; absolute noninterference corresponds to $\epsilon = 0$.

Computational Indistinguishability. Computational indistinguishability is not a security notion itself but a concept drawn from the probability and complexity theories that underlies several foundational concepts of the modern cryptography. Loosely speaking, it captures the fact that two probability distributions cannot be told apart efficiently and can thus be considered equivalent for practical purposes. To define it formally, we rely on the concept of statistical distance, which constitutes a metric over the space of probability distributions. Assume that μ_1 and μ_2 are a pair of distributions over some set A. The *statistical distance* between μ_1 and μ_2 is defined as

$$\Delta\left(\mu_1, \mu_2\right) \stackrel{\text{def}}{=} \sup_{A_0 \subseteq A} \left|\Pr\left[\mu_1 : A_0\right] - \Pr\left[\mu_2 : A_0\right]\right|.$$

Then we say that two distributions μ and μ' are ϵ-*computationally indistinguishable* if

$$\Delta\left(D(\mu), D(\mu')\right) \leq \epsilon$$

for any probabilistic polynomial time distinguisher D.[2]

Computational indistinguishability underlies a large body of security properties. Among these are pseudorandomness, secure symmetric-key encryption, computational zero-knowledge and secure multi-party computation, to name but a few. In this dissertation we will consider more closely the notion of indifferentiability.[3]

The indifferentiability framework [Maurer *et al.*, 2004] is used to justify rigorously the substitution of an idealized component in a system by a concrete implementation. Using this framework, we prove that a system satisfy a security property in two steps. We first prove that the system complies its goal assuming that some of its components is ideal. Then we show that the ideal component and its implementation are indifferentiable. It then follows that the resulting system remains secure.

[2] The statistical distance between two distributions over a binary set B can be computed as the (absolute value of the) difference between the probability that they assign to either element of B. Therefore the distance $\Delta\left(D(\mu), D(\mu')\right)$ is usually presented as $\left|\Pr\left[D(\mu) = 1\right] - \Pr\left[D(\mu') = 1\right]\right|$ in the literature.

[3] Do not confuse the foregoing notion of computational indistinguishability with that of system component indistinguishability. The latter is a particular case of indifferentiability.

1.2 Dissertation Contributions

In the previous section we have argued that verified security is an emerging methodology to reason about the security of computer systems but it has fallen short of covering a large class of security notions. The aim of this dissertation is to provide verification techniques that expand the frontiers of the verified security methodology to allow reasoning about this class of security notions.

To this end let us examine more closely the examples discussed in the previous section. Note that both approximate noninterference and differential privacy can be construed as a quantitative 2-safety property [Clarkson & Schneider, 2010; Terauchi & Aiken, 2005]; informally this means that proving that a program satisfies either property requires reasoning simultaneously about two executions of the program. Computational indistinguishability is strongly related to 2-safety properties as well. Indeed, for each distinguisher D, the bound $\Delta(D(\mu), D(\mu')) \leq \epsilon$ can be viewed as a 2-safety property of D. Building on this observation and following Benton [2004], we advocate the use of approximate relational logics to reason about the foregoing class of security notions.

In its most general setting, approximate relational logic judgments have the form

$$c_1 \sim_{d,\delta} c_2 : \Psi \Rightarrow \Phi, \tag{1.1}$$

where c_1 and c_2 are probabilistic programs, Ψ and Φ are binary relations over program states, d is a metric (or weaker notion of distance) between probability distributions and δ is either a value in $\mathbb{R}^{\geq 0}$ or a mapping from program states to values in $\mathbb{R}^{\geq 0}$. The validity of such a judgment implies that δ is an upper bound for the d-distance between the probability distributions generated by programs c_1 and c_2, modulo relational pre- and post-conditions Ψ and Φ on program states.

In this dissertation we formalize several variants of an approximate relational logic in the Coq proof-assistant [The Coq development team, 2010]. The basis of our formalization is CertiCrypt [Barthe *et al.*, 2009b; Zanella Béguelin, 2010], a machine-checked framework for verifying cryptographic proofs in Coq. The resulting framework significantly broadens the scope of the verified security methodology and goes beyond the state-of-the-art in the following two aspects:

Expressivity. The framework enables modelling and *formally* reasoning about a large and important class of quantitative security notions that encompasses e.g. (approximate) differential privacy, approximate probabilistic noninterference, indifferentiability and computational zero-knowledge.

Homogeneity. Even though there exist techniques—outside the verified security methodology—that support reasoning independently about some of the above notions (see Section 6.1 for a discussion of the related work), the use of approximate relational logics allows reasoning about all such notions in a homogeneous and uniform way.

Moreover, the framework inherits the following features from its predecessor CertiCrypt:

Generality. One is not confined to reasoning about a restricted set of domains; the framework inherits the generality of the Coq proof assistant and allows modelling arbitrary objects and incorporating reasoning principles from diverse fields such as algebra, number theory or discrete mathematics. A fundamental consequence of this breadth is that one can justify all kind of intermediate steps and build full security proofs within a single framework.

Extensibility. It can be integrated with complex libraries that formalize involved mathematical results; this ability of the framework will be fully exercised in the case study of Section 3.3.

High-assurance proofs. It delivers fully formalized and independently verifiable proofs. Concretely, every proof yields a proof object that can be checked automatically by a (small and trustworthy) proof checking engine.

In order to illustrate the applicability of our approach we choose two representative security properties and present full security proofs for systems from the recent literature. The first property is indifferentiability from a random oracle, which formally captures the notion of "secure" hash functions; we verify Brier *et al.* [2010]'s construction used to build hash functions into elliptic curves. The second property is differential privacy and we verify several systems: On the one hand, some standard mechanisms that deliver privacy-preserving numerical computations; on the other hand, some approximation algorithms for classical combinatorial optimization problems [Gupta *et al.*, 2010].

Summarizing, we can say that 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, which includes differential privacy, approximate noninterference and indifferentiability. On the practical side, we demonstrate the applicability of our approach by providing the first machine-checked security proof of several constructions from the recent literature.

1.3 Dissertation Outline

In order to make the presentation more accessible to the reader, we first study (Chapters 3 and 4) different variants of an approximate relational logic, each of them being driven by a concrete security notion we aim to model and reason about. Then we demonstrate (Chapter 5) that such logics can be developed in a uniform manner introducing the approximate relational Hoare logic of Equation (1.1).

We next survey in more detail how the rest of the dissertations is organized:

- In Chapter 2 we provide a brief introduction to the CertiCrypt framework, the basis for our formalization. In particular we present the language that we use to describe probabilistic programs, our model of adversaries and the "exact" relational Hoare logic pRHL at the heart of the framework.

- In Chapter 3 we focus on security properties that adopt the statistical distance as metric to compare the output distributions of programs. Concretely, we develop two logics. The first enables reasoning about the approximate observational equivalence of probabilistic programs, which subsumes the notion of approximate noninterference. The second is tailored to bound the statistical distance generated by the call to adversaries that interact with two different environments and is thus suitable to capture the notion of computational indistinguishability. Finally we use these tools to verify that Brier *et al.* [2010]'s construction yield hash functions into elliptic curves that are indifferentiability from a random oracle.

- In Chapter 4 we propose a novel notion of distance between probability distributions coined α-distance and show that the property of differential privacy of a program c can be formulated as an upper bound for the α-distance between the output distributions of c when executed on certain pair of initial memories. Building on this observation, we present a full-fledged relational Hoare logic that allows bounding the α-distance between the output distributions of probabilistic programs. We demonstrate the effectiveness of this logic for proving programs differentially private verifying, among others, two classic numerical mechanisms from the literature and several approximation algorithms for combinatorial optimization problems.

- In Chapter 5 we observe that the statistical distance and the α-distance are members of a more general class of distances between probability distributions known as f-divergences and generalize the relational Hoare logic of Chapter 4 to reason about arbitrary f-divergences. As a preliminary step, we show that the sequential composition theorem of differential privacy extends to an important subset of this class of distance measures.

- In Chapter 6 we survey the related work in the area and conclude.

2
CertiCrypt Overview

In the following chapters we present several variants of an approximate relational program logic and show their applicability for verifying some relevant quantitative security notions. In particular, the developments of Chapters 3 and 4 are formalized in the Coq proof assistant [The Coq development team, 2010] and build upon CertiCrypt, a machine-checked framework for verifying cryptographic proofs in Coq.

This chapter summarizes the main components of CertiCrypt, namely the representation of probability distributions, the probabilistic programming language used to describe games, the thread model used to describe legitimate adversaries and the probabilistic relational Hoare logic pRHL for reasoning about games. We refer the reader to [Zanella Béguelin, 2010] for further details.

2.1 Representation of Distributions

CertiCrypt adopts the monadic representation of distributions provided by the ALEA library [Audebaud & Paulin-Mohring, 2009]. The library builds on an axiomatization of the unit interval $[0,1]$ that supports addition, inversion, multiplication and division as primitive operations (underflows and overflows are mapped to 0 and 1, respectively). The unit interval $[0,1]$ is given the structure of an ω-cpo by taking the usual order and defining an operator "lub" that computes the supremum of monotonic $[0,1]$-valued sequences. This ω-cpo structure on the interval $[0,1]$ readily induces an ω-cpo structure on the function space $A \to [0,1]$ by taking the pointwise order between two functions. In the reminder we use "$\leq$" and "lub" to denote the order and supremum of monotonic sequences over either structure.

A distribution over a set A is defined as a function of type

$$\mathcal{D}(A) \stackrel{\text{def}}{=} (A \to [0,1]) \to [0,1]$$

verifying the following set of (universally quantified) axioms:

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;
Homogeneity of degree 1*:*	$\mu(k \cdot f) = k \cdot \mu(f)$ for any $k \in [0, 1]$;
Additivity:	$f \leq \mathbb{1} - g \implies \mu(f + g) = \mu(f) + \mu(g)$;
Continuity:	$\mu(\text{lub } F) \leq \text{lub } (\mu \circ F)$ for any monotonic sequence $F : \mathbb{N} \to (A \to [0, 1])$.

Observe that we do not require the mass $\mu(\mathbb{1})$ of a distribution μ to be 1; therefore our definition corresponds to probability sub-distributions. This provides an elegant means of giving semantics to runtime assertions and programs that do not terminate with probability 1. In particular, we let μ_0 be the null sub-distribution.

Intuitively, one must view a distribution μ over a set A as an operator mapping a unit-valued random variable (i.e. a function in $A \to [0, 1]$) to its expected value. When A is a discrete, this translates into

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

where $\mu(a)$ denotes the probability density function of μ at a. For instance, the Bernoulli distribution over $\mathbb{B}$ with success probability p is represented as $\lambda f.\ p \cdot f(\mathsf{true}) + (1 - p) \cdot f(\mathsf{false})$, while the distribution over $\mathbb{N}$ that assigns probability $(1/2)^i$ to number i is represented as $\lambda f.\ \sum_{i \in \mathbb{N}} (1/2)^i \cdot f(i)$.

The probability that distribution $\mu \in \mathcal{D}(A)$ assigns to an event $P \subseteq A$ can be computed by measuring its characteristic function $\mathbb{1}_P$, i.e.

$$\Pr[\mu : P] = \mu(\mathbb{1}_P). \tag{2.1}$$

For the sake of notation compactness, we will usually use $\mu(P)$ to denote the probability $\mu(\mathbb{1}_P)$. In particular, when $P = \{a\}$ is a singleton, the probability $\mu(\mathbb{1}_{\{a\}})$ will be denoted as $\mu(a)$.

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:

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

For a value $a \in A$, the expression $\mathsf{unit}\, a$ denotes the Dirac measure on a, which assigns probability 1 to a and 0 to all other values in A (in the continuous case, $\mathsf{unit}\, a$ is the degenerate probability distribution that has all its mass concentrated at a). The bind

operator composes a distribution over A with a conditioned distribution over B given $a \in A$ to yield a distribution over B; when A and B are discrete sets, we have

$$(\mathsf{bind}\ \mu\ M)(b) = \textstyle\sum_{a \in A} \mu(a)\, M(a)(b).$$

For our development it will be enough to consider distributions over discrete sets. Therefore some of the forthcoming definitions related to distributions are specializations to the discrete case. In particular our work builds on the following operations and relations:

$$\begin{array}{rcl}
\mathsf{range}\ P\ \mu & \stackrel{\text{def}}{=} & \forall a \bullet\ \mu(a) > 0 \implies P(a); \\
(\mu_1 \times \mu_2)(a,b) & \stackrel{\text{def}}{=} & \mu_1(a) \cdot \mu_2(b); \\
\pi_1(\mu)(a) & \stackrel{\text{def}}{=} & \textstyle\sum_{b \in B} \mu(a,b); \\
\pi_2(\mu)(b) & \stackrel{\text{def}}{=} & \textstyle\sum_{a \in A} \mu(a,b); \\
w(\mu) & \stackrel{\text{def}}{=} & \mu(\mathbb{1}); \\
\mu \leq \mu' & \stackrel{\text{def}}{=} & \forall a \bullet\ \mu(a) \leq \mu'(a).
\end{array}$$

The formula $\mathsf{range}\ P\ \mu$ says that P overapproximates the support of μ, i.e. the set of elements with non-null probability. (We can view range as a lifting operator that maps predicates over some set A into predicates over $\mathcal{D}(A)$). For a distribution μ_1 over A and a distribution μ_2 over B, $\mu_1 \times \mu_2$ denotes the product distribution (over $A \times B$) between μ_1 and μ_2. 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. Expression $w(\mu)$ denotes the mass of μ. (Proper probability distributions correspond to distributions of unitary mass.) Finally, relation "$\leq$" defines a pointwise partial order on $\mathcal{D}(A)$.

Remark. In our Coq development we slightly depart from the above definitions since we use those provided by the ALEA library, which make sense for a wider family of distributions. For example, the range relation and the π_1 operator are defined as $\mathsf{range}\ P\ \mu \stackrel{\text{def}}{=} \forall f \bullet\ (\forall a \bullet\ P(a) \implies f(a) = 0) \implies \mu(f) = 0$ and $\pi_1(\mu) \stackrel{\text{def}}{=} \mathsf{bind}\ \mu\ (\mathsf{unit} \circ \mathsf{fst})$. However, in our presentation we prefer to stick to the specializations to the discrete case as these definitions are more intuitive and natural.

Besides the above operators over distributions, we will often use μ/R to denote the distribution induced by $\mu \in \mathcal{D}(A)$ on the quotient set A/R, for an equivalence relation $R \subseteq A \times A$; formally it is defined by clause $(\mu/R)([a]) \stackrel{\text{def}}{=} \mu([a])$.

Finally, given a set A and a family of (usually two) distributions $\{\mu_i\}_{i \in I}$ over A we use A_P to denote the subset of A that satisfies formula P, where P is some linear constrain between the probabilities assigned by $\{\mu_i\}_{i \in I}$. For instance, $A_{\mu_1 \leq \mu_2}$ denotes the subset $\{a \mid \mu_1(a) \leq \mu_2(a)\}$ of A.

We conclude this section presenting a lifting operation $\mathcal{L}(\cdot)$ from the probabilistic process algebra [Jonsson *et al.*, 2001] that transforms any binary relation $R \subseteq A \times B$ into a binary relation $\mathcal{L}(R) \subseteq \mathcal{D}(A) \times \mathcal{D}(B)$. This operation is used in Section 2.4 for defining the interpretation of pRHL judgments as this requires lifting a relation over program states into a relation over distributions on program states.

Definition 2.1 (Relation Lifting)**.** *The* lifting *of a relation* $R \subseteq A \times B$ *is a relation* $\mathcal{L}(R) \subseteq \mathcal{D}(A) \times \mathcal{D}(B)$ *such that* $\mu_1 \, \mathcal{L}(R) \, \mu_2$ *iff there exists* $\mu \in \mathcal{D}(A \times B)$ *satisfying*

i) $\mathsf{range}\, R\, \mu$;
ii) $\pi_1(\mu) = \mu_1$;
iii) $\pi_2(\mu) = \mu_2$.

We say that such a distribution μ *is a* witness *for the lifting.*

The lifting operation has close connections with the problem of network flows. Concretely, if R is a finite relation then one can decide the membership to $\mathcal{L}(R)$ by solving a maximum network flow problem. Moreover, if R is an equivalence relation over some set A, then the lifting $\mathcal{L}(R)$ to $\mathcal{D}(A)$ admits a simpler characterization: $\mu_1 \, \mathcal{L}(R) \, \mu_2$ if and only if $\mu_1([a]) = \mu_2([a])$ for every equivalence class $[a]$. In the general case, the relation $\mathcal{L}(R)$ can also be built inductively, starting from pairs of Dirac distributions $\mathsf{unit}\, a$ and $\mathsf{unit}\, b$ where $a \, R \, b$ (in this case, distribution $\mathsf{unit}\, (a, b)$ is a witness for the lifting $(\mathsf{unit}\, a) \, \mathcal{L}(R) \, (\mathsf{unit}\, b)$) [Deng & Du, 2011].

2.2 The pWhile Language

We now describe the programming language adopted by CertiCrypt to describe games. Roughly speaking, games are modeled as probabilistic imperative programs with procedure calls. The set of commands $\mathcal{C}$ is defined inductively by the clauses

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

Here, $\mathcal{V}$ is a set of variables tagged with their scope (either local or global), $\mathcal{P}$ is a set of procedure identifiers, $\mathcal{E}$ is a set of deterministic expressions, and $\mathcal{D}\mathcal{E}$ is a set of distribution expressions that denote distributions from which values can be sampled in random assignments. The base language includes expressions over Booleans, integers, lists, option and product types, but can be extended by the user.

A program consists of a command c and an environment E that maps procedure identifiers to their declaration, specifying their formal parameters, their body, and a return expression that is evaluated upon exit.

$$\mathsf{decl} \stackrel{\text{def}}{=} \{\mathsf{args} : \mathsf{list}\ \mathcal{V};\ \mathsf{body} : \mathcal{C};\ \mathsf{re} : \mathcal{E}\}$$

Although procedures are single-exit, we often write programs using explicit return expressions for the sake of readability. Declarations are subject to well-formedness and well-typedness conditions; these conditions are enforced using the underlying dependent type system of Coq.

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 CertiCrypt programs—and we define the set $\mathcal{M}$ of program states or memories as the set of mappings from variables to values. Then, the semantics of an expression e of type T, a distribution expression d of type T, and a command c, respectively, are given by functions of the following types:

$$[\![e]\!]_{\mathcal{E}} : \mathcal{M} \to [\![T]\!] \qquad [\![d]\!]_{\mathcal{DE}} : \mathcal{M} \to \mathcal{D}([\![T]\!]) \qquad [\![c]\!]_E : \mathcal{M} \to \mathcal{D}(\mathcal{M})$$

Informally, the semantics of an expression e maps a memory to a value in $[\![T]\!]$, the semantics of a distribution expression d maps a memory to a distribution over $[\![T]\!]$, and the semantics of a command c in an environment E maps an initial memory to a distribution over final memories. We often omit the environment of a program when it is irrelevant (e.g. when the program contains no procedure call) or can be inferred from the context. The semantics of programs complies with the expected equations; see Figure 2.1.

$$
\begin{aligned}
[\![\mathsf{skip}]\!]\ m &= \mathsf{unit}\ m \\
[\![c;\ c']\!]\ m &= \mathsf{bind}\ ([\![c]\!]\ m)\ [\![c']\!] \\
[\![x \leftarrow e]\!]\ m &= \mathsf{unit}\ (m\,\{[\![e]\!]_{\mathcal{E}}\ m/x\}) \\
[\![x \xleftarrow{\$} d]\!]\ m &= \mathsf{bind}\ ([\![d]\!]_{\mathcal{DE}}\ m)\ (\lambda v.\ \mathsf{unit}\ (m\,\{v/x\})) \\
[\![\mathsf{assert}\ e]\!]\ m &= \mathbf{if}\ ([\![e]\!]_{\mathcal{E}}\ m = \mathsf{true})\ \mathbf{then}\ (\mathsf{unit}\ m)\ \mathbf{else}\ \mu_0 \\
[\![\mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2]\!]\ m &= \mathbf{if}\ ([\![e]\!]_{\mathcal{E}}\ m = \mathsf{true})\ \mathbf{then}\ ([\![c_1]\!]\ m)\ \mathbf{else}\ ([\![c_2]\!]\ m) \\
[\![\mathsf{while}\ e\ \mathsf{do}\ c]\!]\ m &= \lambda f.\ \mathrm{lub}\ (\lambda n.\ ([\![[\mathsf{while}\ e\ \mathsf{do}\ c]_n]\!]\ m)(f)) \\
&\quad \text{where} \quad [\mathsf{while}\ e\ \mathsf{do}\ c]_0 = \mathsf{assert}\ \neg e \\
&\qquad\qquad\ \ [\mathsf{while}\ e\ \mathsf{do}\ c]_{n+1} = \mathsf{if}\ e\ \mathsf{then}\ c;\ [\mathsf{while}\ e\ \mathsf{do}\ c]_n
\end{aligned}
$$

Figure 2.1: Semantics of pWhile programs.

By specializing the definition of probability $\Pr[\mu : P]$ (see Equation (2.1)) to programs, we have that the probability of an event P in a program c run on an initial memory m is given by

$$\Pr[c(m) : P] = ([\![c]\!]\ m)(\mathbb{1}_P).$$

In particular, if c is assertion-free and samples only from proper probability distributions, $\Pr[c(m) : \mathsf{true}]$ denotes the termination probability of c in the initial memory m. Moreover, we say that a program c is *lossless* iff $\Pr[c(m) : \mathsf{true}] = 1$ for any initial memory m.

In order to reason about program complexity and define the class of probabilistic polynomial-time computations, the semantics of programs is indexed by a security parameter (a natural number) and instrumented to compute the time and memory cost of evaluating a command, given the time and memory cost of each construction in the expression language. We choose not to make this parametrization explicit to avoid cluttering the presentation.

In this dissertation we only consider programs that sample values from discrete distributions, and so their output distributions are also discrete.

2.3 Adversarial Model

Adversaries are modeled as procedures whose code and return expression are unknown. However, adversaries are required to respect basic interface conditions; such conditions enforce scoping, and may ensure for example that the adversary cannot read or write values that he has to guess. Formally, an interface is a triple $(\mathcal{O}, \mathcal{RW}, \mathcal{R})$, where $\mathcal{O}$ is a set of oracles, and $\mathcal{RW}$ and $\mathcal{R}$ are sets of variables. An adversary respects an interface $(\mathcal{O}, \mathcal{RW}, \mathcal{R})$ if he only reads variables in $\mathcal{RW} \cup \mathcal{R}$, only writes variables in $\mathcal{RW}$, and only call oracles in $\mathcal{O}$ or procedures that respect the same interface as himself. In this case we say that $\mathcal{A}$ is *well-formed* w.r.t. interface $(\mathcal{O}, \mathcal{RW}, \mathcal{R})$ and note it $\vdash_{\text{wf}} \mathcal{A}$. This condition is defined inductively by the rules of Figure 2.2. We remark that this set of rules only aims at ensuring the correct use of variables and procedure calls by the adversary and are general enough as to capture the behaviour of any legitimate adversary in standard cryptographic security models. Any additional constraints, such as conditions on the number or form of oracle calls may be stated as post-conditions of security experiments.

The system of Figure 2.2 yields an induction principle for well-formed adversaries of key importance in our development, since it allows to extend any proof system for closed programs to programs with calls to well-formed adversaries. Specifically, in Section 3.2.1 we present a logic to bound the statistical distance between the output distributions of two (structurally similar) programs. The soundness of the rule for calling a well-formed adversary is proved by structural induction on the derivation of its well-formedness.

2.4 Reasoning Tools

CertiCrypt provides several tools for reasoning about games. One main tool is a probabilistic relational Hoare logic coined pRHL. Its judgments are of the form

$$c_1 \sim c_2 : \Psi \Rightarrow \Phi,$$

where c_1 and c_2 are programs and Ψ and Φ are relations over states (we assume a pair E_1 and E_2 of fixed environments for c_1 and c_2). CertiCrypt uses shallow embedding for state relations, and thus inherits the expressiveness of Coq when writing (relational) pre- and post-conditions. Throughout the exposition, we usually specify a relation $m_1 \,\Theta\, m_2$ as a

$$I \vdash \mathsf{skip} : I \qquad \frac{I \vdash i : I' \quad I' \vdash c : O}{I \vdash i; c : O}$$

$$\frac{\mathsf{writable}(x) \quad \mathsf{fv}(e) \subseteq I}{I \vdash x \leftarrow e : I \cup \{x\}} \qquad \frac{\mathsf{writable}(x)}{I \vdash x \xleftarrow{\$} T : I \cup \{x\}}$$

$$\frac{\mathsf{fv}(e) \subseteq I \quad I \vdash c_i : O_i,\ i = 1,2}{I \vdash \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 : O_1 \cap O_2} \qquad \frac{\mathsf{fv}(e) \subseteq I \quad I \vdash c : I}{I \vdash \mathsf{while}\ e\ \mathsf{do}\ c : I}$$

$$\frac{\mathsf{fv}(\vec{e}) \subseteq I \quad \mathsf{writable}(x) \quad p \in \mathcal{O}}{I \vdash x \leftarrow p(\vec{e}) : I \cup \{x\}}$$

$$\frac{\mathsf{fv}(\vec{e}) \subseteq I \quad \mathsf{writable}(x) \quad p \notin \mathcal{O} \quad \vdash_{\mathrm{wf}} p}{I \vdash x \leftarrow p(\vec{e}) : I \cup \{x\}}$$

$$\frac{\mathcal{RW} \cup \mathcal{R} \cup \mathcal{A}.\mathsf{args} \vdash \mathcal{A}.\mathsf{body} : O \quad \mathsf{fv}(\mathcal{A}.\mathsf{re}) \subseteq O}{\vdash_{\mathrm{wf}} \mathcal{A}}$$

$$\mathsf{writable}(x) \stackrel{\text{def}}{=} \mathsf{local}(x) \vee x \in \mathcal{RW}$$

Figure 2.2: Rules for well-formedness of an adversary against interface $(\mathcal{O}, \mathcal{RW}, \mathcal{R})$. A judgment of the form $I \vdash c : O$ reads as follows: assuming variables in I may be read, the adversarial code fragment c respects the interface, and after its execution variables in O may be read. Thus, $I \vdash c : O \implies I \subseteq O$.

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 formula $e_1\langle 1 \rangle < e_2\langle 2 \rangle$ denotes the relation $\{(m_1, m_2) \mid [\![e_1]\!]_{\mathcal{E}}\ m_1 < [\![e_2]\!]_{\mathcal{E}}\ m_2\}$.

Formally, a judgment $c_1 \sim c_2 : \Psi \Rightarrow \Phi$ is *valid*, written $\models c_1 \sim c_2 : \Psi \Rightarrow \Phi$, iff for all memories m_1 and m_2 we have

$$m_1\ \Psi\ m_2 \implies ([\![c_1]\!]\ m_1)\ \mathcal{L}\ (\Phi)\ ([\![c_2]\!]\ m_2).$$

Figure 2.3 provides an excerpt of the set of rules of the relational Hoare logic. The logic can be used to prove (in)equalities between probability quantities; to this end we can rely on the following rules:

$$\frac{m_1\ \Psi\ m_2 \qquad \models c_1 \sim c_2 : \Psi \Rightarrow \Phi \qquad \Phi \implies (A\langle 1 \rangle \implies B\langle 2 \rangle)}{\Pr[c_1(m_1) : A] \leq \Pr[c_2(m_2) : B]}\ [\mathsf{PrLe}]$$

$$\frac{m_1\ \Psi\ m_2 \qquad \models c_1 \sim c_2 : \Psi \Rightarrow \Phi \qquad \Phi \implies (A\langle 1 \rangle \iff B\langle 2 \rangle)}{\Pr[c_1(m_1) : A] = \Pr[c_2(m_2) : B]}\ [\mathsf{PrEq}]$$

Observational equivalence is defined by specializing judgments to relations Ψ and Φ corresponding to the equality relation on subsets of program variables. Formally, let X be a set of variables, $m_1, m_2 \in \mathcal{M}$ and $f_1, f_2 : \mathcal{M} \to [0,1]$. We define

$$\begin{aligned} m_1 =_X m_2 &\stackrel{\text{def}}{=} \forall x \in X \bullet m_1(x) = m_2(x); \\ f_1 =_X f_2 &\stackrel{\text{def}}{=} \forall m_1, m_2 \bullet m_1 =_X m_2 \implies f_1(m_1) = f_2(m_2). \end{aligned}$$

$$\frac{\forall m_1, m_2 \bullet\ m_1\, \Psi\, m_2 \implies (m_1\,\{[\![e_1]\!]_{\mathcal{E}}\ m_1/x_1\})\ \Phi\ (m_2\,\{[\![e_2]\!]_{\mathcal{E}}\ m_2/x_2\})}{\models x_1 \leftarrow e_1 \sim x_2 \leftarrow e_2 : \Psi \Rightarrow \Phi}\ [\mathsf{assn}]$$

$$\frac{\begin{array}{c}\forall m_1, m_2 \bullet\ m_1\, \Psi\, m_2 \implies ([\![d_1]\!]_{\mathcal{DE}}\ m_1)\,\mathcal{L}\,(\Theta)\ ([\![d_2]\!]_{\mathcal{DE}}\ m_2) \\ \text{where } v_1\,\Theta\,v_2 \stackrel{\text{def}}{=} (m_1\,\{v_1/x_1\})\,\Phi\,(m_2\,\{v_2/x_2\})\end{array}}{\models x_1 \stackrel{\$}{\leftarrow} d_1 \sim x_2 \stackrel{\$}{\leftarrow} d_2 : \Psi \Rightarrow \Phi}\ [\mathsf{rnd}]$$

$$\frac{\Psi \implies b_1\langle 1\rangle = b_2\langle 2\rangle}{\models \mathsf{assert}\ b_1 \sim \mathsf{assert}\ b_2 : \Psi \Rightarrow \Psi \wedge b_1\langle 1\rangle}\ [\mathsf{assert}]$$

$$\frac{\Psi \implies b_1\langle 1\rangle = b_2\langle 2\rangle \quad \models c_1 \sim c_2 : \Psi \wedge b_1\langle 1\rangle \Rightarrow \Phi \qquad \models c_1' \sim c_2' : \Psi \wedge \neg b_1\langle 1\rangle \Rightarrow \Phi}{\models \mathsf{if}\ b_1\ \mathsf{then}\ c_1\ \mathsf{else}\ c_1' \sim \mathsf{if}\ b_2\ \mathsf{then}\ c_2\ \mathsf{else}\ c_2' : \Psi \Rightarrow \Phi}\ [\mathsf{cond}]$$

$$\frac{\Theta \implies b_1\langle 1\rangle = b_2\langle 2\rangle \quad \models c_1 \sim c_2 : \Theta \wedge b_1\langle 1\rangle \Rightarrow \Theta}{\models \mathsf{while}\ b_1\ \mathsf{do}\ c_1 \sim \mathsf{while}\ b_2\ \mathsf{do}\ c_2 : \Theta \Rightarrow \Theta \wedge \neg b_1\langle 1\rangle}\ [\mathsf{while}]$$

$$\models \mathsf{skip} \sim \mathsf{skip} : \Phi \Rightarrow \Phi\ [\mathsf{skip}] \qquad \frac{\models c_1 \sim c_2 : \Psi \Rightarrow \Theta \qquad \models c_1' \sim c_2' : \Theta \Rightarrow \Phi}{\models c_1; c_1' \sim c_2; c_2' : \Psi \Rightarrow \Phi}\ [\mathsf{seq}]$$

$$\frac{\Psi \implies \Psi' \qquad \Phi' \implies \Phi \quad \models c_1 \sim c_2 : \Psi' \Rightarrow \Phi'}{\models c_1 \sim c_2 : \Psi \Rightarrow \Phi}\ [\mathsf{weak}] \qquad \frac{\models c_1 \sim c_2 : \Psi \wedge \Psi' \Rightarrow \Phi \quad \models c_1 \sim c_2 : \Psi \wedge \neg\Psi' \Rightarrow \Phi}{\models c_1 \sim c_2 : \Psi \Rightarrow \Phi}\ [\mathsf{case}]$$

Figure 2.3: Selected proof rules of pRHL.

Then, two programs c_1 and c_2 are *observationally equivalent* w.r.t. an input set of variables I and an output set of variables O, written $\models c_1 \simeq_O^I c_2$, iff

$$\models c_1 \sim c_2 : =_I \Rightarrow =_O.$$

Equivalently, $\models c_1 \simeq_O^I c_2$ iff for all memories $m_1, m_2 \in \mathcal{M}$ and functions $f_1, f_2 : \mathcal{M} \to [0,1]$,

$$m_1 =_I m_2 \wedge f_1 =_O f_2 \implies ([\![c_1]\!]\, m_1)(f_1) = ([\![c_2]\!]\, m_2)(f_2).$$

Observational equivalence is amenable to automation. CertiCrypt provides mechanized tactics based on dependency analyses to perform common program transformations and to prove that two programs satisfy an observational equivalence specification. Relevant mechanized transformations used in this dissertation include dead code elimination, procedure call inlining, code motion and expression propagation. Since observational equivalence

is only a partial equivalence relation, it might be of interest to prove that a program is observational equivalent to itself; CertiCrypt also provide an automatic tactic to prove a program self-equivalent.

In the exposition we sometimes use a standard (probabilistic) Hoare logic for reasoning about individual programs. Its judgments are of the form

$$\{P\}\ c\ \{Q\},$$

where c is a program and P and Q are predicates on states. Formally, a judgment $\{P\}\ c\ \{Q\}$ is *valid* iff for every memory $m \in \mathcal{M}$,

$$P(m) \implies \mathsf{range}\ (\llbracket c \rrbracket\ m)\ Q.$$

3

Security Analysis based on the Statistical Distance

Following an established trend [Goldwasser & Micali, 1984], the prevailing methodology for building secure cryptosystems is to conduct a rigorous analysis that proves security under standard hypotheses. Sometimes this analysis is performed assuming that some components of the system have an ideal behavior. However, ideal functionalities are difficult or even impossible to realize, leading to situations where provably secure systems have no secure implementation. An alternative methodology is to devise systems based on constructions that do not deviate significantly from ideal ones, and to account for these deviations in the security analysis.

For quantifying the deviation between idealized functionalities and their implementations, the common practice is to rely on the notion of statistical distance. The aim of this chapter is to develop several quantitative notions of program equivalence and related logics for upper-bounding the statistical distance between distributions generated by probabilistic programs. More specifically, we introduce a logic for bounding the statistical distance between the output distributions of two programs in the presence of adversaries (Section 3.2.1) and an approximate variant of observational equivalence that supports reasoning through an equational theory (Section 3.2.2).

Finally we show the applicability of these tools by presenting a machine-checked proof of Brier *et al.* [2010]'s construction for building hash functions into elliptic curves (Section 3.3); loosely speaking, we prove that their construction yields hash functions whose behaviour is similar to the ideal one—formally modeled as a random oracle—and thus can be used safely to implement provable secure cryptosystems. These guarantees are formally established using the indifferentiability framework of Maurer *et al.* [2004].

3.1 Statistical Distance

Statistical distance quantifies the largest difference between the probability that two distributions assign to the same event, and underlies many concepts in cryptography, such as indifferentiability [Maurer *et al.*, 2004] and statistical zero-knowledge proofs [Goldreich, 2002]. We review its definition next, and provide below an alternative characterization that is more appropriate for reasoning about the monadic representation of distributions that we use in our development; we refer to Shoup [2009] or Sahai & Vadhan [1999] for an in-depth presentation of statistical distance and its properties.

Definition 3.1 (Statisitical Distance)**.** *The statistical distance* $\Delta(\mu_1, \mu_2)$ *between two distributions* μ_1 *and* μ_2 *over a set* A *is defined as*

$$\Delta(\mu_1, \mu_2) \stackrel{\text{def}}{=} \sup_{f:A\to\{0,1\}} |\mu_1(f) - \mu_2(f)|. \tag{3.1}$$

Note that for any f, the expression $|\mu_1(f) - \mu_2(f)|$ is upper-bounded by 1; hence the supremum exists and $\Delta(\mu_1, \mu_2)$ is well defined. Statistical distance satisfies the metric axioms, and is non-increasing w.r.t. the bind operator, i.e.

i) $0 \le \Delta(\mu_1, \mu_2) \le 1$, and $\Delta(\mu_1, \mu_2) = 0$ iff $\mu_1 = \mu_2$;
ii) $\Delta(\mu_1, \mu_2) = \Delta(\mu_2, \mu_1)$;
iii) $\Delta(\mu_1, \mu_3) \le \Delta(\mu_1, \mu_2) + \Delta(\mu_2, \mu_3)$;
iv) $\Delta(\mathsf{bind}\ \mu_1\ M, \mathsf{bind}\ \mu_2\ M) \le \Delta(\mu_1, \mu_2)$.

For discrete distributions, an equivalent definition is obtained if one lets f in (3.1) range over the real interval $[0, 1]$ rather than over Boolean values. This characterization of statistical distance is more convenient for reasoning about our monadic formalization of distributions and corresponds to the definition that we adopt in the Coq development.

Lemma 3.1. *For any pair of discrete distributions* μ_1 *and* μ_2 *over a set* A*,*

$$\Delta(\mu_1, \mu_2) = \sup_{f:A\to[0,1]} |\mu_1(f) - \mu_2(f)|.$$

Proof. The proof follows by taking $\alpha = 1$ in Lemma 4.1. ■

3.2 Weak Notions of Program Equivalence

3.2.1 A Logic for Bounding the Statistical Distance

In this section, we consider the problem of bounding the statistical distance between the distributions output by two programs given the same initial memory. Formally, let c_1 and c_2 be two programs, which we assume to be executed in two fixed environments E_1 and E_2 and let $\Delta_m(c_1, c_2)$ denote the distance $\Delta([\![c_1]\!]\ m, [\![c_2]\!]\ m)$. We define a logic that allows

upper-bounding the distance $\Delta_m(c_1, c_2)$ by a function of the memory m. Concretely, we consider judgments of the form

$$\langle\!\langle\, c_1, c_2 \,\rangle\!\rangle \preceq g,$$

where g has type $\mathcal{M} \to [0,1]$. Such a judgment is *valid*, written $\models \langle\!\langle\, c_1, c_2 \,\rangle\!\rangle \preceq g$, iff

$$\forall m \bullet\ \Delta_m(c_1, c_2) \leq g(m).$$

Figure 3.1 presents the main rules of the logic; contrary to the logic of Sections 3.2.2 and 5.2, this logic is not restricted to constant functions g. The logic deals with programs that are structurally similar, and as shown by Lemma 3.2, supports a rule for reasoning about adversaries.

$$\models \langle\!\langle\, x \leftarrow e, x \leftarrow e \,\rangle\!\rangle \preceq \lambda m.\ 0 \qquad \models \langle\!\langle\, \mathsf{assert}\ b, \mathsf{assert}\ b \,\rangle\!\rangle \preceq \lambda m.\ 0$$

$$\frac{\forall m \bullet\ \Delta\left([\![d_1]\!]_{\mathcal{DE}}\ m, [\![d_2]\!]_{\mathcal{DE}}\ m\right) \leq g(m)}{\models \langle\!\langle\, x \xleftarrow{\$} d_1, x \xleftarrow{\$} d_2 \,\rangle\!\rangle \preceq g}$$

$$\frac{\models \langle\!\langle\, c_1, c_1' \,\rangle\!\rangle \preceq g_1 \quad \models \langle\!\langle\, c_2, c_2' \,\rangle\!\rangle \preceq g_2 \qquad g(m) \stackrel{\text{def}}{=} \textbf{if}\ [\![b]\!]_{\mathcal{E}}\ m\ \textbf{then}\ g_1(m)\ \textbf{else}\ g_2(m)}{\models \langle\!\langle\, \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \mathsf{if}\ b\ \mathsf{then}\ c_1'\ \mathsf{else}\ c_2' \,\rangle\!\rangle \preceq g}$$

$$\frac{\models \langle\!\langle\, c_1, c_2 \,\rangle\!\rangle \preceq g \qquad g_0(m) \stackrel{\text{def}}{=} 0 \quad g_{n+1}(m) \stackrel{\text{def}}{=} \textbf{if}\ [\![b]\!]_{\mathcal{E}}\ m\ \textbf{then}\ ([\![c_1]\!]\ m)(g_n) + g(m)\ \textbf{else}\ 0}{\models \langle\!\langle\, \mathsf{while}\ b\ \mathsf{do}\ c_1, \mathsf{while}\ b\ \mathsf{do}\ c_2 \,\rangle\!\rangle \preceq \lambda m.\ \mathrm{lub}\,(\lambda n.\ g_n(m))}$$

$$\frac{\models \langle\!\langle\, E_1(p).\mathsf{body}, E_2(p).\mathsf{body} \,\rangle\!\rangle \preceq g \qquad g =_X g \quad \forall x \bullet\ x \in X \Rightarrow \mathsf{global}(x)}{\models \langle\!\langle\, y \leftarrow p(x), y \leftarrow p(x) \,\rangle\!\rangle \preceq g}$$

$$\models \langle\!\langle\, \mathsf{skip}, \mathsf{skip} \,\rangle\!\rangle \preceq \lambda m.\ 0 \qquad \frac{\models \langle\!\langle\, c_1, c_2 \,\rangle\!\rangle \preceq g \quad \models \langle\!\langle\, c_1', c_2' \,\rangle\!\rangle \preceq g'}{\models \langle\!\langle\, c_1; c_1', c_2; c_2' \,\rangle\!\rangle \preceq \lambda m.\ ([\![c_1]\!]\ m)(g') + g(m)}$$

Figure 3.1: Logic to bound the statistical distance between two probabilistic programs.

To prove the soundness, for instance, of the rule for sequential composition, we introduce an intermediate program $c_1; c_2'$ (where c_1 is executed in environment E_1 and c_2' in environment E_2) and prove that the distance between $[\![c_1; c_1']\!]\ m$ and $[\![c_1; c_2']\!]\ m$ is bounded by $([\![c_1]\!]\ m)(g')$, while the distance between $[\![c_1; c_2']\!]\ m$ and $[\![c_2; c_2']\!]\ m$ is bounded by $g(m)$. The rule for loops relies on the characterization of the semantics of a while loop as the supremum of its n-th unrolling $[\mathsf{while}\ e\ \mathsf{do}\ c]_n$ (see Figure 2.1), and on the auxiliary rule

$$\frac{\models \langle\!\langle\, [\mathsf{while}\ b\ \mathsf{do}\ c_1]_n, [\mathsf{while}\ b\ \mathsf{do}\ c_2]_n \,\rangle\!\rangle \preceq g_n}{\models \langle\!\langle\, \mathsf{while}\ b\ \mathsf{do}\ c_1, \mathsf{while}\ b\ \mathsf{do}\ c_2 \,\rangle\!\rangle \preceq \lambda m.\ \mathrm{lub}\,(\lambda n.\ g_n(m))}$$

The rule for procedure calls builds on the fact that the semantics of a call $y \leftarrow p(x)$ in memory m is basically given by an unfolding of p's code, where the resulting command is executed in a memory m' that only differs from m in the set of (local) variables that correspond to p's formal parameters; global variables have the same values in m and m'. The last two premises of the rule are thus required to guarantee that $g(m') = g(m)$.

While the rules in Figure 3.1 are sufficient to reason about closed programs, they do not allow to reason about programs in the presence of adversaries. We enhance the logic with a rule that allows drawing conclusions of the form[1] $\models (\!| \mathcal{A}, \mathcal{A} |\!) \preceq g$, i.e. to bound the statistical distance between calls to an adversary $\mathcal{A}$ executed in two different environments E_1 and E_2.

Although the code of $\mathcal{A}$ is unknown, the only statements in his code that can increase statistical distance are calls to oracles (since these are the only instructions whose semantics may vary between the two environments E_1 and E_2). The rule we formalize captures this intuition, by providing an upper bound for the statistical distance between the final distributions in terms of the statistical distance induced by individual oracle calls, and the number of oracle calls made by the adversary. In its simplest formulation, the rule assumes that oracles are instrumented with a counter that keeps track of the number of queries made by the adversary, and that the statistical distance between the distributions induced by a call to an oracle $x \leftarrow \mathcal{O}(\vec{e})$ in E_1 and E_2 is upper-bounded by some constant δ, i.e. $\models (\!| \mathcal{O}, \mathcal{O} |\!) \preceq \lambda m.\ \delta$. In this case, the statistical distance between calls to the adversary $\mathcal{A}$ in E_1 and E_2 is upper-bounded by $q \cdot \delta$, where q is an upper bound on the number of oracle calls made.

For the application presented in Section 3.3, we need to formalize a more expressive rule, in which the statistical distance between two oracle calls can also depend on the program state. Moreover, we allow the counter to be any integer expression, and only require that it does not decrease across oracle calls.

Lemma 3.2 (Adversary Rule)**.** *Let $\mathcal{A}$ be an adversary and let* cnt *be an integer expression whose variables are global and cannot be written by $\mathcal{A}$. Let $h : \mathbb{N} \to [0,1]$ and define*

$$\bar{h}_{\mathsf{cnt}}(m, m') \stackrel{\text{def}}{=} \min \left(1, \sum_{i=[\![\mathsf{cnt}]\!]_{\mathcal{E}}\, m}^{[\![\mathsf{cnt}]\!]_{\mathcal{E}}\, m'-1} h(i) \right)$$

Assume that for every oracle $\mathcal{O}$,

$$\models (\!| \mathcal{O}, \mathcal{O} |\!) \preceq \lambda m.\ ([\![E_1(\mathcal{O}).\mathsf{body}]\!]\, m)(\lambda m'.\ \bar{h}_{\mathsf{cnt}}(m, m'))$$

and moreover,

$$\{\mathsf{cnt} = i\}\ E_1(\mathcal{O}).\mathsf{body}\ \{\mathsf{cnt} \geq i\}.$$

Then,

$$\models (\!| \mathcal{A}, \mathcal{A} |\!) \preceq \lambda m.\ ([\![E_1(\mathcal{A}).\mathsf{body}]\!]\, m)\left(\lambda m'.\ \bar{h}_{\mathsf{cnt}}(m, m')\right).$$

[1] For the sake of readability, we write $\models (\!| \mathcal{A}, \mathcal{A} |\!) \preceq g$ instead of $\models (\!| x \leftarrow \mathcal{A}(\vec{e}), x \leftarrow \mathcal{A}(\vec{e}) |\!) \preceq g$, and likewise for oracles.

The first hypothesis states that the distance $\Delta_m(E_1(\mathcal{O}).\mathsf{body}, E_2(\mathcal{O}).\mathsf{body})$ can be bounded in terms of the value of cnt before and after executing $\mathcal{O}$; for instance, if a call to $\mathcal{O}$ always increments cnt by 2, then the distance $\Delta_m(E_1(\mathcal{O}).\mathsf{body}, E_2(\mathcal{O}).\mathsf{body})$ is bounded by $h([\![\mathsf{cnt}]\!]_{\mathcal{E}}\ m) + h([\![\mathsf{cnt}]\!]_{\mathcal{E}}\ m+1)$. The second hypothesis captures the monotonicity property of the counter.

Proof. As in [Barthe *et al.*, 2010], the rule is derived from the induction principle induced by the definition of well-formed adversary using the rules in Figure 3.1. ■

Failure events constitute a major reasoning pattern in cryptographic game-based proofs. We now build on this reasoning pattern to derive a related rule of the logic.

Transitions based on failure events allow transforming a program into another program that is semantically equivalent unless some *failure* condition is triggered. This kind of program transformations rely on the following lemma:

Lemma 3.3 (Fundamental Lemma of the Game-Playing Technique)**.** *Consider two programs c_1, c_2 and let A, B, and F be events. For every initial memory m, if*

$$\Pr[c_1(m) : A \wedge \neg F] = \Pr[c_2(m) : B \wedge \neg F],$$

then

$$|\Pr[c_1(m) : A] - \Pr[c_2(m) : B]| \leq \max\{\Pr[c_1(m) : F], \Pr[c_2(m) : F]\}.$$

When $A = B$ and $F = \mathbf{bad}$ for some Boolean variable **bad**, the hypothesis of the lemma can be automatically established by inspecting the code of c_1 and c_2: it holds if their code differs only after program points setting **bad** to true and **bad** is never reset to false. In this case we say that c_1 and c_2 are *upto-bad* programs.

As a corollary of Lemma 3.3 and in view of Lemma 3.1 one can derive the following result.

Lemma 3.4 (Upto-bad Rule)**.** *Let c_1 and c_2 be a pair of upto-bad programs such that c_2 is lossless. Then,*

$$\models (\!| c_1, c_2 |\!) \preceq \lambda m.\ \Pr[c_2(m) : \mathbf{bad}].$$

The proof of Lemma 3.4 also relies on the fact that for upto-bad programs c_1 and c_2, $\Pr[c_1(m) : bad] \leq \Pr[c_2(m) : bad]$ whenever c_2 is lossless.

3.2.2 Approximate Observational Equivalence

Approximate observational equivalence generalizes observational equivalence between two programs by allowing that their output distributions differ up to some quantity δ. Informally, two programs c_1 and c_2 are δ-observationally equivalent w.r.t. an input set of variables I and an output set of variables O iff for every pair of memories m_1, m_2 coinciding on I, the statistical distance between the quotient distributions $([\![c_1]\!]\ m_1)/=_O$ and $([\![c_2]\!]\ m_2)/=_O$ over $\mathcal{M}/=_O$ is upper-bounded by δ. For the purpose of formalization, it is more convenient to rely on the following alternative characterization that does not use quotient distributions.

Definition 3.2 (Approximate Observational Equivalence)**.** *Two programs c_1 and c_2 are* δ-observationally equivalent *w.r.t. an input set of variables I and an output set of variables O, written* $\models c_1 \simeq_O^I c_2 \preceq \delta$*, iff for all memories $m_1, m_2 \in \mathcal{M}$ and functions $f_1, f_2 : \mathcal{M} \to [0,1]$,*

$$m_1 =_I m_2 \wedge f_1 =_O f_2 \implies \left|([\![c_1]\!]\, m_1)(f_1) - ([\![c_2]\!]\, m_2)(f_2)\right| \leq \delta.$$

Figure 3.2 provides an excerpt of an equational theory for approximate observational equivalence; further and more general rules appear in the Coq development.

$$\frac{\models c_1 \simeq_O^I c_2 \preceq \delta_1 \quad \models c_2 \simeq_O^I c_3 \preceq \delta_2}{\models c_1 \simeq_O^I c_3 \preceq \delta_1 + \delta_2}\ [\mathsf{trans}] \qquad \frac{\models c_1 \simeq_{O'}^I c_2 \preceq \delta_1 \quad \models c_1' \simeq_O^{O'} c_2' \preceq \delta_2}{\models c_1; c_1' \simeq_O^I c_2; c_2' \preceq \delta_1 + \delta_2}\ [\mathsf{seq}]$$

$$\frac{\models c_1 \simeq_{O'}^{I'} c_2 \preceq \delta' \quad I' \subseteq I \quad O \subseteq O' \quad \delta' \leq \delta}{\models c_1 \simeq_O^I c_2 \preceq \delta}\ [\mathsf{weak}]$$

$$\frac{\models c_1 \simeq_O^I c_1' \preceq \delta \quad \models c_2 \simeq_O^I c_2' \preceq \delta \quad =_I \implies b\langle 1\rangle = b'\langle 2\rangle}{\models \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \simeq_O^I \mathsf{if}\ b'\ \mathsf{then}\ c_1'\ \mathsf{else}\ c_2' \preceq \delta}\ [\mathsf{cond}]$$

$$\frac{\forall m_1, m_2 \bullet\ m_1 =_I m_2 \implies \Delta\left([\![d_1]\!]_{\mathcal{DE}}\, m_1, [\![d_2]\!]_{\mathcal{DE}}\, m_2\right) \leq \delta}{\models x \xleftarrow{\$} d_1 \simeq_{I\cup\{x\}}^I x \xleftarrow{\$} d_2 \preceq \delta}\ [\mathsf{rand}]$$

Figure 3.2: Selected rules for reasoning about approximate observational equivalence.

Most rules generalize observational equivalence in the expected way. For instance, the rule for random assignments considers the case of two distribution expressions δ-away from each other. Let μ_1 and μ_2 be their interpretation. In case $\mu_1 = \mu_2$, one obtains $\delta = 0$. Furthermore, if μ_1 and μ_2 are uniform distributions over subsets $A_1, A_2 \subseteq T$ of some underlying type, one has $\Delta(\mu_1, \mu_2) = \max\{{}^{|A_1 \setminus A_2|}/_{|A_1|},\ {}^{|A_2 \setminus A_1|}/_{|A_2|}\}$.

As an illustrative example we sketch the soundness proof of the rule for random assignments. Consider $m_1, m_2 \in \mathcal{M}$ and $f_1, f_2 : \mathcal{M} \to [0,1]$ such that $m_1 =_I m_2$ and $f_1 =_{I\cup\{x\}} f_2$. Now define $S_1 = \{a \mid ([\![d_1]\!]\, m_1)(a) > 0\}$, $g_1(a) = f_1(m_1\,\{a/x\})$ and analogously for S_2 and g_2, $A_0 = (S_1 \cap S_2 \cap A_{\mu_1 \geq \mu_2}) \cup (S_1 \setminus S_2)$ and $A_1 = (S_1 \cap S_2 \cap A_{\mu_1 < \mu_2}) \cup (S_2 \setminus S_1)$. Then we have

$$\begin{aligned}
&\left|([\![x \xleftarrow{\$} d_1]\!]\, m_1)(f_1) - ([\![x \xleftarrow{\$} d_2]\!]\, m_2)(f_2)\right| \\
&\quad \overset{(1)}{=} \left|\sum_{a\in S_1} \mu_1(a)\, g_1(a) - \sum_{a\in S_2} \mu_2(a)\, g_2(a)\right| \\
&\quad = \left|\sum_{a\in S_1\cap S_2} \mu_1(a)\, g_1(a) - \mu_2(a)\, g_2(a) + \sum_{a\in S_1\setminus S_2} \mu_1(a)\, g_1(a) - \sum_{a\in S_2\setminus S_1} \mu_2(a)\, g_2(a)\right| \\
&\quad \overset{(2)}{=} \left|\sum_{a\in A_0} (\mu_1(a) - \mu_2(a))\, g_1(a) - \sum_{a\in A_1} (\mu_2(a) - \mu_1(a))\, g_2(a)\right|
\end{aligned}$$

$$\begin{aligned}
&\overset{(3)}{\leq} \max\left\{ \sum_{a\in A_0} (\mu_1(a) - \mu_2(a))\, g_1(a), \sum_{a\in A_1} (\mu_2(a) - \mu_1(a))\, g_2(a) \right\} \\
&\overset{(4)}{\leq} \max\left\{ \sum_{a\in A_0} \mu_1(a) - \mu_2(a), \sum_{a\in A_1} \mu_2(a) - \mu_1(a) \right\} \\
&\leq \Delta\left(\mu_1, \mu_2\right).
\end{aligned}$$

Equation (1) follows from unfolding the semantics of random assignments; equality (2) follows from the definitions of A_0, A_1, S_1 and S_2, and the fact that $g_1(a) = g_2(a)$ for all $a \in S_1 \cap S_2$. Finally inequality (3) holds since for all reals x, y satisfying $0 \leq x, y \leq \delta$ one has $|x - y| \leq \delta$ and inequality (4) holds because the max operator is monotonic (in its both arguments).

We highlight that the notion of (approximate) observational equivalence subsumes that of (approximate) non-interference (see Section 1.1); concretely, a program c is δ-non-interferent if $\models c \simeq_L^L c \preceq \delta$, where L stands for the set of public (or low sensitivity) variables. Therefore, the logic of Figure 3.2 finds applications in approximate information flow analysis, besides the case study about indifferentiability described in Section 3.3.

3.2.2.1 A Conditional Variant

The application we describe in Section 3.3 requires reasoning about conditional approximate observational equivalence, a generalization of approximate observational equivalence. We define for a distribution μ and event P the conditional distribution $\mu|_P$ as

$$\mu|_P \overset{\text{def}}{=} \lambda f.\ \mu\left(\lambda a.\ \frac{f(a)\,\mathbb{1}_P(a)}{\mu(\mathbb{1}_P)}\right).$$

Intuitively, $\mu|_P\ \mathbb{1}_Q$ yields the conditional probability of Q given P.

Definition 3.3 (Conditional Approximate Observational Equivalence)**.** *A program* c_1 conditioned *on predicate* P_1 *is* δ-observationally equivalent *to a program* c_2 conditioned *on* P_2 *w.r.t. an input set of variables* I *and an output set of variables* O*, written* $\models [c_1]_{P_1} \simeq_O^I [c_2]_{P_2} \preceq \delta$*, iff for any* $m_1, m_2 \in \mathcal{M}$ *and* $f_1, f_2 : \mathcal{M} \to [0,1]$*,*

$$m_1 =_I m_2 \ \wedge\ f_1 =_O f_2 \implies \left|(\llbracket c_1 \rrbracket\, m_1)|_{P_1}(f_1) - (\llbracket c_2 \rrbracket\, m_2)|_{P_2}(f_2)\right| \leq \delta.$$

Remark. At first glance it might seem possible to define conditional observational equivalence as a specialization of standard observational equivalence by interpreting judgment $\models [c_1]_{P_1} \simeq_O^I [c_2]_{P_2} \preceq \delta$ as $\models c_1; \mathsf{assert}\ P_1 \simeq_O^I c_2; \mathsf{assert}\ P_2 \preceq \delta$. However, this is not the case. To see why, observe that both distributions $(\llbracket c \rrbracket\, m)|_P$ and $\llbracket c; \mathsf{assert}\ P \rrbracket\, m$ assign the same—null—probability to those final memories that do not satisfy P, but for those final memories that do satisfy P, the former includes a normalizing factor $1/(\llbracket c \rrbracket\, m)(\mathbb{1}_P)$, while the latter does not.

Conditional approximate observational equivalence subsumes classic approximate observational equivalence, which can be recovered by taking $P_1 = P_2 = \mathsf{true}$.

3.3 Indifferentiable Hash Functions into Elliptic Curves

A customary technique to design cryptosystems that are both efficient and provable secure is to assume that some of their components are ideal, for instance, that they have a "truly" random output. Such kind of ideal components are called *random oracles* and the technique, coined *random oracle model* (ROM), was put in solid grounds by Bellare & Rogaway [1993].

When implementing such cryptosystems, random oracles are instantiated by concrete hash functions; the concept of indifferentiability [Maurer *et al.*, 2004] allows rigorously justifying this substitution (and more generally, the substitution of any idealized component by a concrete implementation): a cryptosystem that is provable secure in the ROM remains secure if the random oracle is replaced by a hash function that is indifferentiable from the random oracle.

Although the ROM has been under fierce criticism [Canetti *et al.*, 2004] and the indifferentiability framework turns out to be weaker than initially believed [Fleischmann *et al.*, 2010; Ristenpart *et al.*, 2011], it is generally accepted that proofs in these models provide some evidence that a system is secure. Not coincidentally, all finalists in the NIST Cryptographic Hash Algorithm competition have been proved indifferentiable from a random oracle.

A class of hash functions particularly useful in the domain of cryptography are those mapping values onto the group induced by an elliptic curve. In general, elliptic curve cryptography allows building efficient public-key cryptographic systems with comparatively short keys and as such is an attractive solution for resource-constrained applications. In contrast to other approaches to public-key cryptography, where candidates to instantiate random oracles into bitstrings, residue classes, or finite fields have been known for some time, constructions of random oracles into ordinary elliptic curves have remained elusive.

In 2010, Brier *et al.* [2010] proposed the first generic construction indifferentiable from a random oracle into elliptic curves. This construction is of practical significance since, as discussed above, it allows securely implementing elliptic curve cryptosystems. In this section we present a machine-checked proof of the security of this construction using the tools described in Section 3.2. The proof builds additionally on several large developments (including Théry & Hanrot [2007]'s formalization of elliptic curves and Gonthier *et al.* [2007]'s formalization of finite groups) and demonstrates that the verified security technique is apt to support proofs involving advanced algebraic and number-theoretical reasoning.

3.3.1 Construction of Indifferentiable Hash Functions

Let us recall the notion of indifferentiability from a random oracle. First we explain briefly the notion of random oracle and then we recall the definition of indifferentiability.

A *random oracle* is an ideal primitive that maps elements in some domain into uniformly and independently distributed values in a finite set; queries are answered consistently so that identical queries are given the same answer. A proof conducted in the ROM for a

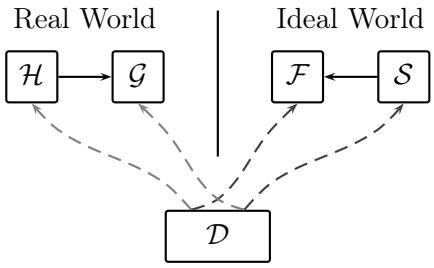


Figure 3.3: The two scenarios in the definition of indifferentiability of construction $\mathcal{H}$ from an ideal primitive $\mathcal{F}$

function $\mathcal{H}$ assumes that $\mathcal{H}$ is a random oracle and makes it publicly available to all parties. In games, random oracles are represented as stateful procedures.

Definition 3.4 (Indifferentiability)**.** *A construction $\mathcal{H}$ built from a primitive $\mathcal{G}$ is said to be* $(t_{\mathcal{S}}, t_{\mathcal{D}}, q_1, q_2, \delta)$-indifferentiable *from an ideal primitive $\mathcal{F}$ iff there exists a simulator $\mathcal{S}$ with oracle access to $\mathcal{F}$ such that any distinguisher $\mathcal{D}$ running within time $t_{\mathcal{D}}$ has at most probability δ of distinguishing a scenario where it is given $\mathcal{H}$ and $\mathcal{G}$ as oracles from a scenario where it is given $\mathcal{F}$ and $\mathcal{S}$ instead:*

$$\left|\Pr\left[b \leftarrow \mathcal{D}^{\mathcal{H}^{\mathcal{G}},\mathcal{G}}() : b = \mathsf{true}\right] - \Pr\left[b \leftarrow \mathcal{D}^{\mathcal{F},\mathcal{S}^{\mathcal{F}}}() : b = \mathsf{true}\right]\right| \leq \delta.$$

The distinguisher $\mathcal{D}$ is allowed to make at most q_1 queries to $\mathcal{H}$ (resp. $\mathcal{F}$) and at most q_2 queries to $\mathcal{G}$ (resp. $\mathcal{S}$).

Intuitively, $\mathcal{S}$ must simulate the primitive $\mathcal{G}$ so that no distinguisher can tell whether it is interacting with $\mathcal{H}^{\mathcal{G}}$ and $\mathcal{G}$ or with $\mathcal{F}$ and $\mathcal{S}^{\mathcal{F}}$ (see Figure 3.3). The simulator must do so without access to the internal state (if any) of $\mathcal{F}$ or to its interaction with the distinguisher.

Random oracles into elliptic curves over finite fields are typically built from a random oracle $\mathcal{G}$ on the underlying field and a deterministic encoding f that maps elements of the field into the elliptic curve. Examples of such encodings include Icart's function [Icart, 2009] and the Shallue-Woestijne-Ulas (SWU) algorithm [Shallue & van de Woestijne, 2006]. In general (and for the aforementioned mappings) the function f is not surjective and only covers a fraction of points in the curve. Hence, the naive definition of a hash function H as $f \circ \mathcal{G}$ would not cover the whole curve, contradicting the assumption that H behaves as a random oracle.

In a recent paper, Brier *et al.* [2010] show how to build hash functions into elliptic curves that are indifferentiable from a random oracle for a particular class of encodings, including both SWU and Icart's encodings. They prove that if $(\mathbb{G}, \otimes)$ is a finite cyclic group of order N with generator g, a function into $\mathbb{G}$ indifferentiable from a random oracle can be built from any polynomially invertible function $f : A \to \mathbb{G}$ and hash functions

$\mathcal{G}_1 : \{0,1\}^* \to A$ and $\mathcal{G}_2 : \{0,1\}^* \to \mathbb{Z}_N$ that behave as random oracles as follows:

$$\mathcal{H}(m) \stackrel{\text{def}}{=} f(\mathcal{G}_1(m)) \otimes g^{\mathcal{G}_2(m)}.$$

Intuitively, the term $g^{\mathcal{G}_2(m)}$ behaves as a one-time pad and ensures that $\mathcal{H}$ covers all points in the group even if f covers only a fraction. This construction can be seen as the composition of the function $F(a,p) = f(a) \otimes g^p$ and a random oracle into $A \times \mathbb{Z}_N$.

We prove in CertiCrypt the indifferentiability of a generalization of Brier et al.'s construction to finitely generated abelian groups. The proof introduces two intermediate constructions and is structured in three steps:

i) We first prove that any efficiently invertible encoding f can be turned into a *weak encoding* (Theorem 3.5);
ii) We then show an efficient construction to transform any weak encoding f into an *admissible encoding* (Theorem 3.6);
iii) Finally, we prove that any admissible encoding can be turned into a hash function indifferentiable from a random oracle (Theorem 3.7).

Moreover, we show in Section 3.3.2 that Icart's and SWU encodings are efficiently invertible and thus yield hash functions indifferentiable from a random oracle when plugged in into the above construction.

We recall the alternative definitions of weak and admissible encoding from [Icart, 2010]. Note that these do not match the definitions of Brier *et al.* [2010], but, in comparison, are better behaved: e.g. admissible encodings as we define them are closed under functional composition and cartesian product. In the reminder, for any finite set A, the instruction $x \stackrel{\$}{\leftarrow} A$ assigns to x a value uniformly chosen from A.

Definition 3.5 (Weak encoding)**.** *A function $f : S \to R$ is an (α, δ)-*weak encoding *iff it is computable in polynomial-time and there exists a probabilistic polynomial-time algorithm $\mathcal{I}_f : R \to S_\perp$ such that*

i) $\{\mathsf{true}\}\ r \stackrel{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r)\ \{s = \perp \vee f(s) = r\}$;
ii) $\models \left[r \stackrel{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r)\right]_{s \neq \perp} \simeq^{\emptyset}_{\{s\}} [s \stackrel{\$}{\leftarrow} S] \preceq \delta$;
iii) $\Pr\left[r \stackrel{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r) : s = \perp\right] \leq 1 - \alpha^{-1}$.

Condition i) states that $\mathcal{I}_f$ either inverts f or fails; ii) states that given a random input, $\mathcal{I}_f$ returns a pre-image chosen almost uniformly when it does not fail and iii) states that $\mathcal{I}_f$ does not fail too often.

Definition 3.6 (Admissible encoding)**.** *A function $f : S \to R$ is an δ-*admissible encoding *if it is computable in polynomial-time and there exists a probabilistic polynomial-time algorithm $\mathcal{I}_f : R \to S_\perp$ such that*

i) $\{\mathsf{true}\}\ r \stackrel{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r)\ \{s = \perp \vee f(s) = r\}$;
ii) $\models r \stackrel{\$}{\leftarrow} R;\ s \leftarrow \mathcal{I}_f(r) \simeq^{\emptyset}_{\{s\}} s \stackrel{\$}{\leftarrow} S \preceq \delta$.

Compared to a weak encoding, an admissible encoding may seem to impose no explicit bound on the probability of $\mathcal{I}_f$ failing. However, condition ii) requires the output of inverter $\mathcal{I}_f$ on a random input be statistically indistinguishable from the uniform distribution on S; the bound δ must account for the probability of $\mathcal{I}_f$ failing.

We begin our proof by showing that any efficiently invertible encoding is a weak encoding.

Theorem 3.5. *Let $f : S \to R$ be a function computable in polynomial-time such that for any $r \in R$, $\left|f^{-1}(r)\right| \leq B$. Assume there exists a polynomial-time algorithm $\mathcal{I}$ that given $r \in R$ outputs the set $f^{-1}(r)$. Then, f is an $(\alpha, 0)$-weak encoding, with $\alpha = B\,|R|\,/\,|S|$.*

Proof. Using $\mathcal{I}$, we build a partial inverter $\mathcal{I}_f : R \to S_\perp$ of f that satisfies the properties in Definition 3.5:

$$\begin{aligned}\mathcal{I}_f(r): \quad & X \leftarrow \mathcal{I}(r);\ b \xleftarrow{\$} \mathsf{true} \oplus_{|X|/B} \mathsf{false};\\ & \mathsf{if}\ b = \mathsf{true}\ \mathsf{then}\ s \xleftarrow{\$} X;\ \mathsf{return}\ s\ \mathsf{else\ return}\ \perp\end{aligned}$$

In the above construction of $\mathcal{I}_f$, $\mathsf{true} \oplus_\beta \mathsf{false}$ denotes the Bernoulli distribution with success probability β, so that the instruction $b \xleftarrow{\$} \mathsf{true} \oplus_\beta \mathsf{false}$ assigns true to b with probability β.

First observe that $\mathcal{I}_f(r)$ fails with probability $1 - \left|f^{-1}(r)\right|/B$ or else returns an element uniformly chosen from the set of pre-images of r, and thus satisfies the first property trivially. In addition we have

$$\Pr\left[r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) : s \neq \perp\right] = \sum_{r \in R} \frac{1}{|R|} \frac{\left|f^{-1}(r)\right|}{B} = \frac{|S|}{B\,|R|},$$

and for any $x \in S$,

$$\Pr\left[r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) : s = x\right] = \sum_{r \in R} \frac{1}{|R|} \frac{\left|f^{-1}(r)\right|}{B} \frac{\mathbb{1}_{f^{-1}(r)}(x)}{|f^{-1}(r)|} = \frac{1}{B\,|R|}.$$

Hence, for a uniformly chosen r, the probability of $\mathcal{I}_f(r)$ failing is exactly $1 - \alpha^{-1}$, and the probability of returning any particular value in S conditioned to not failing is uniform. ■

We show next how to construct an admissible encoding $F : A \times P \to Q$ from a weak encoding $f : A \to Q$ for appropriate sets P and Q. This construction generalizes Brier *et al.*'s original result (see [Brier *et al.*, 2010, Theorem 3]) that restricts the analysis to the case where Q is a cyclic group of order N and P is $\mathbb{Z}_N$.

The proof that we present relies on the application of two padding lemmas involving a pair of expressions of type P and Q. To capture these properties we introduce a novel algebraic structure coined *padding algebra*.

Definition 3.7 (Padding algebra)**.** *Let P and Q be two finite sets equipped with binary operations $\circledast, \odot : P \times Q \to Q$ and $\oslash : Q \times Q \to P$. We say that $(P, Q, \circledast, \odot, \oslash)$ is a* padding algebra *iff:*

i) $(p \circledast q) \oslash q = p$ *for all $p \in P, q \in Q$;*

ii) $q \oslash (p \odot q) = p \quad$ *for all* $p \in P, q \in Q$;
iii) *for all* $q \in Q$, *the function* $\lambda p.\ p \circledast q$ *is an isomorphism between* P *and* Q;
iv) *for all* $q \in Q$, *the function* $\lambda p.\ p \odot q$ *is an isomorphism between* P *and* Q.

For such a structure one can prove the following algebraic equivalences:

$$\models p \xleftarrow{\$} P;\ q_2 \leftarrow p \circledast q_1 \simeq^{\{q_1\}}_{\{q_1,q_2,p\}} q_2 \xleftarrow{\$} Q;\ p \leftarrow q_2 \oslash q_1, \tag{3.2}$$

$$\models q_2 \xleftarrow{\$} Q;\ p \leftarrow q_1 \oslash q_2 \simeq^{\{q_1\}}_{\{q_1,q_2,p\}} p \xleftarrow{\$} P;\ q_2 \leftarrow p \odot q_1. \tag{3.3}$$

To do so we need to rely on the rules for random assignments and sequential composition [rnd] and [seq] of CertiCrypt's logic (see Figure 2.3) and on the above set of axioms—axioms i) and iii) for the former equivalence and axioms ii) and iv) for the latter. In Section 3.3.2 we show that every finite abelian group induces a padding algebra and use this fact to instantiate the results presented in this section.

We now show how to turn a weak encoding $f : A \to Q$ into an admissible encoding $F : A \times P \to Q$ when P and Q can be given the structure of a padding algebra.

Theorem 3.6. *Let* $(P, Q, \circledast, \odot, \oslash)$ *be a padding algebra such that*

$$(q_1 \oslash q_2) \odot q_1 = q_2 \qquad \forall q_1, q_2 \in Q,$$

and operations $\circledast$ *and* $\odot$ *can be computed in polynomial-time. Then, for any* (α, δ)*-weak encoding* $f : A \to Q$, *the function*

$$\begin{array}{lcl} F & : & A \times P \to Q \\ F(a,p) & \stackrel{\text{def}}{=} & p \odot f(a) \end{array}$$

is an δ'*-admissible encoding into* Q, *with* $\delta' = \delta + \left(1 - \alpha^{-1}\right)^{T+1}$ *for any value* T *polynomial in the security parameter.*

Proof. Since f is a weak encoding, there exists a polynomial-time computable inverter $\mathcal{I}_f$ of f satisfying the conditions in Definition 3.5. Let T be polynomial in the security parameter. Using $\mathcal{I}_f$, we build a partial inverter $\mathcal{I}_F$ of F that satisfies the properties in Definition 3.6:

$\mathcal{I}_F(q)$:
$\quad i \leftarrow 0;\ a \leftarrow \bot;$
$\quad$ while $(i \leq T \wedge a = \bot)$ do
$\quad\quad p \xleftarrow{\$} P;$
$\quad\quad x \leftarrow p \circledast q;$
$\quad\quad a \leftarrow \mathcal{I}_f(x);$
$\quad\quad i \leftarrow i + 1$
$\quad$ end;
$\quad$ if $a \neq \bot$ then return (a, p) else return $\bot$

The partial inverter $\mathcal{I}_F$ runs in time $t_{\mathcal{I}_F} = (T+1)\, t_{\mathcal{I}_f}$, where $t_{\mathcal{I}_f}$ is a bound on the running time of $\mathcal{I}_f$. Hence, $\mathcal{I}_F$ is polynomial-time.

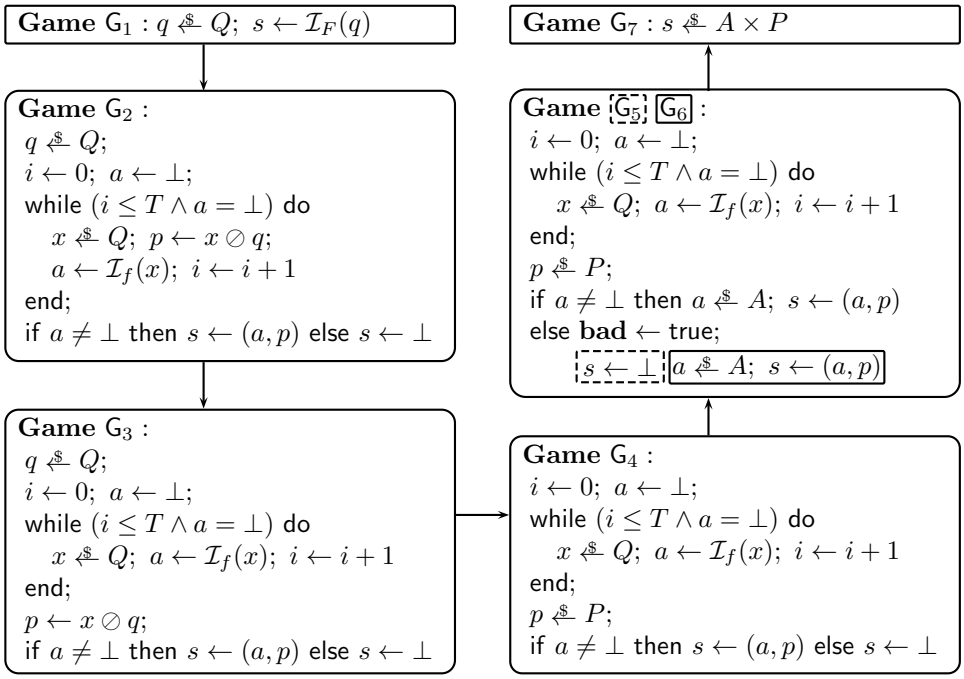


Figure 3.4: Games used in the proof of Theorem 3.6.

We prove that

$$\models q \xleftarrow{\$} Q;\ s \leftarrow \mathcal{I}_F(q) \simeq^{\emptyset}_{\{s\}} s \xleftarrow{\$} A \times P \preceq \delta'$$

using the sequence of games $\mathsf{G}_1, \ldots, \mathsf{G}_7$ shown in Figure 3.4, the mechanized program transformations of CertiCrypt, and the proof rules for observational and approximate observational equivalence. We briefly describe the proof below.

We obtain game G_2 by first inlining the call to $\mathcal{I}_F$ in the initial game and then applying the algebraic equivalence (3.2) to transform the body of the while loop.

Game G_3 is obtained by moving the assignment to p outside the loop in game G_2. This transformation is semantics-preserving because p is never used inside the loop and the value that it has when exiting the loop only depends on the value of x in the last iteration. Formally, this is proven by unfolding the first iteration of the loop and establishing that the relation

$$=_{\{i,x,a,q\}} \wedge\ (p = x \oslash q)\langle 1 \rangle$$

is a relational invariant between the loop in G_2 and the loop resulting from removing the assignment to p. By appending $p \leftarrow x \oslash q$ to the latter loop, we recover equivalence on p.

Observe that games G_2 and G_3 use operation $\oslash$, which might not be computable in polynomial-time (for elliptic curve groups, it would require computing a discrete logarithm). This is a valid proof technique that does not undermine the validity of the analysis; we prove all necessary equivalences.

Since q is no longer used inside the loop, we can postpone choosing it after the loop and use the algebraic equivalence (3.3) to sample p instead of q. We obtain G_4 by additionally removing the assignment to q, which is now dead code.

For the next step in the proof we use the fact that f is a weak encoding and therefore the distribution of a after a call $a \leftarrow \mathcal{I}_f(x)$ conditioned to $a \neq \bot$ is δ-away from the uniform distribution. This allows us to re-sample the value of a after the loop, provided $a \neq \bot$, incurring a penalty δ on the statistical distance of the distribution of s between G_4 and G_5. To prove this formally, let b be the condition of the loop and c its body. Observe that the semantics of the loop coincides with the semantics of its $(T+1)$-unrolling $[\mathsf{while}\ b\ \mathsf{do}\ c]_{T+1}$. We show by induction on T that for any $[0,1]$-valued functions f and g such that $f =_{\{a'\}} g$,

$$m_1 =_{\{a,i\}} m_2 \ \wedge\ m_1(a) = \bot \implies \left|([\![c_1]\!]\, m_1)(f') - ([\![c_2]\!]\, m_2)(g')\right| \leq \delta,$$

where

$$\begin{array}{rcl} c_1 & \stackrel{\text{def}}{=} & [\mathsf{while}\ b\ \mathsf{do}\ c]_{T+1};\ \mathsf{if}\ a \neq \bot\ \mathsf{then}\ a' \leftarrow a \\ c_2 & \stackrel{\text{def}}{=} & [\mathsf{while}\ b\ \mathsf{do}\ c]_{T+1};\ \mathsf{if}\ a \neq \bot\ \mathsf{then}\ a' \xleftarrow{\$} A \\ f'(m) & \stackrel{\text{def}}{=} & \textbf{if}\ m(a) \neq \bot\ \textbf{then}\ f(m)\ \textbf{else}\ 0 \\ g'(m) & \stackrel{\text{def}}{=} & \textbf{if}\ m(a) \neq \bot\ \textbf{then}\ g(m)\ \textbf{else}\ 0 \end{array}$$

and use this to conclude the δ-approximate equivalence of G_4 and G_5.

Since G_5 and G_6 are syntactically equivalent except for code appearing after the flag **bad** is set, we apply Lemma 3.4 to obtain the bound

$$\models (\!|\, \mathsf{G}_5, \mathsf{G}_6 \,|\!) \preceq \Pr\left[\mathsf{G}_5 : \textbf{bad}\right].$$

Since the probability of failure of $\mathcal{I}_f$ on a uniformly chosen input is upper-bounded by $1 - \alpha^{-1}$, we can show by induction on T that

$$\Pr\left[\mathsf{G}_5 : \textbf{bad}\right] \leq \left(1 - \alpha^{-1}\right)^{T+1},$$

from which we conclude $\models \mathsf{G}_5 \simeq^{\emptyset}_{\{s\}} \mathsf{G}_6 \preceq \left(1 - \alpha^{-1}\right)^{T+1}$.

By coalescing the branches in the conditional at the end of G_6 and removing dead code, we prove that the game is observational equivalent w.r.t. a and p to the game $a \xleftarrow{\$} A;\ p \xleftarrow{\$} P;\ s \leftarrow (a, p)$, which is trivially equivalent to G_7.

By composing the above results, we conclude

$$\models \mathsf{G}_1 \simeq^{\emptyset}_{\{s\}} \mathsf{G}_7 \preceq \delta + \left(1 - \alpha^{-1}\right)^{T+1}.$$

We must also show that $s = \bot \vee F(s) = q$ is a post-condition of G_1. As G_1 and G_3 are observationally equivalent with respect to s and q, it is sufficient to establish the validity of the post-condition for G_3. We show that $a \neq \bot \Rightarrow x = f(a)$ is an invariant of the loop. When the loop finishes, either $a = \bot$ and in this case $s = \bot$, or $a \neq \bot$ and we have $F(s) = p \odot f(a) = (x \oslash q) \odot x = q$. ■

Finally, we show that the composition of an admissible encoding $f : S \to R$ and a random oracle into S is indifferentiable from a random oracle into R.

Theorem 3.7. *Let $f : S \to R$ be an δ-admissible encoding with inverter algorithm $\mathcal{I}_f$ and let $\mathcal{G} : \{0,1\}^* \to S$ be a random oracle. Then, $f \circ \mathcal{G}$ is $(t_{\mathcal{S}}, t_{\mathcal{D}}, q_1, q_2, \delta')$-indifferentiable from a random oracle into R, where $t_{\mathcal{S}} = q_1\ t_{\mathcal{I}_f}$ and $\delta' = 2(q_1 + q_2)\delta$.*

Before moving to the proof of Theorem 3.7, we prove the following useful result.

Proposition 3.8. *Let $f : S \to R$ be an δ-admissible encoding with inverter algorithm $\mathcal{I}_f$. Then*

$$\models s \xleftarrow{\$} S;\ r \leftarrow f(s) \simeq^{\emptyset}_{\{r,s\}} r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r) \preceq 2\delta.$$

Proof. Define

$$\begin{aligned}
c_i &= s \xleftarrow{\$} S;\ r \leftarrow f(s)\\
c_f &= r \xleftarrow{\$} R;\ s \leftarrow \mathcal{I}_f(r)\\
c_1 &= c_i;\ \mathsf{if}\ s = \bot\ \mathsf{then}\ r \xleftarrow{\$} R\ \mathsf{else}\ r \leftarrow f(s)\\
c_2 &= c_f;\ \mathsf{if}\ s = \bot\ \mathsf{then}\ \mathbf{bad} \leftarrow \mathsf{true};\ r \xleftarrow{\$} R\ \mathsf{else}\ r \leftarrow f(s)\\
c_3 &= c_f;\ \mathsf{if}\ s = \bot\ \mathsf{then}\ \mathbf{bad} \leftarrow \mathsf{true}\ \mathsf{else}\ r \leftarrow f(s)
\end{aligned}$$

Since the first branch of the conditional in c_1 is never executed, we have

$$\models c_i \simeq^{\emptyset}_{\{r,s\}} c_1.$$

Due to the second property of Definition 3.6, the distributions of s after executing c_i and c_f are δ-away. Using the rules for approximate observational equivalence, we obtain

$$\models c_1 \simeq^{\emptyset}_{\{r,s\}} c_2 \preceq \delta.$$

Lemma 3.4 implies that $\models (\!| c_2, c_3 |\!) \preceq \Pr[c_2 : \mathbf{bad}]$. Moreover,

$$\Pr[c_2 : \mathbf{bad}] = 1 - \Pr\left[c_f : s \neq \bot\right] = \Pr[s \xleftarrow{\$} S : s \neq \bot] - \Pr\left[c_f : s \neq \bot\right] \leq \delta.$$

where the last inequality holds again because of the second property of Definition 3.6. Since the final values of r and s in programs c_2 and c_3 are independent of the initial memory, we have

$$\models c_2 \simeq^{\emptyset}_{\{r,s\}} c_3 \preceq \delta.$$

Because $\mathcal{I}_f$ is a partial inverter for f, the else branch of the conditional in c_3 has no effect and can be removed, and thus $\models c_3 \simeq^{\emptyset}_{\{r,s\}} c_f$. We conclude by transitivity of approximate observational equivalence. ■

Proof of Theorem 3.7. Let $\mathcal{D}$ be a distinguisher against the indifferentiability of $\mathcal{H} = f \circ \mathcal{G}$ from a random oracle $\mathcal{F}$ into R. We show that the simulator $\mathcal{S}$ constructed as $\mathcal{I}_f \circ \mathcal{F}$ is good enough, i.e. $\mathcal{D}$ cannot distinguish with probability greater than δ' between a game G

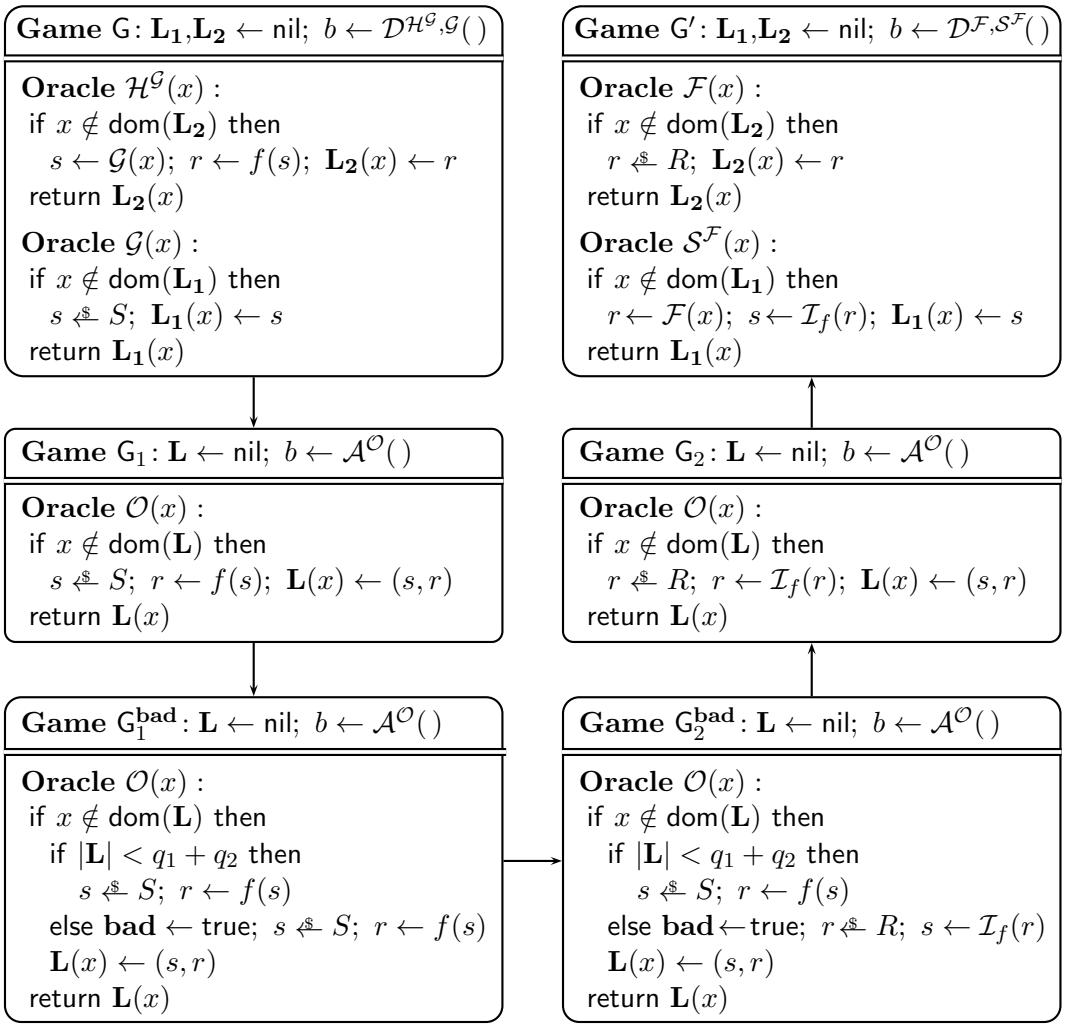


Figure 3.5: Games used in the proof of Theorem 3.7.

where it is given $\mathcal{H}$ and $\mathcal{G}$ as oracles and a game G' where it is given $\mathcal{F}$ and $\mathcal{S}$ instead. An overview of the proof, including these two games is shown in Figure 3.5.

Our goal is to prove

$$\left|\Pr\left[\mathsf{G} : b = \mathsf{true}\right] - \Pr\left[\mathsf{G}' : b = \mathsf{true}\right]\right| \leq 2(q_1 + q_2)\delta.$$

The crux of the proof is an application of Lemma 3.2. In order to apply it, we need first to transform game G (resp. G') to replace oracles $\mathcal{H}$ and $\mathcal{G}$ (resp. $\mathcal{F}$ and $\mathcal{S}$) by a single joint oracle that simultaneously returns the responses of both. Using $\mathcal{D}$, we construct an adversary $\mathcal{A}$ with access to a single joint oracle, such that game G (resp. G') is equivalent to game G_1 (resp. G_2) in the figure. Adversary $\mathcal{A}$ simply calls the distinguisher $\mathcal{D}$ and

forwards the value it returns; it simulates $\mathcal{H}$ and $\mathcal{G}$ (resp. $\mathcal{F}$ and $\mathcal{S}$) by using its own oracle $\mathcal{O}$.

We assume, as Brier *et al.* [2010] do, that whenever the distinguisher makes a query x to one of its oracles, it also makes the same query to its other oracle. Under this assumption, game G is equivalent to G_1, and game G' is equivalent to G_2. We thus have

$$\Pr\left[\mathsf{G} : b = \mathsf{true}\right] = \Pr\left[\mathsf{G}_1 : b = \mathsf{true}\right] \quad \text{and} \quad \Pr\left[\mathsf{G}' : b = \mathsf{true}\right] = \Pr\left[\mathsf{G}_2 : b = \mathsf{true}\right].$$

Furthermore, since $\mathcal{D}$ makes at most q_1 and q_2 queries to each of its oracles, $\mathcal{A}$ makes at most $q = q_1 + q_2$ queries to its joint oracle.

We next transform the implementation of oracle $\mathcal{O}$ in game G_1 so that its behavior after the first q queries is the same as in G_2. This transformation will pave the way to applying Lemma 3.2. The desired behavior of oracle $\mathcal{O}$ is represented in game $\mathsf{G}_2^{\mathbf{bad}}$. Observe that $\mathsf{G}_2^{\mathbf{bad}}$ is annotated with a flag **bad** that is set to true when the allotted number of queries is reached. This is because we essentially rely on Lemma 3.3 to justify the equivalence between G_1 and $\mathsf{G}_2^{\mathbf{bad}}$. To apply this lemma we need however to introduce an intermediate game $\mathsf{G}_1^{\mathbf{bad}}$. Since G_1 and $\mathsf{G}_1^{\mathbf{bad}}$ are trivially equivalent, we have

$$\Pr\left[\mathsf{G}_1 : b = \mathsf{true}\right] = \Pr\left[\mathsf{G}_1^{\mathbf{bad}} : b = \mathsf{true}\right]$$

An application of Lemma 3.3 between games $\mathsf{G}_1^{\mathbf{bad}}$ and $\mathsf{G}_2^{\mathbf{bad}}$ gives

$$\Pr\left[\mathsf{G}_1^{\mathbf{bad}} : b = \mathsf{true} \wedge \neg\mathbf{bad}\right] = \Pr\left[\mathsf{G}_2^{\mathbf{bad}} : b = \mathsf{true} \wedge \neg\mathbf{bad}\right].$$

But since $\mathbf{bad} \implies q < |\mathbf{L}|$ is an invariant and $|\mathbf{L}| \leq q$ a post-condition of both $\mathsf{G}_1^{\mathbf{bad}}$ and $\mathsf{G}_2^{\mathbf{bad}}$, we have

$$\Pr\left[\mathsf{G}_1^{\mathbf{bad}} : b = \mathsf{true}\right] = \Pr\left[\mathsf{G}_2^{\mathbf{bad}} : b = \mathsf{true}\right].$$

We can now apply Lemma 3.2 to the games G_2 and $\mathsf{G}_2^{\mathbf{bad}}$, defining $\mathsf{cnt} = |\mathbf{L}|$ and $h(i) = \mathbf{if}\ i < q\ \mathbf{then}\ 2\delta\ \mathbf{else}\ 0$. The second hypothesis of the lemma, i.e. that a call to $E_2(\mathcal{O})$ cannot decrease $|\mathbf{L}|$, is immediate. We can assume that $2q\delta < 1$ (otherwise the theorem is trivially true). Then for any pair of initial and final memories m and m',

$$\sum_{i=[\![\mathsf{cnt}]\!]_{\mathcal{E}}\, m}^{[\![\mathsf{cnt}]\!]_{\mathcal{E}}\, m'-1} h(i) \leq 2q\delta < 1, \qquad \text{and} \qquad \bar{h}_{\mathsf{cnt}}(m, m') = \sum_{i=[\![\mathsf{cnt}]\!]_{\mathcal{E}}\, m}^{[\![\mathsf{cnt}]\!]_{\mathcal{E}}\, m'-1} h(i).$$

We are only left to prove that

$$\models (\!| E_2(\mathcal{O}).\mathsf{body}, E_2^{\mathbf{bad}}(\mathcal{O}).\mathsf{body} |\!) \preceq \lambda m.\ ([\![E_2(\mathcal{O}).\mathsf{body}]\!]\ m)(\lambda m'.\ \bar{h}_{\mathsf{cnt}}(m, m')).$$

A case analysis on the conditions $x \in \mathsf{dom}(\mathbf{L})$ and $|\mathbf{L}| < q$ yields three cases; two of them yield a null distance and are immediate. The remaining case, where $x \notin \mathsf{dom}(\mathbf{L})$

and $|\mathbf{L}| < q$, yields a distance of 2δ and follows from Proposition 3.8. We finally obtain $\models (\!| \mathsf{G}_2, \mathsf{G}_2^{\mathbf{bad}} |\!) \preceq 2(q_1 + q_2)\delta$, which entails

$$\left| \Pr\left[\mathsf{G}_2 : b = \mathsf{true}\right] - \Pr\left[\mathsf{G}_2^{\mathbf{bad}} : b = \mathsf{true}\right] \right| \leq 2(q_1 + q_2)\delta.$$

This, combined with the previous results implies the desired inequality. ■

3.3.2 Application to Elliptic Curves

We now discuss the instantiation of the proof presented in the previous section to hashing into elliptic curves. We proceed in two steps. First, we show that every finite abelian group with an efficiently computable law can be given the structure of a padding algebra that satisfies the hypotheses of Theorem 3.6. It follows that any efficiently invertible encoding into such a group induces a hash function onto the group that is indifferentiable from a random oracle. Second, we show that the set of points of an elliptic curve form a finite abelian group with an efficiently computable law, and that Icart's function is an efficiently invertible encoding. The formalization of both steps in the Coq proof assistant relies on independently developed libraries of mathematics. In addition, it assumes standard mathematical results that are not formalized in Coq (e.g. Cassels' theorem).

We begin with some mathematical background, before describing the Coq formalization.

3.3.2.1 Mathematical Background

Fundamental theorem of finite groups. The fundamental theorem of finite groups states that every finite abelian group $(\mathbb{G}, \otimes)$ is isomorphic to a product of cyclic groups $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$. Moreover the decomposition can be made unique by fixing additional conditions on $n_1 \ldots n_k$. Assume that g_i is a generator of the group $\mathbb{Z}_{n_i}$, for all $1 \leq i \leq k$. Then for every group element $x \in \mathbb{G}$ there exists a unique vector $(z_1, \ldots, z_k) \in \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$ such that

$$x = g_1^{z_1} \otimes \cdots \otimes g_k^{z_k}$$

In the sequel, we use $\log x$ to denote $(z_1, \ldots, z_k)$ and $\vec{g}^{\,\vec{z}}$ to denote $g_1^{z_1} \otimes \cdots \otimes g_k^{z_k}$, as above.

Elliptic Curves over Finite Fields. Recall that the *order* of a finite field is of the form p^m, where p is a prime number called the *characteristic* of the field, and that finite fields of the same order are unique up to isomorphism. In the following, we let $\mathbb{F}_n$ refer to any finite field of order n (i.e. containing n elements).

For our purposes, it is sufficient to consider elliptic curves over fields of order p^m with $p > 3$, that is, finite fields of characteristic different from 2 and 3. For the sake of readability, we specialize all our definitions to this case. Let $a, b \in \mathbb{F}_{p^m}$ such that $4a^3 + 27b^2 \neq 0$. The elliptic curve induced by a and b contains all points $(X, Y) \in \mathbb{F}_{p^m} \times \mathbb{F}_{p^m}$ that satisfy the *Weierstrass equation*

$$Y^2 = X^3 + aX + b.$$

Note that the condition on a and b is equivalent to requiring that the polynomial X^3+aX+b has distinct roots, and ensures that the curve is non-singular (i.e. it has no cusps or self-intersections) and is thus a proper elliptic curve.

The set of points of an elliptic curve can be given the structure of an abelian group by adding an additional "idealized" *point at infinity* $\mathcal{O}$:

$$\mathbb{E}_{a,b}(\mathbb{F}_{p^m}) = \{(X,Y) \in \mathbb{F}_{p^m} \times \mathbb{F}_{p^m} \mid Y^2 = X^3 + aX + b\} \cup \{\mathcal{O}\}$$

The point $\mathcal{O}$ behaves as the identity; the inverse of an element is given by equations

$$-\mathcal{O} \stackrel{\text{def}}{=} \mathcal{O} \qquad \text{and} \qquad -(X,Y) \stackrel{\text{def}}{=} (X,-Y).$$

The definition of the group law rests on the following property of elliptic curves, which follows from Bézout's theorem [Hartshorne, 1977, Corollary 7.8]: The line defined by two points P_1 and P_2 in a curve[2] intersects the curve at a third point P_3 (which might coincide with P_1 or P_2). The group law is defined as

$$P_1 \oplus P_2 \stackrel{\text{def}}{=} -P_3.$$

For an in-depth algebraic description of elliptic curves we refer the interested reader to [Hankerson *et al.*, 2004] or [Silverman, 2009].

The construction of hash functions onto elliptic curves exploits the particular group structure of this kind of curves; Cassels' theorem [Hankerson *et al.*, 2004, Theorem 3.12] states that every elliptic curve $\mathbb{E}_{a,b}(\mathbb{F}_n)$ over a finite field $\mathbb{F}_n$ is isomorphic to the product of the two cyclic groups $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ where n_1 and n_2 are uniquely determined; moreover n_2 divides both n_1 and $n-1$. As an immediate corollary, observe that $\left|\mathbb{E}_{a,b}(\mathbb{F}_n)\right| = n_1 n_2$, and that the group is cyclic when $n_2 = 1$.

Encodings into elliptic curves. Brier *et al.* [2010] present two encodings into elliptic curves: Icart's function [Icart, 2009] and (a simplified version of) the Shallue-Woestijne-Ulas (SWU) algorithm [Shallue & van de Woestijne, 2006]. We review their definition and prove that they can be computed and inverted in polynomial-time.

Icart's encoding. Let $p > 3$ be a prime such that $p^m \equiv 2 \pmod 3$. Icart's function $f_{a,b} : \mathbb{F}_{p^m} \to \mathbb{E}_{a,b}(\mathbb{F}_{p^m})$ is defined as:

$$f_{a,b}(t) \stackrel{\text{def}}{=} \begin{cases} (x, tx+v) & \text{if } t \neq 0 \\ ((-b)^{1/3}, 0) & \text{if } t = 0 \wedge a = 0 \\ \mathcal{O} & \text{if } t = 0 \wedge a \neq 0 \end{cases}$$

$$\text{where} \qquad x = \left(v^2 - b - \frac{t^6}{27}\right)^{\frac{1}{3}} + \frac{t^2}{3} \qquad v = \frac{3a - t^4}{6t}$$

[2] If $P_1 = P_2 = (X_0, Y_0)$ we let $\{(X,Y) \in \mathbb{F}_n \times \mathbb{F}_n \mid \alpha(X - X_0) + \beta(Y - Y_0) = 0\}$ be the line defined by P_1 and P_2, where $\alpha = 3X_0^2 + a$ and $\beta = -2Y_0$.

As a side remark, observe that the original definition only deals with the case $a \neq 0$; the definition for the case $a = 0$ was suggested to us by Icart in a private communication.

One can prove by computation that the image of t through $f_{a,b}$ belongs to the curve $Y^2 = X^3 + aX + b$ for every t in $\mathbb{F}_{p^m}$. Moreover, the set of pre-images under Icart's function of a point in the curve can be characterized as the set of roots of a polynomial over $\mathbb{F}_{p^m}$:

$$f_{a,b}^{-1}(\mathcal{O}) = \begin{cases} \{0\} & \text{if } a \neq 0 \\ \emptyset & \text{if } a = 0 \end{cases}$$

$$f_{a,b}^{-1}(X,Y) = \begin{cases} \{t \mid t^3 - 6tX + 6Y = 0\} & \text{if } a = 0 \\ \{t \mid t^4 - 6t^2X + 6tY = 3a\} & \text{if } a \neq 0 \end{cases}$$

Since the polynomials are of degree at most 4, every point in the curve has at most 4 pre-images. Moreover, the pre-images can be computed in polynomial-time using algorithms for factoring polynomials over finite fields, e.g. Berlekamp's algorithm. Thus, Icart's encoding is polynomially invertible.

The Shallue-Woestijne-Ulas (SWU) encoding. Let $p > 3$ be a prime such that $p^m \equiv 3 \pmod 4$ and let $a, b \neq 0$ belong to $\mathbb{F}_{p^m}$. We use $g(X)$ as a shorthand for the polynomial $X^3 + aX + b$. For every $t \in \mathbb{F}_{p^m}$ we let

$$X_1(t) \stackrel{\text{def}}{=} -ba^{-1}\left(1 + (t^4 - t^2)^{-1}\right) \qquad X_2(t) \stackrel{\text{def}}{=} -t^2\, X_1(t) \qquad U(t) \stackrel{\text{def}}{=} t^3\, g(X_1(t))$$

Note that $U(t)^2 = -g(X_1(t))\, g(X_2(t))$. As -1 is a quadratic non-residue in $\mathbb{F}_{p^m}$, it follows that exactly one of $g(X_1(t))$ and $g(X_2(t))$ is a square and thus either $X_1(t)$ or $X_2(t)$ is the abscissa of a point on the curve $Y^2 = g(X)$ [Fouque & Tibouchi, 2010]. This motivates the definition of the SWU function $f'_{a,b}$ as follows:

$$f'_{a,b}(t) \stackrel{\text{def}}{=} \begin{cases} \left(X_1(t),\, g(X_1(t))^{1/2}\right) & \text{if } g(X_1(t)) \text{ is a square} \\ \left(X_2(t),\, g(X_2(t))^{1/2}\right) & \text{otherwise} \end{cases}$$

To compute the pre-images t of a point (X, Y) we need to solve equations $X_1(t) = X$ and $X_2(t) = X$, keeping the solutions that verify $g(X_1(t)) = Y^2$ and $g(X_2(t)) = Y^2$ respectively. Each of the constraints $X_1(t) = X$ and $X_2(t) = X$ can be reduced to a polynomial equation and the inverse of the SWU function can be computed as

$$f'^{-1}_{a,b}(\mathcal{O}) = \emptyset$$

$$f'^{-1}_{a,b}(X,Y) = \left\{t \mid -\alpha_1 t^4 + \alpha_1 t^2 - ba^{-1} = 0 \ \wedge\ g(X_1(t)) = Y^2\right\} \cup \left\{t \mid -ba^{-1}t^4 + \alpha_2 t^2 - \alpha_2 = 0 \ \wedge\ g(X_2(t)) = Y^2\right\}$$

where $\alpha_1 = ba^{-1} + X$ and $\alpha_2 = ba^{-1} - X$. Since each polynomial has degree 4 and for every t, exactly one of $g(X_1(t))$ and $g(X_2(t))$ is a square, every point in the curve admits at most

4 pre-images. As with Icart's function, they can be computed in polynomial-time using, for instance, the Berlekamp's algorithm. Therefore, the SWU encoding can be inverted in polynomial time.

Finally we claim that both encodings can be computed in polynomial time. This basically amounts to verifying that square (in the case of the SWU function) and cubic roots (in the case of Icart's function) can be computed in polynomial-time, which is in turn entailed by the restrictions imposed on the order p^m of the field. More precisely, for every $x \in \mathbb{F}_{p^m}$, condition $p^m \equiv 2 \pmod 3$ entails formula $x^{1/3} = x^{2p^m-1}$ while condition $p^m \equiv 3 \pmod 4$ implies that $x^{1/2} = x^{(p^m+1)/4}$.

3.3.2.2 Formalization in Coq

To instantiate our generic proof of indifferentiability to Icart's encoding, we proceeded as follows:

i) We showed that every finite abelian group $\mathbb{G}$ can be given the structure of a padding algebra. The formalization exploits the fundamental theorem of finite groups to decompose $\mathbb{G}$ as a product of cyclic groups $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$. We set $P = \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$, $Q = \mathbb{G}$ and

$$\vec{z} \circledast x = x \otimes \vec{g}^{\,-\vec{z}} \qquad \vec{z} \odot x = x \otimes \vec{g}^{\,\vec{z}} \qquad x_1 \oslash x_2 = \log\left({x_1}^{-1} \otimes x_2\right).$$

Our formalization relies on the SSReflect standard library [Gonthier *et al.*, 2007], which provides a wealth of results on finite abelian groups, including the fundamental theorem of finite groups.

It follows from Theorems 3.5, 3.6 and 3.7 that one can build a hash function onto $\mathbb{G}$ from a polynomially invertible function $f : A \to \mathbb{G}$ and random oracles $\mathcal{G}_1 : \{0,1\}^* \to A$ and $\mathcal{G}_2 : \{0,1\}^* \to \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$.

Lemma 3.9 (Indifferentiable Hashing into Finite Abelian Groups)**.** *Let $\mathbb{G}$ be a finite abelian group with an efficiently computable law. Let $\mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$ be the decomposition of $\mathbb{G}$ as a product of cyclic groups and let g_i be a generator of $\mathbb{Z}_{n_i}$ for $i = 1 \ldots k$. Assume that $f : A \to \mathbb{G}$ is a polynomial-time function such that for each x, the set $f^{-1}(x)$ can be computed by a probabilistic polynomial-time algorithm $\mathcal{I}_f$ and its size is bounded by B. Then, for any pair of random oracles $\mathcal{G}_1 : \{0,1\}^* \to A$ and $\mathcal{G}_2 : \{0,1\}^* \to \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$ and any value T polynomial in the security parameter, the construction*

$$\mathcal{H}(m) \stackrel{\text{def}}{=} f(\mathcal{G}_1(m)) \otimes \vec{g}^{\,\mathcal{G}_2(m)}$$

is $(t_{\mathcal{S}}, t_{\mathcal{D}}, q_1, q_2, \delta)$-indifferentiable from a random oracle into $\mathbb{G}$, where $t_{\mathcal{S}} = q_1(T+1)t_{\mathcal{I}_f}$ and $\delta = 2(q_1+q_2)\left(1 - |A| / (B\,|\mathbb{G}|)\right)^{T+1}$.

ii) We showed that the set of points of the elliptic curve $\mathbb{E}_{a,b}(\mathbb{F}_{p^m})$ can be construed as a finite abelian group. To carry out this step, we adapted Théry & Hanrot [2007]'s

formalization of elliptic curves to match the definition of finite group used by SSReflect. As a result, one obtains an instantiation of Lemma 3.9 to elliptic curves (the fact that the group law is efficiently computable is assumed).

Given its complexity, we have not attempted to formalize the proof of Cassels' theorem. However, Bartzia [2011] is working towards completing a formalization of Cassels' theorem. Since their formalization is based on the SSReflect library of finite groups, it would be immediate to use it for specializing the lemma further. Specifically, one could define $\mathcal{G}_{2,1}$ and $\mathcal{G}_{2,2}$ as the first and second projections of $\mathcal{G}_2$ and let

$$\mathcal{H}(m) \stackrel{\text{def}}{=} f(\mathcal{G}_1(m)) \otimes g_1{}^{\mathcal{G}_{2,1}(m)} \otimes g_2{}^{\mathcal{G}_{2,2}(m)}.$$

Under the previous assumptions, the function H is indifferentiable from a random oracle.

iii) We defined Icart's function $f_{a,b}$, and showed that it generates points in the curve $\mathbb{E}_{a,b}$. This required showing the existence of cubic roots in the field $\mathbb{F}_{p^m}$ when $p^m \equiv 2 \pmod 3$. Moreover, we defined the inverse of Icart's function, and assumed that it is polynomially computable. Discharging this assumption would require to show the existence of an efficient method for factoring polynomials over the underlying field, and is left as future work (Berlekamp's algorithm, for instance, computes the roots of a polynomial of degree d over field $\mathbb{F}_n$ in time $\mathcal{O}(d^2 \log^3 n)$). It then follows from Lemma 3.9 that $f_{a,b}$ induces a hash function onto $\mathbb{E}_{a,b}(\mathbb{F}_{p^m})$ that is $(t_{\mathcal{S}}, t_{\mathcal{D}}, q_1, q_2, 2(q_1+q_2)\delta)$-indifferentiable from a random oracle into $\mathbb{E}_{a,b}(\mathbb{F}_{p^m})$, where $t_{\mathcal{S}} = q_1$ $t_{\mathcal{I}_F} = q_1(T+1)t_{f^{-1}}$ and $t_{f^{-1}}$ is an upper bound on the time needed to compute the pre-image of a point under Icart's function, i.e. to solve a polynomial of degree 4 in $\mathbb{F}_{p^m}$.

We could also instantiate Lemma 3.9 to the SWU encoding. However, this would require showing that equality $U(t)^2 = -g(X_1(t))g(X_2(t))$ entails that either $X_1(t)$ or $X_2(t)$ is the abscissa of point on the curve $Y^2 = g(X)$, which involves some reasoning about quadratic residues. Nowak [2009]'s formalization of quadratic residues would be a good starting point.

Overall, the formalization consists of over 65,000 lines of Coq (without counting components reused from the standard libraries of Coq and SSReflect), which break down as follows: 45,000 lines corresponding to the original CertiCrypt framework, 3,500 lines of extensions to CertiCrypt, 7,000 lines written originally for our application to indifferentiability, and 10,000 lines of a slightly adapted version of Théry & Hanrot [2007]'s elliptic curve library.

We conclude with the observation that our development in the current section points to some underdeveloped areas in formalized mathematics. Although there have been substantial efforts to develop machine-checked libraries of mathematics, covering relevant topics such as polynomials and finite fields, the libraries lack many important results. For instance, we are not aware of any formalization of factorization algorithms for polynomials over finite fields. Since such algorithms, and in particular Berlekamp's algorithm, are

used by many computer algebra systems, we believe that it would be of general interest to provide a machine-checked proof of their correctness. Moreover, elliptic curve theory is a fascinating area of mathematics, and it would be particularly appealing to develop formalizations of some of the most important results in the area.

4

Security Analysis based on the α-distance

The aim of this chapter is to present a full-fledged relational Hoare logic for approximate reasoning between probabilistic programs. This program logic is primarily inspired by applications in privacy-preserving data analysis, even though it can be used to model a wider class of quantitative confidentiality properties such as probabilistic non-interference. We begin by reviewing differential privacy, the policy for private data analysis that motivates our development.

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] has emerged as a compelling confidentiality policy that achieves an attractive trade-off between these two conflicting requirements.

Loosely speaking, differential privacy captures the idea that joining an statistical database should not increase the risks to the contributors' privacy. To achieve this goal, Dwork *et al.* suggested that algorithms mining the database should be probabilistic and their outputs should be insensitive to the contribution of any single individual. Formally, we say that a randomized computation or *mechanism* c is ϵ-*differentially private* iff for any two input databases D_1 and D_2 differing in at most one row[1],

$$\Pr[c(D_1) : P] \leq e^{\epsilon} \Pr[c(D_2) : P]$$

[1] We assume that each row in the database corresponds to the contribution of an individual.

for every event P on the output domain of c. Sometimes differential privacy is hard to achieve in practice and we have to resort to a weaker notion of confidentiality known as *approximate* differential privacy. In these cases, the above equation is relaxed with a slack δ, i.e.

$$\Pr[c(D_1) : P] \leq e^{\epsilon} \Pr[c(D_2) : P] + \delta,$$

and we say that c is (ϵ, δ)-*differentially private*.

In this chapter we report on α-pRHL, an approximate relational Hoare logic for reasoning about (standard and approximate) differential privacy. α-pRHL can be viewed as a quantitative extension of the native logic pRHL of CertiCrypt that allows modeling approximate properties of probabilistic programs. In particular, it allows reasoning about the statistical distance between probabilistic programs and subsumes the equational theory of Chapter 3 used to reason about approximate observational equivalence.

Moreover, we extend CertiCrypt to provide complete tool support for α-pRHL. In doing so, we present a proof system to derive valid α-pRHL judgments and provide a machine-checked proof in Coq of its soundness. The resulting framework allows constructing fully formalized proofs about, e.g. differential privacy, and crisply extends the frontiers of the verified security methodology.

Conceptually, our development of α-pRHL for reasoning about differential privacy comprises two major steps.

i) First we introduce a notion of distance between distributions (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 e^{ϵ}-distance between the output distributions obtained by running c on two adjacent memories; in the dissertation we consider a more general view of differential privacy where we assume given a (discrete) metric over the input domain of mechanisms, and the adjacency relation is stated as an upper bound of 1 on this metric.

ii) Then, we introduce our relational Hoare logic α-pRHL, whose judgment have the form

$$c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi, \tag{4.1}$$

where c_1 and c_2 are probabilistic programs, Ψ and Φ are binary relations over program states, $\alpha \in \mathbb{R}^{\geq 1}$ and $\delta \in [0, 1]$. We establish the connection of α-pRHL with differential privacy by showing that judgments with the equality on programs as post-condition yield (α, δ)-closeness conditions between the output of programs. Concretely, if Φ represents the equality on program states, the validity of judgment (4.1) implies that δ is an upper bound for the α-distance between the output distribution of c_1 and c_2 when executed on two initial states related by Ψ. As a corollary, if we let $c_1 = c_2 = c$, $\alpha = e^{\epsilon}$, Ψ represent the adjacency relation and Φ represent the equality on program states, judgment (4.1) entails that c is (ϵ, δ)-differentially private.

Before outlining our development in details, let us review how the rest of the chapter is organized. In Section 4.1 we present the semantic foundations of α-pRHL; in particular we

introduce the α-distance between distributions and demonstrate how the notion of differential privacy can be stated in terms of it. In Section 4.2 we show that α-pRHL judgments characterize differential privacy and provide a detailed overview of the α-pRHL proof system. In Section 4.3 we report on several case studies that we have fully formalized in our framework. Finally, we sketch pencil-and-paper proofs of all our results in Appendix 4.A.

4.1 Preliminaries

4.1.1 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 $\alpha \geq 1$, which will later play the role of the factor e^ϵ in the definition of differential privacy. Namely, we define the α-distance $\Delta_\alpha^\blacklozenge(a, b)$ between a and b as

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

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

Definition 4.1 (α-distance)**.** *For $\alpha \in \mathbb{R}^{\geq 1}$, the α-distance $\Delta_\alpha^\blacklozenge(\mu_1, \mu_2)$ between two distributions μ_1 and μ_2 in $\mathcal{D}(A)$ is defined as*

$$\Delta_\alpha^\blacklozenge(\mu_1, \mu_2) \stackrel{\text{def}}{=} \sup_{f : A \to \{0,1\}} \Delta_\alpha^\blacklozenge(\mu_1(f), \mu_2(f)).$$

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

Observe that α-distance generalizes the notion of statistical distance from Section 3.1, which is recovered by taking $\alpha = 1$.

The definition of α-distance considers only Boolean-valued functions, i.e. those corresponding to characteristic functions of events. The next lemma shows that for discrete distributions this definition is equivalent to an alternative definition that considers all $[0, 1]$-valued functions.

Lemma 4.1. *For all distributions μ_1 and μ_2 over a discrete set A,*

$$\Delta_\alpha^\blacklozenge(\mu_1, \mu_2) = \sup_{f : A \to [0,1]} \Delta_\alpha^\blacklozenge(\mu_1(f), \mu_2(f)).$$

This characterization of α-distance generalizes that of statistical distance given in Lemma 3.1. Likewise, this is the characterization that we adopt for our Coq development.

We now state some basic properties of α-distance; these properties are the keystone for reasoning about relation lifting that is used to give meaning to our α-pRHL judgments. All properties are implicitly universally quantified.

Lemma 4.2 (Basic Properties of α-distance)**.**

i) $0 \leq \Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) \leq 1$ *and* $\Delta^{\blacklozenge}_{\alpha}(\mu, \mu) = 0$*;*
ii) $\Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) = \Delta^{\blacklozenge}_{\alpha}(\mu_2, \mu_1)$*;*
iii) $\Delta^{\blacklozenge}_{\alpha\alpha'}(\mu_1, \mu_3) \leq \max\{\alpha' \, \Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) + \Delta^{\blacklozenge}_{\alpha'}(\mu_2, \mu_3)\,,$
$\Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) + \alpha \, \Delta^{\blacklozenge}_{\alpha'}(\mu_2, \mu_3)\}$;
iv) $\alpha \leq \alpha' \implies \Delta^{\blacklozenge}_{\alpha'}(\mu_1, \mu_2) \leq \Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2)$*;*
v) $\Delta^{\blacklozenge}_{\alpha}(\mathsf{bind}\ \mu_1\ M, \mathsf{bind}\ \mu_2\ M) \leq \Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2)$.

Most of the above properties are self-explanatory; we briefly highlight the most important ones. Property iii) generalizes the triangle inequality with appropriate skew factors; iv) states that α-distance is anti-monotonic with respect to α; v) states that probabilistic computations do not increase the distance (which is a well-known fact for statistical distance); a particularly useful specialization of this property is obtained when μ_1 and μ_2 are distributions over a product space and M represent their left or right projections, i.e.

$$\Delta^{\blacklozenge}_{\alpha}(\pi_1(\mu_1), \pi_1(\mu_2)) \leq \Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) \qquad \text{and} \qquad \Delta^{\blacklozenge}_{\alpha}(\pi_2(\mu_1), \pi_2(\mu_2)) \leq \Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2)\,.$$

Lemma 4.17 in the appendix further generalizes v). In contrast to statistical distance, α-distance does not satisfy the identity of indiscernibles, i.e. one may have $\Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) = 0$ and $\mu_1 \neq \mu_2$.

4.1.2 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 e^{ϵ}. *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.

Definition 4.2 (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*

$$\forall a, a' \in A \bullet\ d(a, a') \leq 1 \implies \Delta^{\blacklozenge}_{e^{\epsilon}}\big(M(a), M(a')\big) \leq \delta.$$

For algorithms that terminate with probability 1 (i.e. when $\Pr[M(a) : \mathsf{true}] = 1$ 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 4.2 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 4.2 for $\delta = 0$ and $\epsilon \geq \ln(2)$. However, for any ϵ,

$$\frac{1}{2} = 1 - \Pr\left[M(a') : \mathsf{true}\right] > e^{\epsilon}\left(1 - \Pr\left[M(a) : \mathsf{true}\right]\right) = 0,$$

which would violate privacy when an adversary can observe non-termination (note that $1 - \Pr[M(a) : \mathsf{true}]$ and $1 - \Pr[M(a') : \mathsf{true}]$ represent the probability of non-termination of M on inputs a and a').

A termination-sensitive definition of differential privacy can be obtained by considering in Definition 4.2 the extension $M_\perp$ of M to $B_\perp = B \cup \{\perp\}$, letting $\Pr[M_\perp(a) = \perp] \stackrel{\text{def}}{=} 1 - \Pr[M(a) : \mathsf{true}]$. As the following lemma shows, we can account for differences in termination on adjacent inputs by shifting these differences to the additive slack.

Lemma 4.3. *Let $M : A \to \mathcal{D}(B)$ be an (ϵ, δ)-differentially private algorithm. Then, $M_\perp$ is $(\epsilon, \delta + \delta')$-differentially private, where*

$$\delta' \stackrel{\text{def}}{=} \sup_{a,a' \mid d(a,a') \leq 1} \left|\Pr\left[M(a) : \mathsf{true}\right] - \Pr\left[M(a') : \mathsf{true}\right]\right|.$$

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).

In what follows we will describe differentially private mechanisms as imperative programs written in the language described in Section 2.2. We assume that these programs will sample values from proper probability distributions (i.e. distributions of unitary mass) only and will be assertion free. It is for this class of programs that the probability of termination is given by the mass of their output distributions—or equivalently, by the probability that they assign to true.

To conclude the section we highlight 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 4.4. *For all distributions μ_1 and μ_2 over a discrete set A,*

$$\Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) \leq \sum_{a \in A} \Delta^{\blacklozenge}_{\alpha}(\mu_1(a), \mu_2(a)).$$

4.1.3 Approximate Lifting of Relations to Distributions

Our α-pRHL is an approximate variant of the "exact" logic pRHL, which in turn elaborates on Benton [2004]'s relational Hoare logic. 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 states. The validity of such a judgment requires that terminating executions of programs c_1 and c_2 in initial states related by Ψ result in final states related by Φ. In pRHL and α-pRHL, judgments have the same shape: assertions are still binary relations over program states, but programs are probabilistic. Since in this setting a program execution results in a distribution over states rather than a single final state, 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 will be the cornerstone for defining validity of α-pRHL judgments.

Given $\alpha \in \mathbb{R}^{\geq 1}$ and $\delta \in [0,1]$, the (α,δ)-lifting of $R \subseteq A \times B$ is a relation between $\mathcal{D}(A)$ and $\mathcal{D}(B)$ formally defined as follows.

Definition 4.3 ((α,δ)-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 $\mathcal{L}^{\blacklozenge}_{\alpha,\delta}(R) \subseteq \mathcal{D}(A) \times \mathcal{D}(B)$ such that $\mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(R) \, \mu_2$ iff there exists a pair of distributions $\mu_L, \mu_R \in \mathcal{D}(A \times B)$ satisfying the following conditions:*

i) $\mathsf{range}\, R\, \mu_L \;\wedge\; \mathsf{range}\, R\, \mu_R$;
ii) $\pi_1(\mu_L) = \mu_1 \;\wedge\; \pi_2(\mu_R) = \mu_2$;
iii) $\Delta^{\blacklozenge}_{\alpha}(\mu_L, \mu_R) \leq \delta$.

The distributions μ_L and μ_R are called the left *and* right witnesses *for the lifting, respectively.*

In our development, we slightly depart from the definition of the (α,δ)-lifting used in [Barthe *et al.*, 2012], which is built on the basis of a single witness distribution. By adopting two witnesses, we reduce some technical burden when proving several properties of the lifting operation. These includes e.g. the forthcoming Theorem 4.8 or Lemma 4.9 (we refer the reader to [Barthe *et al.*, 2013b] for the proofs of these results using the lifting based on a single witness).

The notion of (α,δ)-lifting generalizes previous notions of lifting, such as that of Jonsson *et al.* [2001] discussed in Section 2.1, which is obtained by taking $(\alpha,\delta) = (1,0)$, and the δ-lifting [Desharnais *et al.*, 2008; Segala & Turrini, 2007], 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.

Lemma 4.5. *Let R be an equivalence relation over a discrete set A and let $\mu_1, \mu_{\in}\mathcal{D}(A)$. Then,*

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(R) \, \mu_2 \iff \Delta^{\blacklozenge}_{\alpha}(\mu_1/R, \mu_2/R) \leq \delta.$$

Jonsson *et al.* also show that for equivalence relations, their definition of lifting coincides with the more intuitive notion that requires related distributions to assign equal probabilities to all equivalence classes. This result can be recovered from Lemma 4.5 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 $\alpha > 1$, the (α, δ)-lifting is more permissive than the previously proposed notions of lifting.

Lemma 4.6. *For all $1 \leq \alpha \leq \alpha'$, $\delta \leq \delta'$, and relations $S \subseteq S'$,*

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(S) \, \mu_2 \implies \mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha',\delta'}(S') \, \mu_2.$$

We next present a fundamental property of (α, δ)-lifting, which is central to the applicability of α-pRHL 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 $b \in B$, $a \, R \, b$ implies $f(a) = g(b)$. In what follows we use $\equiv$ to denote the identity relation over arbitrary sets.

Theorem 4.7 (Fundamental Property of the (α, δ)-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 $f_2 : B \to [0, 1]$,*

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(R) \, \mu_2 \wedge f_1 =_R f_2 \implies \Delta^{\blacklozenge}_{\alpha}(\mu_1 \, f_1, \mu_2 \, f_2) \leq \delta.$$

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

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(\equiv) \, \mu_2 \implies \Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) \leq \delta.$$

Theorem 4.7 provides an interpretation of (α, δ)-lifting in terms of α-distance. Next we present a result that enables us to actually construct witnesses for such liftings.

The result is the converse of Theorem 4.7 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 α-pRHL rule for random assignments.

Theorem 4.8. *Let μ_1 and μ_2 be distributions over a discrete set A. Then*

$$\Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) \leq \delta \implies \mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(\equiv) \, \mu_2.$$

The proof is immediate by considering as a witnesses for the lifting the following distributions:

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

As a side remark, observe that the equivalence $\Delta^{\blacklozenge}_{\alpha}(\mu_1,\mu_2) \leq \delta \iff \mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(\equiv) \, \mu_2$ is immediate from Lemma 4.5. However, we prefer to keep separate statements and proofs for each direction (Theorems 4.7 and 4.8), because these correspond to theorems in our Coq formalization, while we only give a pencil-and-paper proof of Lemma 4.5 in Appendix 4.A.

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.

Lemma 4.9. *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*

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(R) \, \mu_2 \quad \textit{and} \quad \forall a, b \bullet a \, R \, b \implies M_1(a) \, \mathcal{L}^{\blacklozenge}_{\alpha',\delta'}(R') \, M_2(b)$$

we have

$$(\mathsf{bind}\ \mu_1\ M_1) \, \mathcal{L}^{\blacklozenge}_{\alpha\alpha',\delta+\delta'}(R') \, (\mathsf{bind}\ \mu_2\ M_2).$$

4.2 Approximate Relational Hoare Logic

This section introduces α-pRHL, an approximate probabilistic relational Hoare logic that is used to establish differential privacy guarantees of programs. We first define relational judgments and show that they generalize differential privacy. We then define a proof system for deriving valid judgments and finally we present an asymmetric variant of the logic.

4.2.1 Validity and Privacy

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

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

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*.

An α-pRHL judgment is *valid* iff, 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 4.4 (Validity in α-pRHL)**.** *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*

$$\forall m_1, m_2 \bullet m_1 \,\Psi\, m_2 \implies (\llbracket c_1 \rrbracket\, m_1)\, \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(\Phi)\, (\llbracket c_2 \rrbracket\, m_2).$$

The following lemma is a direct consequence of the fundamental property of the (α, δ)-lifting applied to Definition 4.4. It shows that statements about programs derived using α-pRHL imply bounds on the α-distance of their output distributions.

Lemma 4.10. *If* $\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$*, then for all memories* m_1, m_2 *and functions* $f_1, f_2 : \mathcal{M} \to [0,1]$*,*

$$m_1 \,\Psi\, m_2 \wedge f_1 =_\Phi f_2 \implies \Delta^{\blacklozenge}_\alpha((\llbracket c_1 \rrbracket\, m_1)(f_1), (\llbracket c_2 \rrbracket\, m_2)(f_2)) \leq \delta.$$

By specializing f_1 and f_2 to the characteristic function of two events A and B one obtain the following rule to derive claims about probability quantities (cf. rule [PrEq] from the exact logic pRHL on page 17):

$$\frac{m_1 \,\Psi\, m_2 \qquad \models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi \qquad \Phi \implies (A\langle 1\rangle \iff B\langle 2\rangle)}{\Delta^{\blacklozenge}_\alpha(\Pr[c_1(m_1) : A], \Pr[c_2(m_2) : B]) \leq \delta}\ [\mathsf{PrEq\text{-}}\Delta^{\blacklozenge}_\alpha]$$

This rule can be further specialized to a statement about the differential privacy of programs.

Corollary 4.11. *Let* d *be a metric on* $\mathcal{M}$ *and* Ψ *an assertion expressing that* $d(m_1, m_2) \leq 1$*. If* $\models c \sim_{e^\epsilon,\delta} c : \Psi \Rightarrow \equiv$*, then* c *satisfies* (ϵ, δ)*-differential privacy.*

Corollary 4.11 is the central result for deriving differential privacy guarantees in α-pRHL. Using Theorem 4.8, one can prove the converse to Corollary 4.11. These two results together imply that α-pRHL judgments completely characterize approximate differential privacy.

4.2.2 Logic

This section introduces a set of proof rules for reasoning about the validity of α-pRHL 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 α-pRHL rules in Figure 4.1; all rules generalize their counterparts in pRHL (see Figure 2.3), which can be recovered by setting $\alpha = 1$ and $\delta = 0$. (Any valid pRHL derivation admits an immediate translation into α-pRHL.)

The [skip], [assert] and [assn] rules are direct transpositions of the corresponding pRHL rules. Rule [rand] states that for any two distribution expressions d_1 and d_2 of type A, the random assignments $x_1 \xleftarrow{\$} d_1$ and $x_2 \xleftarrow{\$} d_2$ are (α, δ)-related w.r.t. pre-condition Ψ and

$$\frac{\forall m_1, m_2 \bullet\ m_1\ \Psi\ m_2 \implies (m_1\,\{[\![e_1]\!]_{\mathcal{E}}\ m_1/x_1\})\ \Phi\ (m_2\,\{[\![e_2]\!]_{\mathcal{E}}\ m_2/x_2\})}{\models x_1 \leftarrow e_1 \sim_{1,0} x_2 \leftarrow e_2 : \Psi \Rightarrow \Phi}\ [\mathsf{assn}]$$

$$\frac{\begin{array}{c}\forall m_1, m_2 \bullet\ m_1\ \Psi\ m_2 \implies \Delta^{\blacklozenge}_{\alpha}([\![d_1]\!]_{\mathcal{DE}}\ m_1, [\![d_2]\!]_{\mathcal{DE}}\ m_2) \le \delta \\ m_1\ \Psi'\ m_2 \stackrel{\text{def}}{=} \exists v_1, v_2 \bullet\ (m_1\,\{v_1/x_1\})\ \Psi\ (m_2\,\{v_2/x_2\})\end{array}}{\models x_1 \xleftarrow{\$} d_1 \sim_{\alpha,\delta} x_2 \xleftarrow{\$} d_2 : \Psi \Rightarrow x_1\langle 1\rangle = x_2\langle 2\rangle\ \wedge\ \Psi'}\ [\mathsf{rand}]$$

$$\frac{\Psi \implies b_1\langle 1\rangle = b_2\langle 2\rangle}{\models \mathsf{assert}\ b_1 \sim_{1,0} \mathsf{assert}\ b_2 : \Psi \Rightarrow \Psi \wedge b_1\langle 1\rangle}\ [\mathsf{assert}]$$

$$\frac{\begin{array}{c}\Psi \implies b_1\langle 1\rangle = b_2\langle 2\rangle \\ \models c_1 \sim_{\alpha,\delta} c_2 : \Psi \wedge b_1\langle 1\rangle \Rightarrow \Phi \quad \models c_1' \sim_{\alpha,\delta} c_2' : \Psi \wedge \neg b_1\langle 1\rangle \Rightarrow \Phi\end{array}}{\models \mathsf{if}\ b_1\ \mathsf{then}\ c_1\ \mathsf{else}\ c_1' \sim_{\alpha,\delta} \mathsf{if}\ b_2\ \mathsf{then}\ c_2\ \mathsf{else}\ c_2' : \Psi \Rightarrow \Phi}\ [\mathsf{cond}]$$

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

$$\models \mathsf{skip} \sim_{1,0} \mathsf{skip} : \Psi \Rightarrow \Psi\ [\mathsf{skip}] \qquad \frac{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi' \quad \models c_1' \sim_{\alpha',\delta'} c_2' : \Phi' \Rightarrow \Phi}{\models c_1; c_1' \sim_{\alpha\alpha',\delta+\delta'} c_2; c_2' : \Psi \Rightarrow \Phi}\ [\mathsf{seq}]$$

$$\frac{\begin{array}{c}\Psi \implies \Psi' \quad \Phi' \implies \Phi \quad \alpha' \le \alpha \quad \delta' \le \delta \\ \models c_1 \sim_{\alpha',\delta'} c_2 : \Psi' \Rightarrow \Phi'\end{array}}{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi}\ [\mathsf{weak}] \qquad \frac{\begin{array}{c}\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \wedge \Theta \Rightarrow \Phi \\ \models c_1 \sim_{\alpha,\delta} c_2 : \Psi \wedge \neg\Theta \Rightarrow \Phi\end{array}}{\models c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi}\ [\mathsf{case}]$$

Figure 4.1: Core proof rules of α-pRHL.

post-condition $x_1\langle 1\rangle = x_2\langle 2\rangle \wedge \Phi$, where $m_1\ \Phi\ m_2 \stackrel{\text{def}}{=} \exists v_1, v_2 \bullet\ (m_1\,\{v_1/x_1\})\ \Psi\ (m_2\,\{v_2/x_2\})$, provided the α-distance between the distributions $[\![d_1]\!]_{\mathcal{DE}}\ m_1$ and $[\![d_2]\!]_{\mathcal{DE}}\ m_2$ is smaller than δ for any m_1 and m_2 related by Ψ. (Observe that rule [rand] is stated in a strongest post-condition style, whereas rule [assgn] is stated using a weakest pre-condition style).

Rule [seq] encodes the sequential composition theorem of approximate differential privacy, further elaborated in Section 4.2.4.

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 $\Psi \wedge b_1\langle 1\rangle$ and $\Psi \wedge \neg b_1\langle 1\rangle$, 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 (standard) Hoare logic by

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

Figure 4.2: Generalized α-pRHL rule for loops.

allowing to increase the skew and slack.

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 requires exhibiting a 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 4.3.3, 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 4.3.4, require applying more sophisticated rules in which the skew and the slack may vary across iterations. For instance, the rule [gwhile] shown in Figure 4.2 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 $\alpha_1(j)$ if P does not hold after their execution, and with skew α_2 if 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 a $\ln(\alpha_2)$ privacy penalty; the remaining iterations preserve P and do not add to the privacy bound.

The proofs of soundness of α-pRHL rules in Coq rely on properties of the approximate lifting. For instance, the soundness of rules [weak] and [seq] follows directly from Lemmas 4.6 and 4.9, respectively. To illustrate the kind of reasoning such proofs involve, we sketch the soundness proof of [rand]. To establish the validity of judgment

$$x_1 \stackrel{\$}{\leftarrow} d_1 \sim_{\alpha,\delta} x_2 \stackrel{\$}{\leftarrow} d_2 : \Psi \Rightarrow \Phi,$$

where Φ is defined as in the statement of the rule in Figure 4.1, we have to show that for every pair of Ψ-related memories m_1 and m_2, distributions

$$\mathsf{bind}\,(\llbracket d_1 \rrbracket_{\mathcal{DE}}\, m_1)\,(\lambda v.\ \mathsf{unit}\,(m_1\,\{v/x_1\}) \quad \text{and} \quad \mathsf{bind}\,(\llbracket d_2 \rrbracket_{\mathcal{DE}}\, m_2)\,(\lambda v.\ \mathsf{unit}\,(m_2\,\{v/x_2\})$$

are related by $\mathcal{L}^{\blacklozenge}_{\alpha,\delta}(\Phi)$. We prove this by applying Lemma 4.9 with $(\alpha', \delta') = (1, 0)$, and R the identity relation. The hypotheses of the lemma simplify to

$$([\![d_1]\!]_{\mathcal{DE}}\ m_1)\ \mathcal{L}^{\blacklozenge}_{\alpha,\delta}(\equiv)\ ([\![d_2]\!]_{\mathcal{DE}}\ m_2) \quad \text{and} \quad \mathsf{unit}\,(m_1\,\{v/x_1\})\ \mathcal{L}^{\blacklozenge}_{1,0}(\Phi)\ \mathsf{unit}\,(m_2\,\{v/x_2\}).$$

The first follows from Theorem 4.8 and the premise of the rule, whereas the second follows from Proposition 4.18.

We conclude this section by highlighting that α-pRHL can also be used to reason about the notion of approximate observational equivalence from Chapter 3. To see why recall that statistical distance—the metric underlying approximate observational equivalence—belongs to the family of α-distances; it corresponds, concretely, to the instance $\alpha = 1$. α-pRHL can thus be specialized to reason about the statistical distance between programs by considering judgments of the form $c_1 \sim_{1,\delta} c_2 : \Psi \Rightarrow \Phi$. This kind of judgments succinctly model approximate observational equivalence, which is retrieved by taking as pre and post-conditions the equality over the sets of input and output variables, respectively. To be more precise, the assertion about observational equivalence $\models c_1 \simeq^{I}_{O} c_2 \preceq \delta$ can be represented within α-pRHL as $\models c_1 \sim_{1,\delta} c_2 : =_I \Rightarrow =_O$. (Observe that the application of Lemma 4.10 to judgment $\models c_1 \sim_{1,\delta} c_2 : =_I \Rightarrow =_O$ reads as the definition of $\models c_1 \simeq^{I}_{O} c_2 \preceq \delta$). Moreover, if we drop rule [trans], all rules of the equational theory to reason about approximate observational equivalence (see Figure 3.2) admit a transposition in the proof system of α-pRHL. Therefore, any derivation about approximate observational equivalence can be transposed to a derivation in the proof system of α-pRHL.

4.2.3 An Asymmetric Variant of α-pRHL

The judgments of α-pRHL 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 4.10 entails inequalities

$$([\![c_1]\!]\, m_1)(f_1) \leq \alpha\, ([\![c_2]\!]\, m_2)(f_2) + \delta \qquad \text{and} \qquad ([\![c_2]\!]\, m_2)(f_2) \leq \alpha\, ([\![c_1]\!]\, m_1)(f_1) + \delta$$

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 α-pRHL$^{\blacklozenge}$, an asymmetric variant of α-pRHL that allows deriving only one of the above inequalities, and thus allows an independent and finer-grained choice of the skew α and the slack δ. α-pRHL$^{\blacklozenge}$ 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 4.1. All rules from α-pRHL remain valid in α-pRHL$^{\blacklozenge}$.

We next give formal definitions of the asymmetric counterparts of the notions studied in Sections 4.1.1 and 4.1.3 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 the asymmetric variant of the α-distance

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

where $\Delta_\alpha^{\blacklozenge}(a,b) \stackrel{\text{def}}{=} \max\{a-\alpha\,b,0\}$. 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 $\mathcal{L}^{\blacklozenge}_{\alpha,\delta}(R)$ such that $\mu_1\,\mathcal{L}^{\blacklozenge}_{\alpha,\delta}(R)\,\mu_2$ iff there exists a pair of distributions $\mu_L,\mu_R\in\mathcal{D}(A\times B)$ satisfying

i) $\mathsf{range}\,R\,\mu_L \;\wedge\; \mathsf{range}\,R\,\mu_R$;
ii) $\pi_1(\mu_L)=\mu_1 \;\wedge\; \pi_2(\mu_R)=\mu_2$;
iii) $\Delta_\alpha^{\blacklozenge}(\mu_L,\mu_R)\leq\delta$.

The distance $\Delta_\alpha^{\blacklozenge}$ enjoys all properties of Lemma 4.2, except symmetry; the generalized triangle inequality can be strengthened to $\Delta^{\blacklozenge}_{\alpha\alpha'}(\mu_1,\mu_3) \leq \Delta^{\blacklozenge}_\alpha(\mu_1,\mu_2)+\alpha\,\Delta^{\blacklozenge}_{\alpha'}(\mu_2,\mu_3)$. Lemma 4.16 can be reformulated as $\Delta^{\blacklozenge}_\alpha(\mu_1,\mu_2)=\mu_1(A_{\mu_1\geq\alpha\mu_2})-\alpha\,\mu_2(A_{\mu_1\geq\alpha\mu_2})$, where $\mu_1,\mu_2:\mathcal{D}(A)$. This relates both variants of the α-distance by

$$\Delta_\alpha^{\blacklozenge}(\mu_1,\mu_2)=\max\{\Delta_\alpha^{\blacklozenge}(\mu_1,\mu_2)\,,\Delta_\alpha^{\blacklozenge}(\mu_2,\mu_1)\}. \tag{4.2}$$

Finally, one can upper-bound $\Delta_\alpha^{\blacklozenge}(\mu_1,\mu_2)$ by $\sum_{a\in A}\Delta_\alpha^{\blacklozenge}(\mu_1(a),\mu_2(a))$ as Lemma 4.4 does for standard α-distance.

The new notion of lifting satisfies both the monotonicity condition of Lemma 4.6 and an analogue of Theorem 4.8. 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 4.7 is reformulated as follows:

$$\mu_1\,\mathcal{L}^{\blacklozenge}_{\alpha,\delta}(R)\,\mu_2 \;\wedge\; f_1\leq_R f_2 \implies \Delta^{\blacklozenge}_\alpha(\mu_1(f_1),\mu_2(f_2))\leq\delta.$$

We next define validity in α-pRHL$^{\blacklozenge}$ and show how the asymmetric logic can be used to relate the distributions generated by probabilistic programs.

Definition 4.5 (Validity in α-aRHL$^{\blacklozenge}$)**.** *We say that a judgment $c_1\sim_{\alpha,\delta}c_2:\Psi\Rightarrow\Phi$ is valid in α-pRHL$^{\blacklozenge}$, written $\models^{\blacklozenge} c_1\sim_{\alpha,\delta}c_2:\Psi\Rightarrow\Phi$, iff*

$$\forall m_1,m_2\bullet\; m_1\,\Psi\,m_2 \implies (\llbracket c_1\rrbracket\,m_1)\,\mathcal{L}^{\blacklozenge}_{\alpha,\delta}(\Phi)\,(\llbracket c_2\rrbracket\,m_2).$$

Lemma 4.12. *If $\models^{\blacklozenge} c_1\sim_{\alpha,\delta}c_2:\Psi\Rightarrow\Phi$, then for all memories m_1,m_2 and $[0,1]$-valued functions $f_1,f_2:\mathcal{M}\to[0,1]$,*

$$m_1\,\Psi\,m_2 \;\wedge\; f_1\leq_\Phi f_2 \implies \Delta^{\blacklozenge}_\alpha((\llbracket c_1\rrbracket\,m_1)(f_1),(\llbracket c_2\rrbracket\,m_2)(f_2))\leq\delta.$$

It is not hard to see that Corollary 4.11 and its converse remain valid if the validity of judgment $c \sim_{e^\epsilon,\delta} c : \Psi \Rightarrow \equiv$ is taken in α-pRHL$^\blacklozenge$ instead of α-pRHL. (This is true for any symmetric precondition Ψ). Therefore, approximate differential privacy can also be cast in terms of α-pRHL$^\blacklozenge$. We immediately obtain a proof system for reasoning about the validity of α-pRHL$^\blacklozenge$ judgments. All α-pRHL rules in Figures 4.1 and 4.2 can be transposed to α-pRHL$^\blacklozenge$. For consistency, we keep the names of the original rules and decorate them with symbol $^\blacklozenge$. E.g., the rule for random assignments reads

$$\frac{\begin{array}{c}\forall m_1, m_2 \bullet m_1 \,\Psi\, m_2 \implies \Delta^\blacklozenge_\alpha([\![d_1]\!]_{\mathcal{DE}}\, m_1, [\![d_2]\!]_{\mathcal{DE}}\, m_2) \leq \delta \\ m_1 \,\Psi'\, m_2 \stackrel{\text{def}}{=} \exists v_1, v_2 \bullet (m_1\,\{v_1/x_1\})\,\Psi\,(m_2\,\{v_2/x_2\})\end{array}}{\models^\blacklozenge x_1 \stackrel{\$}{\leftarrow} d_1 \sim_{\alpha,\delta} x_2 \stackrel{\$}{\leftarrow} d_2 : \Psi \Rightarrow x_1\langle 1\rangle = x_2\langle 2\rangle \wedge \Psi'}\ [\mathsf{rand}^\blacklozenge]$$

In Section 4.3.4 we demonstrate the benefits of α-pRHL$^\blacklozenge$ over α-pRHL. Concretely, we show how α-pRHL$^\blacklozenge$ 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 α-pRHL.

4.2.4 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 α-pRHL.

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 α-pRHL 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 α-pRHL. 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.

4.3 Case Studies

We illustrate the versatility of our approach for reasoning about differential privacy we present case studies by proving from first principles the correctness of the three standard mechanisms, namely, the Laplacian, Gaussian and Exponential mechanisms. Then we prove differential privacy for an algorithm solving the k-Median problem, several streaming algorithms, and an approximation algorithm for the Minimum Vertex Cover problem.

4.3.1 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.* [2006a,b] 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

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

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

$$\mathbf{S}_f \stackrel{\text{def}}{=} \sup_{\substack{a,a' \in A \\ d(a,a') \leq 1}} |f(a) - f(a')|.$$

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 constraint later.

One limitation of the Laplacian and Gaussian mechanisms is that they are confined to numerical algorithms. The Exponential mechanism [McSherry & Talwar, 2007] is a general mechanism for building differentially private algorithms with arbitrary 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 discrete Exponential mechanism $\mathcal{E}^{\epsilon}_{s,\mu}$ maps every element in A to a distribution in B whose probability mass at b is:

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

The definition implicitly assumes that the sum in the denominator is bounded for all $a \in A$. McSherry & 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 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:

$$f^\sharp(a)(b) \stackrel{\text{def}}{=} \frac{f(a)(b)}{\sum_{b' \in B} f(a)(b')}.$$

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

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

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

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

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 4.13. *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 $\alpha \in \mathbb{R}^{\geq 1}$ 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 $\alpha' \in \mathbb{R}^{\geq 1}$,*

$$\Delta^{\blacklozenge}_{\alpha\alpha'}\Big(f^{\sharp}(a_1), f^{\sharp}(a_2)\Big) \leq \max\Big\{f^{\sharp}(a_1)(S_1), f^{\sharp}(a_2)(S_2)\Big\},$$

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

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

Corollary 4.14. *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 $\alpha \in \mathbb{R}^{\geq 1}$ 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,*

$$\Delta^{\blacklozenge}_{\alpha^2}\Big(f^{\sharp}(a_1), f^{\sharp}(a_2)\Big) = 0.$$

If moreover $\sum_{b\in B} f(a_1)(b) = \sum_{b\in B} f(a_2)(b)$, then

$$\Delta^{\blacklozenge}_{\alpha}\Big(f^{\sharp}(a_1), f^{\sharp}(a_2)\Big) = 0.$$

The privacy guarantees for the Laplacian, Gaussian and Exponential mechanisms are stated as rules [lap], [norm] and [exp] in Figure 4.3. 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

$$\frac{m_1 \,\Psi\, m_2 \implies \left|[\![r]\!]_{\mathcal{E}}\, m_1 - [\![r]\!]_{\mathcal{E}}\, m_2\right| \leq k \qquad e^{\epsilon} \leq \alpha}{\models x \xleftarrow{\$} \mathcal{L}(r, k/\epsilon) \sim_{\alpha,0} y \xleftarrow{\$} \mathcal{L}(r, k/\epsilon) : \Psi \Rightarrow x\langle 1\rangle = y\langle 2\rangle} \text{ [lap]}$$

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

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

Figure 4.3: Rules for the Laplacian, Gaussian and Exponential mechanisms.

bound. For the sake of generality, we present rule [norm] in a generic way and assume no particular bound for $\mathcal{B}\left(\sigma, x\right)$.

As a further illustration of the expressive power of our technique, 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

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

4.3.2 Statistics over Streams

In this section we present analyses of algorithms for computing private and continual statistics in data streams [Chan *et al.*, 2010]. 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 set $D \subseteq \mathbb{Z}$ such that $|x - y| \leq b$ for all $x, y \in D$. This setting is slightly more general than the one considered by 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.

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 $0 \leq j < n$ 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, the sum $\tilde{c}[j]$ yields poor accuracy because noise is added j times during its computation.

One solution proposed by 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 $j = q\,r + k$, with $k < q$, we compute

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

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 $j = q\,r+k$ times.

We can 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 4.4, the code for computing $\overline{c}$ is given as the function PartialSum' in Figure 4.5, and the code for computing $\hat{c}$ is given as the function SmartSum in Figure 4.6. 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

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

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 (denoted as relation $\doteq$), 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 α-pRHL) 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 $s \leftarrow 0; i \leftarrow 0;$
2 while $i <$ length(a) do
3 $\quad s \leftarrow s + a[i];$
4 $\quad i \leftarrow i + 1;$
5 end;
6 $s \leftarrow \mathcal{L}(s, b/\epsilon)$

Figure 4.4: A simple ϵ-differentially private algorithm for sums over streams.

where c_{1-5} corresponds to the code in lines 1-5 in Figure 4.4, 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 4.3) to $c_6 = s \xleftarrow{\$} \mathcal{L}(s, b/\epsilon)$ (the instruction in line 6) and derive

$$\models c_6 \sim_{e^\epsilon,0} c_6 : |s\langle 1\rangle - s\langle 2\rangle| \leq b \Rightarrow s\langle 1\rangle = s\langle 2\rangle.$$

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.

$$\models \text{PartialSum}(a) \sim_{e^\epsilon,0} \text{PartialSum}(a) : \Psi \Rightarrow s\langle 1\rangle = s\langle 2\rangle.$$

PartialSum' Our implementation of PartialSum' in Figure 4.5 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

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

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 α-pRHL) 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 α-pRHL rules, but is also an immediate consequence of the preservation of α-distance by probabilistic computations (see Lemma 4.2).

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

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

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

```
function 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
```

Figure 4.5: An ϵ-differentially private algorithm for partial sums over streams.

```
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
```

Figure 4.6: 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 4.6 makes use of PartialSum and PartialSum' as building blocks, which enables us to reuse the above proofs. In addition, we use 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 $0 \leq j < q$. We obtain

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

where X stands for the set of variables $\{s, x, c, i, q\}$. We combine this result with the judgments derived for PartialSum and PartialSum' using the rule for sequential composition, obtaining

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

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. 4.2 with $\alpha_1(i) = 1$ and $\alpha_2 = e^{2\epsilon}$. 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.

4.3.3 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 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 that may not be symmetric.) 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

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

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 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.* [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 4.7. 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

function kMedian(C, ϵ, S_0)
1 $i \leftarrow 0; S[0] \leftarrow S_0;$
2 **while** $i < T$ **do**
3 $\quad (x, y) \xleftarrow{\$} \mathsf{PickSwap}_{\epsilon,C,S[i]}(S[i] \times (V \setminus S[i]));$
4 $\quad S[i+1] \leftarrow S[i] \setminus \{x\} \cup \{y\};$
5 $\quad i \leftarrow i + 1$
6 **end;**
7 $j \xleftarrow{\$} \mathsf{PickSolution}_{\epsilon,C,T}(S)$

Figure 4.7: A $2\epsilon\Delta(T+1)$-differentially private algorithm for computing the k-Median.

uses (a variant of) the Exponential mechanism. Specifically, the pair of points (x, y) is drawn from the parametrized distribution PickSwap. Given $C, F \subseteq V$, $R \subseteq V \times V$ and $\epsilon > 0$, distribution $\mathsf{PickSwap}_{\epsilon,C,F}(R)$ assigns to each pair (x, y) in R a probability proportional to $e^{-\epsilon\ cost_C(F \setminus \{x\} \cup \{y\})}$. Technically, this mechanism is defined as an instance of the construction $(\cdot)^\sharp$ introduced in Section 4.3.1:

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

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

$$g_{\epsilon,R}\,(C, F)\,(x, y) \stackrel{\text{def}}{=} e^{-\epsilon\ cost_C(F \setminus \{x\} \cup \{y\})}.$$

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 PickSolution in line 7, which is defined in the same spirit as PickSwap by equation:

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

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

$$h_{\epsilon,T,S}(C, j) \stackrel{\text{def}}{=} e^{-\epsilon\ cost_C(S[j])}.$$

The original proof [Gupta *et al.*, 2010] shows that the algorithm in Figure 4.7 is $2\epsilon\Delta(T+1)$-differentially private, where $\Delta \stackrel{\text{def}}{=} \max_{v_1, v_2 \in V} d(v_1, v_2)$ 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 α-pRHL. The privacy statement is formalized by the judgment

$$\models \mathrm{kMedian}(C, \epsilon, S_0) \sim_{e^{2\epsilon\Delta(T+1)},0} \mathrm{kMedian}(C, \epsilon, S_0) : \Psi \Rightarrow S[j]\langle 1\rangle = S[j]\langle 2\rangle, \tag{4.3}$$

where $\Psi = S_0\langle 1\rangle = S_0\langle 2\rangle \wedge C\langle 1\rangle \doteq C\langle 2\rangle$. We let $c = \mathrm{kMedian}(C, \epsilon, S_0)$ and use the same convention as in Section 4.3.2 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 (4.3) from judgments

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

and

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

where $I = S\langle 1\rangle = S\langle 2\rangle \wedge C\langle 1\rangle \doteq C\langle 2\rangle \wedge i\langle 1\rangle = i\langle 2\rangle$.

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

$$\models c_{3-5} \sim_{e^{2\epsilon\Delta},0} c_{3-5} : I \wedge (i < T)\langle 1\rangle \wedge i\langle 1\rangle = k \Rightarrow I \wedge i\langle 1\rangle > k$$

we use rule [assn] to deal with lines 4 and 5 and rule [rand] to deal with the random assignment in line 3. By setting $d = \mathsf{PickSwap}_{\epsilon,C,S[i]}(S[i] \times (V \setminus S[i]))$, the premise

$$\forall m_1, m_2 \bullet\ m_1\, I\, m_2 \implies \Delta^{\blacklozenge}_{e^{2\epsilon\Delta}}([\![d]\!]_{\mathcal{DE}}\ m_1, [\![d]\!]_{\mathcal{DE}}\ m_2) \leq 0$$

of rule [rand] can be discharged by Corollary 4.14, which requires showing that for all C_1, C_2, F, x and y satisfying $C_1 \doteq C_2\ \wedge\ x \in F\ \wedge\ y \notin F$,

$$e^{-\epsilon\ cost_{C_1}(F\setminus\{x\}\cup\{y\})} \leq e^{\epsilon\Delta}\, e^{-\epsilon\ cost_{C_2}(F\setminus\{x\}\cup\{y\})}.$$

This inequality is a direct consequence of the sensitivity property of 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, applying rule [rand]. Let $d' = \mathsf{PickSolution}_{\epsilon,C,T}(S)$; the premise

$$\forall m_1, m_2 \bullet\ m_1\, I\, m_2 \implies \Delta^{\blacklozenge}_{e^{2\epsilon\Delta}}\left([\![d']\!]_{\mathcal{DE}}\ m_1, [\![d']\!]_{\mathcal{DE}}\ m_2\right) \leq 0$$

of rule [rand] boils down to proving that for all C_1, C_2, S and j satisfying $C_1 \doteq C_2$ and $0 \leq j < T$,

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

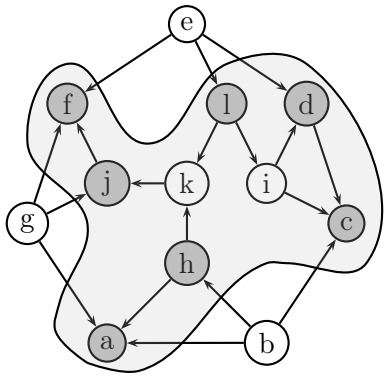
which follows from the sensitivity of function *cost*.

4.3.4 Minimum Vertex Cover

In this section we use α-pRHL$^{\blacklozenge}$ to provide a formal proof of the differential privacy for an approximation algorithm solving the Minimum (Unweighted) Vertex Cover (MVC) problem [Gupta *et al.*, 2010].

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 (presumably in a privacy-preserving distributed manner) by taking for each edge the vertex it points to (see Figure 4.8). 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 4.9 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. This idea can be implemented by the algorithm shown in Figure 4.9 if we require the instruction $v \xleftarrow{\$} \mathsf{PickVertex}_{E,\epsilon,n,i}(V)$ to choose a vertex v from V with probability



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

Figure 4.8: 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.

function VertexCover(V, E, ϵ)
1 $n \leftarrow |V|;\ \pi \leftarrow \mathsf{nil};\ i \leftarrow 0;$
2 while $i < n$ do
3 $\quad v \xleftarrow{\$} \mathsf{PickVertex}_{E,\epsilon,n,i}(V);$
4 $\quad \pi \leftarrow v :: \pi;$
5 $\quad V \leftarrow V \setminus \{v\};\ E \leftarrow E \setminus (\{v\} \times V);$
6 $\quad i \leftarrow i + 1$
7 end

Figure 4.9: A differentially private approximation algorithm for the Minimum Unweighted Vertex Cover problem

proportional to $d_E(v) + w_i$, where $d_E(v)$ denotes the degree of v in E and

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

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

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

We first present an informal analysis on the privacy of the algorithm and then discuss how to justify this reasoning with the tools of Section 4.2. Let $G_1 = (V, E)$ and $G_2 = (V, E \cup \{(t, u)\})$ be a pair of graphs with the same set of vertices but differing in exactly one edge. We can conclude that the above algorithm is ϵ-differentially private by showing

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 e^ϵ from the probability of obtaining π when the input is G_2 and vice versa. To do so we show privacy bounds for each iteration of the loop and apply composition results to conclude. 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 G_1 and G_2 respectively, and its reciprocal. In the analysis below, we use $E\langle 1\rangle$ (resp. $E\langle 2\rangle$) to denote the value of E in the i-th iteration when the algorithm is run with G_1 (resp. G_2) as input; idem for v. 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 $1 + x \leq e^x$ to derive upper bounds in each case:

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

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

(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;

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

(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

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

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

and the ϵ-differential privacy of the algorithm follows.

We now transform the above security analysis into a formal reasoning using the logic of Section 4.2.3. Each of the two above inequalities can be captured by the α-pRHL$^{\blacklozenge}$ judgments

$$\models^{\blacklozenge} \text{VertexCover}(V,E,\epsilon) \sim_{\epsilon\epsilon,0} \text{VertexCover}(V,E,\epsilon) : \Psi_1 \Rightarrow \Phi \tag{4.4}$$

and

$$\models^{\blacklozenge} \text{VertexCover}(V,E,\epsilon) \sim_{e^{\epsilon},0} \text{VertexCover}(V,E,\epsilon) : \Psi_2 \Rightarrow \Phi \tag{4.5}$$

respectively, 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 judgment (4.4). We prove the validity of this judgment using an asymmetric variant of the generalized rule for while loops given in Figure 4.2. This rule, which we call [gwhile$^{\blacklozenge}$], has the same shape as [gwhile], but judgments in the premises and conclusion are interpreted in α-pRHL$^{\blacklozenge}$. We apply the rule with parameters

$$\alpha_1(i) = e^{\frac{2}{(n-i)w_i}} \qquad \alpha_2 = 1,$$

the following invariant

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

(which is established by the initialization code before the loop,) and the stable property

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

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

$$\models^{\blacklozenge} c; \mathsf{assert}\ P \sim_{\alpha,0} c; \mathsf{assert}\ P : \Psi' \Rightarrow \Phi'$$

and

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

where c is the body of the loop. For each of them, we first hoist the assertion immediately after the random assignment that chooses the vertex v in c. As a result, the expression

in the assertions 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 $[\mathsf{weak}^{\blacklozenge}]$ rule, to judgments of the form

$$\models^{\blacklozenge} c' \sim_{\alpha,0} c' : \Theta \wedge i\langle 1\rangle = j \wedge t, u \notin \pi \Rightarrow v\langle 1\rangle = v\langle 2\rangle \wedge \Theta \wedge i\langle 1\rangle = j.$$

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

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

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

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

To establish the validity of both judgments, we cast the code for c' as a random assignment where v is sampled from the interpretation of $\mathsf{PickVertex}_{E,\epsilon,n,i}(V)$ restricted to v satisfying the condition on the assertion. In the first case, the restriction amounts to $v \neq u, t$ whereas in the second case it amounts to $v = t \vee v = u$. For each one of these cases, we apply the rule for random assignments $[\mathsf{rand}^{\blacklozenge}]$ and are thus left to prove that distance $\Delta_{\alpha}^{\blacklozenge}$ between the corresponding distributions is null. In view of the variant of Lemma 4.4 for $\Delta_{\alpha}^{\blacklozenge}$ this in turn amounts to verifying for each element x in the support of the distribution that the ratio between the probability of v being equal to x in the left-hand side program and the right-hand side program 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 4.14.

To prove the remaining judgment (4.5) we follow a similar reasoning. In this case, we apply rule $[\mathsf{gwhile}^{\blacklozenge}]$ with parameters

$$\alpha_1(i) = 1 \qquad \alpha_2 = e^{\epsilon/4},$$

the following 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 α-pRHL$^{\blacklozenge}$ is fundamental to prove the privacy bound ϵ from [Gupta *et al.*, 2010], as opposed to [Barthe *et al.*, 2012], where the

use of α-pRHL yields a looser bound of $5\epsilon/4$. This is because the proof in α-pRHL$^{\blacklozenge}$ allows to prove independently that e^{ϵ} is a bound for the ratios

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

while a proof in α-pRHL requires to prove this simultaneously. As a consequence, the proof in α-pRHL$^{\blacklozenge}$ consists of two independent applications of the asymmetric rule [gwhile$^{\blacklozenge}$]. 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 $(\alpha_1(i),\alpha_2) = \left(e^{2/((n-i)w_i)},1\right)$ and $(\alpha_1(i),\alpha_2) = (1,e^{\epsilon/4})$. In contrast, when using the symmetric logic α-pRHL, one needs to choose a single pair of parameters to bound both ratios simultaneously, namely $(\alpha_1(i),\alpha_2) = \left(e^{2/((n-i)w_i)},e^{\epsilon/4}\right)$. This translates into a looser privacy bound.

4.A Appendix

In this section we present proof sketches for all the results of the present chapter. Some results can be obtained as specializations to the α-distance of results in Chapter 5, modulo the symmetric/asymmetric setting considered in each case (the results of Chapter 5 apply to arbitrary f-divergences, which, as we shall see, include α-distance). In these cases, we directly point out to the corresponding result of Chapter 5.

All results presented here and in the body of the chapter have been formally verified using the Coq proof assistant, with the only exception of Lemma 4.5, which is not central to our development.

We present first some auxiliary lemmas and then turn to the proofs.

4.A.1 Auxiliary Lemmas

Proposition 4.15. *Let μ_1 and μ_2 be two distributions over a discrete set A. Then for any $f : A \to [0,1]$,*

$$\Delta_{\alpha}^{\blacklozenge}(\mu_1(f),\mu_2(f)) \leq \max\left\{\mu_1(A_0) - \alpha\,\mu_2(A_0), \mu_2(A_1) - \alpha\,\mu_1(A_1)\right\},$$

where $A_0 = A_{\mu_1 \geq \alpha\mu_2}$ and $A_1 = A_{\mu_2 \geq \alpha\mu_1}$.

Proof. Observe that

$$\mu_1(f) - \alpha\,\mu_2(f) = \sum_{a\in A}\mu_1(a)f(a) - \alpha\sum_{a\in A}\mu_2(a)f(a)$$

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

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. ■

Lemma 4.16. *Let μ_1 and μ_2 be two distributions over a discrete set A. Then,*

$$\Delta^{\blacklozenge}_{\alpha}(\mu_1,\mu_2) = \max\big\{\mu_1(A_0) - \alpha\,\mu_2(A_0), \mu_2(A_1) - \alpha\,\mu_1(A_1)\big\},$$

where $A_0 \stackrel{\text{def}}{=} A_{\mu_1\geq\alpha\mu_2}$ and $A_1 \stackrel{\text{def}}{=} A_{\mu_2\geq\alpha\mu_1}$.

Proof. The inequality $\Delta^{\blacklozenge}_{\alpha}(\mu_1,\mu_2) \geq \max\big\{\mu_1(A_0) - \alpha\mu_2(A_0), \mu_2(A_1) - \alpha\mu_1(A_1)\big\}$ follows trivially from the definition of α-distance between distributions. The converse inequality is derived from Proposition 4.15 as follows:

$$\begin{aligned} \Delta^{\blacklozenge}_{\alpha}(\mu_1,\mu_2) &= \sup_{f:A\to\{0,1\}} \Delta^{\blacklozenge}_{\alpha}(\mu_1(f),\mu_2(f)) \leq \sup_{f:A\to[0,1]} \Delta^{\blacklozenge}_{\alpha}(\mu_1(f),\mu_2(f)) \\ &\leq \max\big\{\mu_1(A_0) - \alpha\,\mu_2(A_0), \mu_2(A_1) - \alpha\,\mu_1(A_1)\big\}. \end{aligned}$$ ■

Lemma 4.17. *Let A and B be two discrete sets. Then, for any $\mu_1,\mu_2 \in \mathcal{D}(A)$ and $M_1, M_2 : A \to \mathcal{D}(B)$,*

$$\Delta^{\blacklozenge}_{\alpha\alpha'}(\mathsf{bind}\;\mu_1\,M_1, \mathsf{bind}\;\mu_2\,M_2) \leq \Delta^{\blacklozenge}_{\alpha}(\mu_1,\mu_2) + \sup_a\, \Delta^{\blacklozenge}_{\alpha'}(M_1(a), M_2(a))\,.$$

Proof. As a first step, observe that

$$\Delta^{\blacklozenge}_{\alpha\alpha'}(\mathsf{bind}\;\mu_1\,M_1, \mathsf{bind}\;\mu_2\,M_2) \leq \Delta^{\blacklozenge}_{\alpha\alpha'}(\theta_1,\theta_2)\,,$$

where distributions $\theta_1,\theta_2 \in \mathcal{D}(A\times B)$ are defined as $\theta_1(a,b) = \mu_1(a)\,M_1(a)(b)$ and $\theta_2(a,b) = \mu_2(a)\,M_2(a)(b)$. This claim follows from Lemma 4.2.v), since distributions $\mathsf{bind}\;\mu_1\,M_1$ and $\mathsf{bind}\mu_2\,M_2$ can be computed from θ_1 and θ_2 as $\mathsf{bind}\mu_1\,M_1 = \pi_2(\theta_1)$ and $\mathsf{bind}\mu_2\,M_2 = \pi_2(\theta_2)$.

We now apply Lemma 4.16 to bound $\Delta^{\blacklozenge}_{\alpha\alpha'}(\theta_1,\theta_2)$. 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 $X_0 = \{(a,b) \mid \theta_1(a,b) \geq \alpha\alpha'\,\theta_2(a,b)\}$, $X_1 = \{(a,b) \mid \theta_2(a,b) \geq \alpha\alpha'\,\theta_1(a,b)\}$, $\delta = \Delta^{\blacklozenge}_{\alpha}(\mu_1,\mu_2)$ and $\delta' = \sup_a \Delta^{\blacklozenge}_{\alpha'}(M_1(a),M_2(a))$. We now prove the first inequality. In doing so, we use the following notation. Given $R \subseteq A\times B$, we let $\pi_1(R)$ and $R(a)$ denote sets $\{a\in A \mid \exists b.\ (a,b)\in R\}$ and $\{b\in B \mid (a,b)\in R\}$, respectively.

$\theta_1(X_0) - \alpha\alpha'\,\theta_2(X_0)$

$$
\begin{aligned}
&= \sum_{(a,b)\in X_0} \mu_1(a)\, M_1(a)(b) - \alpha\alpha'\, \mu_2(a)\, M_2(a)(b) \\
&\overset{(1)}{=} \sum_{a\in\pi_1(X_0)} \sum_{b\in X_0(a)} \mu_1(a)\, M_1(a)(b) - \alpha\alpha'\, \mu_2(a)\, M_2(a)(b) \\
&= \sum_{a\in\pi_1(X_0)} \mu_1(a)\, \nu_1(a) - \alpha\alpha'\, \mu_2(a)\, \nu_2(a) \quad \text{where } \nu_i(a) \overset{\text{def}}{=} M_i(a)(X_0(a)) \\
&\overset{(2)}{\leq} \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)>1}} \mu_1(a) - \alpha\, \mu_2(a) \;+ \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1}} \mu_1(a)(\alpha'\, \nu_2(a) + \delta') - \alpha\alpha'\, \mu_2(a)\, \nu_2(a) \\
&= \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)>1}} \mu_1(a) - \alpha\, \mu_2(a) + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1}} \mu_1(a)\, \delta' + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1}} \alpha'\, \nu_2(a)\big(\mu_1(a) - \alpha\, \mu_2(a)\big) \\
&\leq \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)>1}} \mu_1(a) - \alpha\, \mu_2(a) + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1}} \mu_1(a)\, \delta' + \sum_{\substack{a\in\pi_1(X_0)\\ \alpha'\nu_2(a)\leq 1\\ \mu_1(a)\geq\alpha\mu_2(a)}} \mu_1(a) - \alpha\, \mu_2(a)
\end{aligned}
$$

Here, (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 the definition of δ' to bound $\nu_1(a)$ by $\alpha'\, \nu_2(a) + \delta'$.

From the above reasoning, by letting $Y_1 = \{a \in \pi_1(X_0) \mid \alpha'\, \nu_2(a) \leq 1 \implies \mu_1(a) \geq \alpha\, \mu_2(a)\}$ and $Y_2 = \{a \in \pi_1(X_0) \mid \alpha'\, \nu_2(a) \leq 1\}$ we obtain

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

We use a similar argument to prove that $\theta_2(X_1) - \alpha\alpha'\, \theta_1(X_1) \leq \delta + \delta'$ and conclude. ■

Proposition 4.18. *For any relation $R \subseteq A \times B$,*

$$a\, R\, b \implies (\mathsf{unit}\, a)\, \mathcal{L}^{\blacklozenge}_{1,0}(R)\, (\mathsf{unit}\, b).$$

Proof. See Proposition 5.20. ■

4.A.2 Proofs

Proof of Lemma 4.1. The inequality $\Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) \leq \sup_{f:A\to[0,1]} \Delta^{\blacklozenge}_{\alpha}(\mu_1(f), \mu_2(f))$ is immediate from the definition of α-distance between distributions. To prove the converse inequality we apply transitivity with $\max\{\mu_1(A_0) - \alpha\, \mu_2(A_0), \mu_2(A_1) - \alpha\, \mu_1(A_1)\}$, where A_0 and A_1 are defined as in Proposition 4.15. The inequality $\Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) \geq \max\{\mu_1(A_0) - \alpha\, \mu_2(A_0), \mu_2(A_1) - \alpha\, \mu_1(A_1)\}$ follows from the definition of α-distance, while inequality $\max\{\mu_1(A_0) - \alpha\, \mu_2(A_0), \mu_2(A_1) - \alpha\, \mu_1(A_1)\} \geq \sup_{f:A\to[0,1]} \Delta^{\blacklozenge}_{\alpha}(\mu_1(f), \mu_2\, f)$ holds by Proposition 4.15. ■

Proof of Lemma 4.2. All properties follow immediately from the definition of α-distance or some straightforward computations. ■

Proof of Lemma 4.3. A direct calculation shows that for $E \subseteq B_\perp$ we have

$$\begin{aligned}
&\Delta^{\blacklozenge}_{e^\epsilon}\big(M_\perp(a)(E), M_\perp(a')(E)\big) \\
&\quad \le \Delta^{\blacklozenge}_{e^\epsilon}\big(M_\perp(a)(E \setminus \{\perp\}), M_\perp(a')(E \setminus \{\perp\})\big) + \Delta^{\blacklozenge}_{e^\epsilon}\big(M_\perp(a)(\perp), M_\perp(a')(\perp)\big) \\
&\quad = \Delta^{\blacklozenge}_{e^\epsilon}\big(M(a)(E \setminus \{\perp\}), M(a')(E \setminus \{\perp\})\big) + \Delta^{\blacklozenge}_{e^\epsilon}\big(1 - M(a)(B), 1 - M(a')(B)\big) \\
&\quad \le \delta + \delta'.
\end{aligned}$$ ■

Proof of Lemma 4.4. Let A_0 and A_1 be defined as in Lemma 4.15. Then,

$$\begin{aligned}
\Delta^{\blacklozenge}_\alpha(\mu_1, \mu_2) &= \max\left\{ \sum_{a \in A_0} \mu_1(a) - \alpha\,\mu_2(a), \sum_{a \in A_1} \mu_2(a) - \alpha\,\mu_1(a) \right\} \\
&= \max\left\{ \sum_{a \in A} \max\{\mu_1(a) - \alpha\,\mu_2(a), 0\}, \sum_{a \in A} \max\{\mu_2(a) - \alpha\,\mu_1(a), 0\} \right\} \\
&\le \sum_{a \in A} \max\{\max\{\mu_1(a) - \alpha\,\mu_2(a), 0\}, \max\{\mu_2(a) - \alpha\,\mu_1(a), 0\}\} \\
&= \sum_{a \in A} \Delta^{\blacklozenge}_\alpha(\mu_1(a), \mu_2(a)).
\end{aligned}$$ ■

Proof of Lemma 4.5. See Lemma 5.5. ■

Proof of Lemma 4.6. See Lemma 5.7. ■

Proof of Theorem 4.7. See Lemma 5.6. ■

Proof of Lemma 4.9. See Lemma 5.8. ■

Proof of Lemma 4.10. See Lemma 5.12. ■

Proof of Lemma 4.13. Let E be a subset of B. The reasoning below shows that $f^\sharp(a_1)(E) - \alpha\alpha'\, f^\sharp(a_2)(E) \le f^\sharp(a_1)(S_1)$.

$$\begin{aligned}
f^\sharp(a_1)(E) &= \frac{\sum_{b \in E \cap \overline{S_1}} f(a_1)(b)}{\sum_{b \in B} f(a_1)(b)} + \frac{\sum_{b \in E \cap S_1} f(a_1)(b)}{\sum_{b \in B} f(a_1)(b)} \\
&\overset{(1)}{\le} \alpha' \frac{\sum_{b \in E \cap \overline{S_1}} f(a_2)(b)}{\sum_{b \in B} f(a_1)(b)} + \frac{\sum_{b \in E \cap S_1} f(a_1)(b)}{\sum_{b \in B} f(a_1)(b)} \\
&\overset{(2)}{\le} \alpha\alpha' \frac{\sum_{b \in E \cap \overline{S_1}} f(a_2)(b)}{\sum_{b \in B} f(a_2)(b)} + \frac{\sum_{b \in E \cap S_1} f(a_1)(b)}{\sum_{b \in B} f(a_1)(b)}
\end{aligned}$$

$$\begin{aligned}&\leq \alpha\alpha' \frac{\sum_{b\in E} f(a_2)(b)}{\sum_{b\in B} f(a_2)(b)} + \frac{\sum_{b\in S_1} f(a_1)(b)}{\sum_{b\in B} f(a_1)(b)}\\ &= \alpha\alpha'\, f^\sharp(a_2)(E) + f^\sharp(a_1)(S_1).\end{aligned}$$

Here (1) follows from the definition of $\overline{S_1}$, while (2) follows from hypothesis $\sum_{b\in B} f(a_2)(b) \leq \alpha \sum_{b\in B} f(a_1)(b)$. Similarly, we can show that $f^\sharp(a_2)(E) - \alpha\alpha' f^\sharp(a_1)(E) \leq f^\sharp(a_2)(S_2)$. The final result follows from Lemma 4.16. ■

Proof of Corollary 4.14. The first claim $\Delta^\blacklozenge_{\alpha^2}\big(f^\sharp(a_1), f^\sharp(a_2)\big) = 0$ follows from Lemma 4.13 by taking $\alpha' = \alpha$. Observe that hypothesis

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

implies $S_1 = S_2 = \emptyset$. Hence, $f^\sharp(a_1)(S_1) = f^\sharp(a_2)(S_2) = 0$, and thus $\Delta^\blacklozenge_{\alpha^2}\big(f^\sharp(a_1), f^\sharp(a_2)\big) = 0$.

The second claim $\Delta^\blacklozenge_{\alpha}\big(f^\sharp(a_1), f^\sharp(a_2)\big) = 0$ follows from instantiating the parameters (α, α') of Lemma 4.13 with $(1, \alpha)$. (The occurrence of α within pair $(1, \alpha)$ stands for the parameter of Corollary 4.14.) ■

Soundness Proof of Rule [exp]. Applying rule [rand], we are left to prove that for any pair of memories m_1, m_2 such that $m_1\, \Psi\, m_2$,

$$\Delta^\blacklozenge_{e^{2k\epsilon\mathbf{S}_s}}\Big(\mathcal{E}^\epsilon_{s,\mu}([\![a]\!]_\mathcal{E}\ m_1), \mathcal{E}^\epsilon_{s,\mu}([\![a]\!]_\mathcal{E}\ m_2)\Big) \leq 0. \tag{4.6}$$

Let $f(a)(b) = e^{\mu(b)\, s(a,b)\, \epsilon}$, $\alpha = e^{k\epsilon\mathbf{S}_s}$, $a_1 = [\![a]\!]_\mathcal{E}\ m_1$, and $a_2 = [\![a]\!]_\mathcal{E}\ m_2$.

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,

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

Hence, for all $b \in B$, $f(a_1)(b) \leq \alpha\, f(a_2)(b)$, and analogously $f(a_2)(b) \leq \alpha\, f(a_1)(b)$. Observe that (4.6) is equivalent to $\Delta^\blacklozenge_{\alpha^2}\big(f^\sharp(a_1), f^\sharp(a_2)\big) \leq 0$, which follows from Corollary 4.14. ■

Soundness Proof of Rule [norm]. Applying rule [rand], we are left to prove that for every pair of memories m_1, m_2 such that $m_1\, \Psi\, m_2$,

$$\Delta^\blacklozenge_{e^\epsilon}(\mathcal{N}([\![r]\!]_\mathcal{E}\ m_1, \sigma), \mathcal{N}([\![r]\!]_\mathcal{E}\ m_2, \sigma)) \leq \mathcal{B}\left(\sigma, \frac{\sigma\epsilon - k^2}{2k}\right) \leq \delta \tag{4.7}$$

We prove (4.7) by applying Lemma 4.13 with $A = B = \mathbb{Z}$, $f(a)(b) = e^{-|b-a|^2/\sigma}$, $a_1 = [\![r]\!]_{\mathcal{E}}\, m_1$, $a_2 = [\![r]\!]_{\mathcal{E}}\, m_2$, $\alpha = 1$ and $\alpha' = e^{\epsilon}$. We conclude by showing inequality

$$\max\left\{f^{\sharp}(a_1)(S_1), f^{\sharp}(a_2)(S_2)\right\} \leq \mathcal{B}\left(\sigma, \frac{\sigma\epsilon - k^2}{2k}\right) \tag{4.8}$$

(Here S_1 and S_2 are defined as in the statement of Lemma 4.13.) The reader can verify that (4.8) is entailed by $|a_1 - a_2| \leq k$, which follows from the first premise of the rule. ■

5

Security Analysis based on Arbitrary f-divergences

In Chapter 3 we considered an approximate version of the observational equivalence between probabilistic programs based on the notion of statistical distance and developed an equational theory to reason thereof. In the following chapter we introduced the notion of α-distance—which subsumes statistical distance—and showed that this equational theory can be generalized to a full-fledged relational Hoare logic α-pRHL that enables reasoning about the α-distance between probabilistic programs. In this chapter we show that this relational logic approach for reasoning about the distance between probabilistic programs is not confined to the statistical distance or α-distance, but can be extended to an important and well-known family of distance measures known as f-divergences.

Concretely, we generalize the α-pRHL logic to reason about arbitrary f-divergences. The resulting logic, coined f-pRHL, paves the way for introducing alternative security analyses based on measures different from the classic statistical distance (or the α-distance introduced herein). The benefits of this approach have already been confirmed e.g. in the context of cryptography, where Steinberger [2012] improved the security analysis of key-alternating ciphers using the Hellinger distance, a representative member of the class of f-divergences.

All the development in the present chapter is supported by correctness proofs outlined in Appendix 5.A. However, providing machine-checked versions of these proofs or tool support for f-pRHL is left as future work. Our aim here is to lay the theoretical foundations for supporting verified security based on arbitrary f-divergences.

To achieve this goal we take the following steps:

i) As a preliminary observation, we prove that the notion of α-distance used to characterize differential privacy is, in fact, an f-divergence. (The fact that the statistical distance is an f-divergences is already well-known.)

ii) We define a notion of composability between f-divergences which will be used later for introducing the sequential composition rule of f-pRHL. This notion of compos-

ability is inspired by—and generalizes to an important subset of f-divergences—the sequential composition theorem of differential privacy.

iii) We generalize the notion of lifting used in α-pRHL to f-divergences and prove its compatibility with composable f-divergences.

iv) We define f-pRHL, a relational Hoare logic for reasoning about f-divergences between programs, and prove its soundness.

For the sake of concreteness, we choose the asymmetric version f-pRHL$^{\blacklozenge}$ of the logic to be thoroughly developed; the modifications to implement the symmetric version are straightforward and briefly discussed in Section 5.2.3.

The remainder of the chapter is structured as follows. In Section 5.1 we review all the preliminaries to define our logic f-pRHL. In Section 5.2 we introduce the logic and in Appendix 5.A we sketch detailed proofs of all our results.

5.1 Preliminaries

5.1.1 The Family of f-divergences

In this section we recall the definition of f-divergences and show that the notion of α-distance used to characterize differential privacy belongs to the family of f-divergences.

Briefly, f-divergences represent distance measures between probability distributions. They were introduced independently by Csiszár and Ali & Silvey in the sixties and over the years, they have found multiple applications in diverse fields such as information theory, machine learning and cryptography.

Each distance measure in this class is basically defined by a convex function f. Formally, let $\mathcal{F}$ be the set of non-negative convex functions $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ such that f is continuous at 0 and $f(1) = 0$; each function in $\mathcal{F}$ induces a notion of distance between probability distributions as follows [Ali & Silvey, 1966; Csiszár, 1963]:

Definition 5.1 (f-divergence)**.** *Given $f \in \mathcal{F}$, the f-divergence $\Delta_f^{\blacklozenge}(\mu_1, \mu_2)$ between two distributions μ_1 and μ_2 in $\mathcal{D}(A)$ is defined as*

$$\Delta_f^{\blacklozenge}(\mu_1, \mu_2) \stackrel{\text{def}}{=} \sum_{a \in A} \mu_2(a)\, f\left(\frac{\mu_1(a)}{\mu_2(a)}\right).$$

The definition adopts the following conventions, which are used consistently in the remainder of the dissertation:

$$0\, f(0/0) = 0 \qquad \text{and} \qquad 0\, f(t/0) = t \lim_{x \to 0^+} x\, f(1/x) \quad \text{if } t > 0.$$

*Moreover, if $\Delta_f^{\blacklozenge}(\mu_1, \mu_2) \leq \delta$ we say that μ_1 and μ_2 are (f, δ)-*close*.*

In Definition 5.1, the restriction of $\mathcal{F}$ to functions that vanish at 1 is required to guarantee, e.g., that $\Delta_f^{\blacklozenge}(\mu,\mu) = 0$. The restriction to positive functions is usually omitted in the literature. In our case, it is required for technical reasons and can be shown not to affect the generality of our development.

Proposition 5.1. *Let $\mathcal{F}'$ be defined as $\mathcal{F}$, except that we allow $f \in \mathcal{F}'$ to take negative values. Then for every $f \in \mathcal{F}'$ there exists $g \in \mathcal{F}$ such that*

$$\Delta_f^{\blacklozenge}(\mu_1,\mu_2) = \Delta_g^{\blacklozenge}(\mu_1,\mu_2) + f'(1)\,(w(\mu_1) - w(\mu_2)).$$

Function g is given by $g(t) \stackrel{\text{def}}{=} f(t) - f'(1)\,(t-1)$; if f is not differentiable at 1, $f'(1)$ can be replaced with any value in the interval $[f_-(1), f_+(1)]$. (Here f'_- and f'_+ denote the left and right derivatives of f, whose existence and relative order can be guaranteed from the convexity of f.)

Said otherwise, $\Delta_f^{\blacklozenge}(\mu_1,\mu_2)$ and $\Delta_g^{\blacklozenge}(\mu_1,\mu_2)$ differ only by an additive term that accounts for the possibility that $w(\mu_1) \neq w(\mu_2)$. In any case, one can easily recover $\Delta_f^{\blacklozenge}(\mu_1,\mu_2)$ from $\Delta_g^{\blacklozenge}(\mu_1,\mu_2)$ and vice versa.

The class of f-divergences includes several popular instances; these comprise statistical distance, relative entropy (also known as Kullback-Leibler divergence), Hellinger distance and χ^2-divergence. In Figure 5.1 we summarize the convex function used to define each of them[1] and also include a simplified form, useful to compute the divergence.

***f*-divergence**	***f***	**Simplified Form**
Statistical distance	$\mathsf{SD}(t) = \frac{1}{2}\,\lvert t-1\rvert$	$\sum_{a\in A} \frac{1}{2}\,\lvert\mu_1(a) - \mu_2(a)\rvert$
Kullback-Leibler[2]	$\mathsf{KL}(t) = t\ln(t) - t + 1$	$\sum_{a\in A} \mu_1(a)\,\ln\left(\frac{\mu_1(a)}{\mu_2(a)}\right) + w(\mu_2) - w(\mu_1)$
Hellinger distance	$\mathsf{HD}(t) = \frac{1}{2}(\sqrt{t}-1)^2$	$\sum_{a\in A} \frac{1}{2}\left(\sqrt{\mu_1(a)} - \sqrt{\mu_2(a)}\right)^2$
χ^2-distance	$\chi^2(t) = (t-1)^2$	$\sum_{a\in A} \frac{(\mu_1(a) - \mu_2(a))^2}{\mu_2(a)}$

Figure 5.1: Examples of f-divergences.

In general, $\Delta_f^{\blacklozenge}$ does not define a metric. The symmetry axiom might be violated and the triangle inequality holds only if $\Delta_f^{\blacklozenge}$ is a (non-negative) multiple of the statistical distance. The identity of indiscernibles does not hold in general, but can be guaranteed if

[1] In case of negative functions, we previously apply the transformation mentioned in Proposition 5.1, so as to be consistent with our definition of f-divergences.

[2] Rigorously speaking, the function used for defining the Kullback-Leibler divergence should be given by $f(t) = t\ln(t) + t - 1$ if $t > 0$ and $f(t) = 1$ if $t = 0$ to guarantee its continuity at 0.

f is strictly convex at 1 (i.e. if there exists some interval around 1 in which f is not linear). The range of $\Delta_f^{\blacklozenge}$ is bounded from below and above by 0 and $f(0) + f^\star(0)$, respectively, where $f^\star(t) = \lim_{t\to 0^+} t\, f(1/t)$.[3] Both bounds are tight; the lower bound is attained e.g. when the compared distributions are equal while the upper bound is attained e.g. when the distributions have unitary mass and disjoint supports. In the case of the statistical distance, for instance, this gives the well-known result $0 \leq \Delta_{\mathsf{SD}}^{\blacklozenge}(\mu_1, \mu_2) \leq 1$ recalled in Section 3.1.

One key property of the f-divergences is a monotonicity result referred to as the *data processing inequality* [Pardo & Vajda, 1997]. Loosely speaking, it says that f-divergences never increase after probabilistic transformations. In our setting, this is captured by the following lemma:

Lemma 5.2 (Data Processing Inequality)**.** *Let μ_1,$\mu_2 \in \mathcal{D}(A)$, $M : A \to \mathcal{D}(B)$ and $f \in \mathcal{F}$. Then,*

$$\Delta_f^{\blacklozenge}(\mathsf{bind}\ \mu_1\ M, \mathsf{bind}\ \mu_2\ M) \leq \Delta_f^{\blacklozenge}(\mu_1, \mu_2)\,.$$

The instantiation of the data processing inequality to the statistical distance has been extensively used in our development of Chapter 3. Furthermore, we have seen that this inequality also holds true for the α-distance (see Lemma 4.2). We conclude the section showing that this property of the α-distance does not have an incidental origin independent of Lemma 5.2.

Concretely, we show that the notion of α-distance can be built as an f-divergence. This result will be of particular importance in the following section as it hints at borrowing the structure of the sequential composition theorem of differential privacy to define our notion of composition between f-divergences.

Lemma 5.3. *For every $\alpha \geq 1$, the α-distance $\Delta_\alpha^{\blacklozenge}(\mu_1, \mu_2)$ coincides with the f-divergence $\Delta_{\mathsf{AD}_\alpha}^{\blacklozenge}(\mu_1, \mu_2)$ associated to function $\mathsf{AD}_\alpha(t) \stackrel{\text{def}}{=} \max\{t - \alpha, 0\}$.*

Now, some basic properties of the α-distance such as its range of possible values, (anti-)monotonicity w.r.t. α or monotonicity w.r.t. the bind operator can be derived for free from standard properties of the f-divergences.

5.1.2 The Composition of f-divergences

The goal of this section is to introduce the notion of composability between f-divergences that will be used for establishing the sequential composition rule of f-pRHL$^{\blacklozenge}$.

The notion of f-divergence composability that we present has its origins in the sequential composition theorem of differential privacy. The core of this composition theorem is a composability result for the α-distance stated in the appendix of Chapter 4; let us recall it (we paraphrase this composability result for the asymmetric version of the α-distance; originally, it is stated for the symmetric version).

[3] If $f^\star(0) = \infty$, then $\Delta_f^{\blacklozenge}$ can take arbitrarily large values. Furthermore, ∞ is in the range of $\Delta_f^{\blacklozenge}$ in these cases.

Lemma 4.17 (α-distance Composition). *Let A and B be discrete sets. Then, for any $\mu_1, \mu_2 \in \mathcal{D}(A)$ and $M_1, M_2 : A \to \mathcal{D}(B)$,*

$$\Delta^{\blacklozenge}_{\alpha\alpha'}(\mathsf{bind}\ \mu_1\ M_1, \mathsf{bind}\ \mu_2\ M_2) \leq \Delta^{\blacklozenge}_{\alpha}(\mu_1, \mu_2) + \sup_a \Delta^{\blacklozenge}_{\alpha'}(M_1(a), M_2(a))\,. \tag{5.1}$$

In order to define our notion of composability between f-divergences we build on the observation that the α-distance is an f-divergence. We basically abstract the α-distances $\Delta^{\blacklozenge}_{\alpha}$, $\Delta^{\blacklozenge}_{\alpha'}$ and $\Delta^{\blacklozenge}_{\alpha\alpha'}$ in (5.1) as arbitrary f-divergences $\Delta^{\blacklozenge}_{f_1}$, $\Delta^{\blacklozenge}_{f_2}$ and $\Delta^{\blacklozenge}_{f_3}$ and augment the RHS with an additive term that accounts for the product between $\Delta^{\blacklozenge}_{f_1}(\mu_1, \mu_2)$ and $\sup_a \Delta^{\blacklozenge}_{f_2}(M_1(a), M_2(a))$. This term is weighted by a parameter $\gamma \geq 0$ that is 0 in the case of α-divergences. We elaborate on the motivation for this additional term shortly afterwards.

Our notion of f-divergence composability comprises two variants. The first considers f-divergences between arbitrary pair of distributions, while the second, weaker, considers f-divergences between distributions that have the same mass.

Definition 5.2 (f-divergence Composability)**.** *Let $f_1, f_2, f_3 \in \mathcal{F}$ and $\gamma \in \mathbb{R}^{\geq 0}$. We say that* (f_1, f_2, γ) strongly-composes into f_3 *iff for all $\mu_1, \mu_2 \in \mathcal{D}(A)$ and $M_1, M_2 : A \to \mathcal{D}(B)$, we have*

$$\begin{aligned}\Delta^{\blacklozenge}_{f_3}(\mathsf{bind}\ \mu_1\ M_1, \mathsf{bind}\ \mu_2\ M_2) \leq \Delta^{\blacklozenge}_{f_1}(\mu_1, \mu_2) + \sup_a \Delta^{\blacklozenge}_{f_2}(M_1(a), M_2(a)) \\ + \gamma\, \Delta^{\blacklozenge}_{f_1}(\mu_1, \mu_2)\ \sup_a \Delta^{\blacklozenge}_{f_2}(M_1(a), M_2(a))\,.\end{aligned}$$

If the above inequality holds provided $w(\mu_1) = w(\mu_2)$ and $w(M_1(a)) = w(M_2(a))$ for all $a \in A$, we say that (f_1, f_2, γ) weakly-composes into f_3.

The way in which we extend the composability of the α-distance to define our composability between f-divergences is motivated by the notion of *additivity*, which can be viewed as a more elementary form of self-composition of distance measures between distributions [Ebanks *et al.*, 1998, Ch. 5]. We say that a distance measure Δ_f is additive iff there exists $\gamma \in \mathbb{R}$ such that

$$\Delta_f\left(\mu_1 \times \mu_1', \mu_2 \times \mu_2'\right) \leq \Delta_f\left(\mu_1, \mu_2\right) + \Delta_f\left(\mu_1', \mu_2'\right) + \gamma\, \Delta_f\left(\mu_1, \mu_2\right)\, \Delta_f\left(\mu_1', \mu_2'\right).$$

It is easily seen that composability, as given in Definition 5.2, subsumes additivity.

To conclude the section, we show that our notion of composability is indeed satisfied by an important subset of the class of f-divergences. In particular, we show that all the f-divergences from Figure 5.1 are composable.

Theorem 5.4.

- $(\mathsf{AD}_{\alpha_1}, \mathsf{AD}_{\alpha_2}, 0)$ *strongly-composes into* $\mathsf{AD}_{\alpha_1\alpha_2}$*;*
- $(\mathsf{SD}, \mathsf{SD}, 0)$ *strongly-composes into* SD*;*

- $(\mathsf{KL}, \mathsf{KL}, 0)$ *weakly-composes into* KL*;*
- $(\mathsf{HD}, \mathsf{HD}, 0)$ *weakly-composes into* HD*;*
- $(\chi^2, \chi^2, 1)$ *weakly-composes into* χ^2*.*

Theorem 5.4 shows that the sequential composition theorem of differential privacy extends naturally to an important subset of the class of f-divergences. For this subset of f-divergences, Theorem 5.4 also subsumes the data processing inequality, which can be recovered by taking $M = M_1 = M_2$.

5.1.3 Lifting Relations to Distributions

The definition of valid α-pRHL judgments rests on the notion of the (α, δ)-lifting. The first step to define our relational logic f-pRHL$^\blacklozenge$ is to extend this notion of lifting to arbitrary f-divergences.

Definition 5.3 ((f,δ)-Lifting)**.** *Let $f \in \mathcal{F}$ and $\delta \in \mathbb{R}^{\geq 0}$. The (f,δ)-*lifting *of a relation $R \subseteq A \times B$ is the relation $\mathcal{L}^{\blacklozenge}_{f,\delta}(R) \subseteq \mathcal{D}(A) \times \mathcal{D}(B)$ such that $\mu_1 \, \mathcal{L}^{\blacklozenge}_{f,\delta}(R) \, \mu_2$ iff there exists a pair of distributions $\mu_L, \mu_R \in \mathcal{D}(A \times B)$ satisfying the following conditions:*

i) $\mathsf{range}\, R\, \mu_L \;\wedge\; \mathsf{range}\, R\, \mu_R$*;*
ii) $\pi_1(\mu_L) = \mu_1 \;\wedge\; \pi_2(\mu_R) = \mu_2$*;*
iii) $\Delta^{\blacklozenge}_f(\mu_L, \mu_R) \leq \delta$*.*

The distributions μ_L and μ_R are called the left *and* right witnesses *for the lifting $\mu_1 \, \mathcal{L}^{\blacklozenge}_{f,\delta}(R) \, \mu_2$, respectively.*

We next review some key properties of the (f,δ)-lifting, which are used to justify the results of the next section. Although these properties are direct translations of the properties of the (α,δ)-lifting studied in Section 4.1.3 (modulo the symmetric/asymmetric variant considered in each case), we paraphrase them in the more general setting of f-divergences to keep the chapter self-contained.

The first property characterizes liftings over equivalence relations, and will be used to show that f-divergences between the output of probabilistic programs can be characterized by f-pRHL$^\blacklozenge$ judgments.

Lemma 5.5 ((f,δ)-lifting of Equivalence Relations)**.** *Let R be an equivalence relation over A and let $\mu_1, \mu_2 \in \mathcal{D}(A)$. Then,*

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{f,\delta}(R) \, \mu_2 \iff \Delta^{\blacklozenge}_f(\mu_1/R, \mu_1/R) \leq \delta.$$

In particular, if R is the identity relation $\equiv$, we have

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{f,\delta}(\equiv) \, \mu_2 \iff \Delta^{\blacklozenge}_f(\mu_1, \mu_2) \leq \delta.$$

Our next result allows deriving closeness conditions between the probabilities $\mu_1(g_1)$ and $\mu_2(g_2)$ whenever μ_1 and μ_2 are related by the lifting of some relation R and functions g_1 and g_2 are R-equivalent (see p.51 for the definition of R-equivalent functions). In what follows, given a $[0,1]$-valued function g, we use $\overline{g}$ to denote its complementary function, i.e. $\overline{g}(t) \stackrel{\text{def}}{=} 1 - g(t)$.

Lemma 5.6 (Fundamental Property of the (f,δ)-lifting)**.** *Let $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$, and $R \subseteq A \times B$. Then, for any two functions $g_1 : A \to [0,1]$ and $g_2 : B \to [0,1]$,*

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{f,\delta}(R) \, \mu_2 \;\wedge\; g_1 =_R g_2 \implies \mu_2(g_2)\, f\left(\frac{\mu_1(g_1)}{\mu_2(g_2)}\right) + \mu_2(\overline{g_2})\, f\left(\frac{\mu_1(\overline{g_1})}{\mu_2(\overline{g_2})}\right) \leq \delta.$$

Let us discuss in more detail the kind of relationship that Lemma 5.6 allows establishing between probabilities $p_1 = \mu_1(g_1)$ and $p_2 = \mu_2(g_2)$. For the sake of simplicity, assume that μ_1 and μ_2 have unitary mass.[4] Then the conclusion of the above implication reduces to

$$p_2\, f\left(\frac{p_1}{p_2}\right) + (1-p_2)\, f\left(\frac{1-p_1}{1-p_2}\right) \leq \delta. \tag{5.2}$$

This inequality imposes certain closeness conditions between p_1 and p_2, and their nature is best understood by considering particular instances of f. For example, for the case of the statistical distance and Hellinger distance, inequality (5.2) simplifies to $|p_1 - p_2| \leq \delta$ and $\left|\sqrt{p_1} - \sqrt{p_2}\right| + \left|\sqrt{1-p_1} - \sqrt{1-p_2}\right| \leq 2\delta$, respectively. The way in which these formulae constrain the values of p_1 and p_2 is illustrated in Figure 5.2. It can be seen that for every value of p_1 there exists a bounded range $[m(p_1), M(p_1)]$ of possible values for p_2. Moreover, since M and m lie above and below the identity, p_1 also belongs to interval $[m(p_1), M(p_1)]$. This entails an implicit closeness constrain between the two probabilities, namely $|p_1 - p_2| \leq M(p_1) - m(p_1)$.

We now show that the (f,δ)-lifting is monotonous w.r.t. its both parameters f and δ, and its input relation.

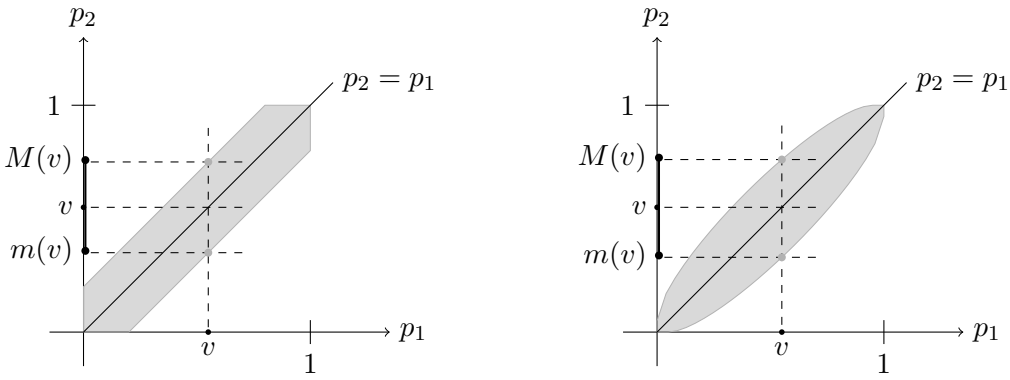
Lemma 5.7. *For all $f' \leq f$, $\delta \leq \delta'$, and relations $R \subseteq R'$,*

$$\mu_1 \, \mathcal{L}^{\blacklozenge}_{f,\delta}(R) \, \mu_2 \implies \mu_1 \, \mathcal{L}^{\blacklozenge}_{f',\delta'}(R') \, \mu_2.$$

The last property of the (f,δ)-lifting required to develop our f-pRHL logic states that, for composable f-divergences, the lifting operation is compatible with respect to the bind operator. This result will be the cornerstone for deriving the sequential composition rule of f-pRHL.

Lemma 5.8 ((f,δ)-lifting Composition)**.** *Let $f_1, f_2, f_3 \in \mathcal{F}$ and $\gamma \in \mathbb{R}^{\geq 0}$ such that (f_1, f_2, γ) strongly-composes into f_3. Moreover let $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$, $M_1 : A \to \mathcal{D}(A')$*

[4] If this is not the case, inequality (5.2) should be replaced with $p_2\, f\left(\frac{p_1}{p_2}\right) + (w(\mu_2) - p_2)\, f\left(\frac{w(\mu_1)-p_1}{w(\mu_2)-p_2}\right) \leq \delta$; the subsequent reasoning remains valid.



(a) Statistical distance (b) Hellinger distance

Figure 5.2: Closeness conditions between probabilities p_1 and p_2 established by inequality (5.2). Shaded area corresponds to the set of feasible solutions of the inequality. The highlighted interval $[m(v), M(v)]$ in the vertical axis corresponds to the set of possible values of p_2 for $p_1 = v$.

and $M_2 : B \to \mathcal{D}(B')$. *If* $\mu_1 \; \mathcal{L}^{\blacklozenge}_{f_1,\delta_1}(R_1)\, \mu_2$ *and* $M_1(a) \; \mathcal{L}^{\blacklozenge}_{f_2,\delta_2}(R_2)\, M_2(b)$ *whenever* $a \, R_1 \, b$, *then we have*

$$(\mathsf{bind}\ \mu_1\ M_1) \; \mathcal{L}^{\blacklozenge}_{f_3,\delta_3}(R_2)\, (\mathsf{bind}\ \mu_2\ M_2)$$

for $\delta_3 = \delta_1 + \delta_2 + \gamma\delta_1\delta_2$. *The result remains valid for weakly-composable* f_1, f_2 *and* f_3 *provided* $w(\mu_1) = w(\mu_2)$ *and* $w(M_1(a)) = w(M_2(b))$ *whenever* $a \, R_1 \, b$.

We conclude this section presenting an inductive characterization of the (f, δ)-lifting. This characterization is independent of the rest of our development, but may be handy in other applications that underly some notion of (weak) equivalence between probabilistic systems, e.g. bisimulation of probabilistic processes. This inductive characterization is stated in terms of Dirac sub-probability distributions. Given $k \in [0, 1]$ and $a \in A$ we use $\wr a \int^k$ to denote the Dirac distribution over A that assigns probability k to a and null probability to any other value in A. (Observe that the unit operator is a shorthand for $\wr \cdot \int^1$.)

Lemma 5.9 (Inductive Characterization of the (f, δ)-lifting)**.** *For any relation* $R \subseteq A \times B$, *the* (f, δ)*-lifting* $\mathcal{L}^{\blacklozenge}_{f,\delta}(R)$ *is the smallest relation over* $\mathcal{D}(A) \times \mathcal{D}(B)$ *that satisfies the following rules:*

$$\frac{a \, R \, b \qquad k' \, f\left(\frac{k}{k'}\right) \leq \delta}{\wr a \int^k \; \mathcal{L}^{\blacklozenge}_{f,\delta}(R)\; \wr b \int^{k'}} \qquad\qquad \frac{\mu_i \; \mathcal{L}^{\blacklozenge}_{f,\delta_i}(R)\, \mu'_i \;\; \forall i \in I \qquad \sum_{i\in I} \delta_i \leq \delta}{\left(\sum_{i\in I} \mu_i\right) \; \mathcal{L}^{\blacklozenge}_{f,\delta}(R)\; \left(\sum_{i\in I} \mu'_i\right)}$$

In the RHS rule we assume that distributions $\sum_{i\in I} \mu_i$ *and* $\sum_{i\in I} \mu'_i$ *are well defined, i.e.* $\sum_{i\in I} w(\mu_i) \leq 1$ *and* $\sum_{i\in I} w(\mu'_i) \leq 1$.

The lifting of R defined inductively by the above rules can be viewed as a generalization to an approximate setting through f-divergences of the exact lifting of Deng *et al.* [2009]. Moreover, Lemma 5.9 extends a previous result in [Deng & Du, 2011] about the equivalence between the exact liftings of Deng *et al.* [2009] and Jonsson *et al.* [2001] to the approximate setting.

5.2 A Relational Logic for f-divergences

Building on the results of the previous section, we now define our relational Hoare logic f-pRHL$^\blacklozenge$, which enables reasoning about f-divergences between probabilistic programs.

5.2.1 Judgments

Judgments in f-pRHL$^\blacklozenge$ have the same shape as in α-pRHL, except that they are parametrized by an arbitrary f-divergence, rather than by an α-divergence. Specifically, they are of the form

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

where c_1 and c_2 are programs, Ψ and Φ are relational assertions, $f \in \mathcal{F}$ and $\delta \in \mathbb{R}^{\geq 0}$.

The notion of validity in f-pRHL$^\blacklozenge$ is also defined as in α-pRHL: an f-pRHL$^\blacklozenge$ judgment is valid iff for every pair of memories related by the pre-condition Ψ, the corresponding pair of program output distributions is related by the (f,δ)-lifting of the post-condition Φ.

Definition 5.4 (Validity in f-pRHL)**.** *A judgment* $c_1 \sim_{f,\delta} c_2 : \Psi \Rightarrow \Phi$ *is* valid*, written* $\models^\blacklozenge c_1 \sim_{f,\delta} c_2 : \Psi \Rightarrow \Phi$*, iff for every pair of memories* m_1, m_2*,*

$$m_1 \,\Psi\, m_2 \implies (\llbracket c_1 \rrbracket\, m_1)\, \mathcal{L}^\blacklozenge_{f,\delta}(\Phi)\, (\llbracket c_2 \rrbracket\, m_2).$$

f-pRHL$^\blacklozenge$ judgments provide a characterization of the f-divergences between probabilistic programs. Concretely, judgments with the identity relation as post-condition can be used to derive (f,δ)-closeness results.

Theorem 5.10. *If* $\models^\blacklozenge c_1 \sim_{f,\delta} c_2 : \Psi \Rightarrow \equiv$*, then for all memories* m_1, m_2*,*

$$m_1 \,\Psi\, m_2 \implies \Delta^\blacklozenge_f(\llbracket c_1 \rrbracket\, m_1, \llbracket c_2 \rrbracket\, m_2) \leq \delta. \tag{5.3}$$

The converse of this theorem, which holds true on account of Lemma 5.5, shows that f-pRHL$^\blacklozenge$ judgments completely characterizes (f,δ)-closeness properties between probabilistic programs.

Judgments with the identity relation as post-condition can also be used to model approximate information flow in a probabilistic setting. To see this, let Ψ be an equivalence relation on initial states and assume for a moment that $\delta = 0$ and $\Delta^\blacklozenge_f$ satisfies the identity of indiscernibles (i.e. $\Delta^\blacklozenge_f(\mu_1,\mu_2) = 0 \iff \mu_1 = \mu_2$). Judgment $\models^\blacklozenge c \sim_{f,\delta} c : \Psi \Rightarrow \equiv$

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 the output of c will only be able to determine the initial state up to its Ψ-equivalence class. If we drop the initial assumptions about $\Delta_f^{\blacklozenge}$ and δ, the judgment models an approximate variant of this information flow property of c.

f-pRHL$^{\blacklozenge}$ can also be used to derive continuity properties of probabilistic programs. We assume a continuity model in which programs are executed on random inputs, i.e. distributions of initial memories, and we use f-divergences as metrics to compare program inputs and outputs.

Lemma 5.11. *Let $f_1, f_2, f_3 \in \mathcal{F}$ and $\gamma \in \mathbb{R}^{\geq 0}$ such that (f_1, f_2, γ) strongly-composes into f_3. If $\models^{\blacklozenge} c_1 \sim_{f_2,\delta_2} c_2 : \equiv \Rightarrow \equiv$, then for any two distributions of initial memories μ_1 and μ_2,*

$$\Delta^{\blacklozenge}_{f_1}(\mu_1, \mu_2) \leq \delta_1 \implies \Delta^{\blacklozenge}_{f_3}(\mathsf{bind}\ \mu_1\ [\![c_1]\!], \mathsf{bind}\ \mu_2\ [\![c_2]\!]) \leq \delta_3,$$

where $\delta_3 = \delta_1 + \delta_2 + \gamma\delta_1\delta_2$. The same result holds for weakly-composable f-divergences provided $w(\mu_1) = w(\mu_2)$ and $w([\![c_1]\!]\, m) = w([\![c_2]\!]\, m)$ for every $m \in \mathcal{M}$.

Finally, we can use judgments with arbitrary post-conditions to relate the probabilities of single events in two programs.

Lemma 5.12. *If $\models^{\blacklozenge} c_1 \sim_{f,\delta} c_2 : \Psi \Rightarrow \Phi$, then for every pair of Ψ-related memories m_1, m_2 and every pair of Φ-equivalent functions $g_1, g_2 : \mathcal{M} \to [0,1]$,*

$$([\![c_2]\!]\, m_2)(g_2)\ f\left(\frac{([\![c_1]\!]\, m_1)(g_1)}{([\![c_2]\!]\, m_2)(g_2)}\right) + ([\![c_2]\!]\, m_2)(\overline{g_2})\ f\left(\frac{([\![c_1]\!]\, m_1)(\overline{g_1})}{([\![c_2]\!]\, m_2)(\overline{g_2})}\right) \leq \delta.$$

If we specialize g_1 and g_2 to the characteristic function of two events A and B, we obtain a rule for relating the probabilities of A and B. In stating the rule, we use $\overline{A}$ (resp. $\overline{B}$) to denote the complementary event of A (resp. B).

$$\frac{m_1\ \Psi\ m_2 \qquad \models^{\blacklozenge} c_1 \sim_{f,\delta} c_2 : \Psi \Rightarrow \Phi \qquad \Phi \implies (A\langle 1\rangle \iff B\langle 2\rangle)}{\Pr\left[c_2(m_2) : B\right]\ f\left(\frac{\Pr[c_1(m_1):A]}{\Pr[c_2(m_2):B]}\right) + \Pr[c_2(m_2) : \overline{B}]\ B\left(\frac{\Pr[c_1(m_1):\overline{A}]}{\Pr[c_2(m_2):\overline{B}]}\right) \leq \delta}\ [\mathsf{PrEq\text{-}\Delta^{\blacklozenge}_f}]$$

The validity of Lemma 5.12 rests on the fundamental property of the (f,δ)-lifting. When discussing this property we have seen that an inequality like the one in the conclusion of rule [PrEq-$\Delta^{\blacklozenge}_f$] establishes certain closeness conditions between the probabilities $p_1 = \Pr\left[c_1(m_1) : A\right]$ and $p_2 = \Pr\left[c_2(m_2) : B\right]$. (Observe that, in general, we have $\Pr[c(m) : \overline{E}] = w([\![c]\!]\, m) - \Pr\left[c(m) : E\right]$; therefore the conclusion of rule [PrEq-$\Delta^{\blacklozenge}_f$] reduces to a formula on p_1 and p_2, only.) Specifically, we have seen that there exist functions $m, M : [0,1] \to [0,1]$ such that for every p_1 in $[0,1]$, both the values of p_1 and p_2 lie within the interval $[m(p_1), M(p_1)]$. For the case of the statistical distance and Hellinger distance, these closeness conditions are illustrated in Figure 5.2. The closeness conditions have also

been studied by Kifer & Lin [2010] in their effort to characterize all "sensible" definitions of statistical privacy. They show, among others, that if f is self-composable, then M is a concave function above the identity and $m(p) = 1 - M(1-p)$. This relationship between m and M is reflected in the symmetry w.r.t. the identity of the shaded areas in Figure 5.2.

5.2.2 Proof System

Figure 5.3 presents a set of core rules for reasoning about the validity of f-pRHL$^{\blacklozenge}$ judgments. All rules are transpositions of the α-pRHL rules listed in Figure 4.1.

The translation from α-pRHL to f-pRHL is straightforward. However, the f-pRHL rules [s-seq] and [s-while] for the sequential composition and bounded loops include an extra premise requiring the composability of the involved f-divergences. When composing arbitrary f-divergences—contrary to the case of plain α-divergences—upper-bounds do not just "add up"; for the computation of the final bound it is also necessary to account for the product between the individual bounds (see Definition 5.2). This translates into more compound final bounds for this pair of rules, namely $\delta_3 = \delta_1 + \delta_2 + \gamma\delta_1\delta_2$ for [s-seq] and $\delta_n = n\delta + R_n$ for [s-while] (in α-pRHL we have $\delta_3 = \delta_1 + \delta_2$ and $\delta_n = n\delta$).

All the rules are thoroughly discussed in Section 4.2. We will only examine in more detail the rule for bounded loops. Rule [s-while] relates two loops that execute in lockstep. The bound it establishes depends on the maximal number of iterations of the loops; we assume given a loop variant e that increases across iterations and guarantees the loop termination upon exceeding a constant value n. We briefly explain the side conditions: $(f_1, \ldots, f_n)$ is γ-strongly-composable iff (f_i, f_1, γ) strongly-composes into f_{i+1} for every $1 \leq i < n$. Moreover, $(f_1, \ldots, f_n)$ is monotonic iff $f_i \leq f_{i+1}$ for $1 \leq i < n$; monotonicity is required for those executions where the loops terminate in less than n iteration. Note that the rule is given for $n \geq 2$; specialized rules exist for $n = 1$ and $n = 0$.

Rules [s-seq] and [s-while] only deals with strongly-composable f-divergences. Their counterparts for weakly-composable f divergences are presented in Figure 5.4. Both variants of the rules have the same structure, with the exception that those dealing with weakly-composable f divergences include an extra side condition requiring that the output distributions of the respective programs have the same mass. Formally, this requirement is captured by the notion of program *eq-weightness*; given a relation over program states Θ, we say that two programs c_1 and c_2 are *eq-weighted* w.r.t. Θ iff for all memories m_1, m_2, we have $m_1 \,\Theta\, m_2 \implies w([\![c_1]\!]\, m_1) = w([\![c_2]\!]\, m_2)$. Recall that for programs that are assertion free and sample values from proper probability distributions, the mass of the output distribution represents the probability of termination. Therefore, for this class of programs the eq-weightness relation can be understood as property on the termination behaviour of programs; concretely, two programs are eq-weighted iff they terminate with the same probability (modulo relational pre-condition).

Eq-weightness assertions can be derived within f-pRHL$^{\blacklozenge}$, by considering an exact variant of the logic. By taking $\delta = 0$ and choosing f so that $\Delta_f^{\blacklozenge}$ satisfies the identity of indis-

$$\frac{\forall m_1, m_2 \bullet\ m_1\,\Psi\,m_2 \implies (m_1\,\{[\![e_1]\!]_{\mathcal{E}}\ m_1/x_1\})\ \Phi\ (m_2\,\{[\![e_2]\!]_{\mathcal{E}}\ m_2/x_2\})}{\models^{\blacklozenge}\ x_1 \leftarrow e_1 \sim_{f,0} x_2 \leftarrow e_2 : \Psi \Rightarrow \Phi}\ [\mathsf{assn}]$$

$$\frac{\begin{array}{c}\forall m_1, m_2 \bullet\ m_1\,\Psi\,m_2 \implies \Delta_f^{\blacklozenge}([\![d_1]\!]_{\mathcal{D}\mathcal{E}}\ m_1, [\![d_2]\!]_{\mathcal{D}\mathcal{E}}\ m_2) \le \delta \\ m_1\,\Psi'\,m_2 \stackrel{\text{def}}{=} \exists v_1, v_2 \bullet\ (m_1\,\{v_1/x_1\})\,\Psi\,(m_2\,\{v_2/x_2\})\end{array}}{\models^{\blacklozenge}\ x_1 \xleftarrow{\$} d_1 \sim_{f,\delta} x_2 \xleftarrow{\$} d_2 : \Psi \Rightarrow x_1\langle 1\rangle = x_2\langle 2\rangle \wedge \Psi'}\ [\mathsf{rand}]$$

$$\frac{\Psi \implies b_1\langle 1\rangle = b_2\langle 2\rangle}{\models^{\blacklozenge}\ \mathsf{assert}\ b_1 \sim_{f,0} \mathsf{assert}\ b_2 : \Psi \Rightarrow \Psi \wedge b_1\langle 1\rangle}\ [\mathsf{assert}] \qquad \frac{}{\models^{\blacklozenge}\ \mathsf{skip} \sim_{f,0} \mathsf{skip} : \Psi \Rightarrow \Psi}\ [\mathsf{skip}]$$

$$\frac{\begin{array}{c}\Psi \implies b_1\langle 1\rangle = b_2\langle 2\rangle \\ \models^{\blacklozenge}\ c_1 \sim_{f,\delta} c_2 : \Psi \wedge b_1\langle 1\rangle \Rightarrow \Phi \qquad \models^{\blacklozenge}\ c_1' \sim_{f,\delta} c_2' : \Psi \wedge \neg b_1\langle 1\rangle \Rightarrow \Phi\end{array}}{\models^{\blacklozenge}\ \mathsf{if}\ b_1\ \mathsf{then}\ c_1\ \mathsf{else}\ c_1' \sim_{f,\delta} \mathsf{if}\ b_2\ \mathsf{then}\ c_2\ \mathsf{else}\ c_2' : \Psi \Rightarrow \Phi}\ [\mathsf{cond}]$$

$$\frac{\begin{array}{c}(f_1, \ldots, f_n) \text{ is } \gamma\text{-strongly-composable and monotonic} \\ \delta_n \stackrel{\text{def}}{=} n\delta + R_n \ \text{ where } R_n \stackrel{\text{def}}{=} \gamma\delta^2 \textstyle\sum_{i=0}^{n-2} (i+1)(1+\gamma\delta)^{n-2-i} \\ \Theta \implies b_1\langle 1\rangle = b_2\langle 2\rangle \qquad \Theta \wedge e\langle 1\rangle \ge n \implies \neg b_1\langle 1\rangle \\ \models^{\blacklozenge}\ c_1 \sim_{f_1,\delta} c_2 : \Theta \wedge b_1\langle 1\rangle \wedge e\langle 1\rangle = k \Rightarrow \Theta \wedge e\langle 1\rangle > k\end{array}}{\models^{\blacklozenge}\ \mathsf{while}\ b_1\ \mathsf{do}\ c_1 \sim_{f_n,\delta_n} \mathsf{while}\ b_2\ \mathsf{do}\ c_2 : \Theta \wedge e\langle 1\rangle \ge 0 \Rightarrow \Theta \wedge \neg b_1\langle 1\rangle}\ [\mathsf{s\text{-}while}]$$

$$\frac{\begin{array}{c}(f_1, f_2, \gamma) \text{ strongly-composes into } f_3 \qquad \delta_3 \stackrel{\text{def}}{=} \delta_1 + \delta_2 + \gamma\delta_1\delta_2 \\ \models^{\blacklozenge}\ c_1 \sim_{f_1,\delta_1} c_2 : \Psi \Rightarrow \Phi' \qquad \models^{\blacklozenge}\ c_1' \sim_{f_2,\delta_2} c_2' : \Phi' \Rightarrow \Phi\end{array}}{\models^{\blacklozenge}\ c_1; c_1' \sim_{f_3,\delta_3} c_2; c_2' : \Psi \Rightarrow \Phi}\ [\mathsf{s\text{-}seq}]$$

$$\frac{\begin{array}{c}\Psi \implies \Psi' \quad \Phi' \implies \Phi \quad f \le f' \quad \delta' \le \delta \\ \models^{\blacklozenge}\ c_1 \sim_{f',\delta'} c_2 : \Psi' \Rightarrow \Phi'\end{array}}{\models^{\blacklozenge}\ c_1 \sim_{f,\delta} c_2 : \Psi \Rightarrow \Phi}\ [\mathsf{weak}] \qquad \frac{\begin{array}{c}\models^{\blacklozenge}\ c_1 \sim_{f,\delta} c_2 : \Psi \wedge \Theta \Rightarrow \Phi \\ \models^{\blacklozenge}\ c_1 \sim_{f,\delta} c_2 : \Psi \wedge \neg\Theta \Rightarrow \Phi\end{array}}{\models^{\blacklozenge}\ c_1 \sim_{f,\delta} c_2 : \Psi \Rightarrow \Phi}\ [\mathsf{case}]$$

Figure 5.3: Core proof rules of f-pRHL$^{\blacklozenge}$.

cernible[5], we obtain an specialization of f-pRHL$^{\blacklozenge}$ for exact—rather than approximate—reasoning between probabilistic programs. A key property of this specialization is that a judgment of the form $\models^{\blacklozenge}\ c_1 \sim_{f,0} c_2 : \Psi \Rightarrow \mathsf{true}$ entails the eq-weightness of c_1 and c_2 w.r.t. Ψ. Alternatively, we can rely on the exact logic pRHL to derive eq-weightness assertions, since a judgment $\models^{\blacklozenge}\ c_1 \sim_{f,0} c_2 : \Psi \Rightarrow \Phi$ in this specialization of f-pRHL is equivalent to the pRHL judgment $\models c_1 \sim c_2 : \Psi \Rightarrow \Phi$.

[5] All f-divergences listed in Figure 5.1 satisfy the identity of indiscernible.

$$\frac{\begin{array}{c}(c_1,c_2)\text{ eq-weighted w.r.t.}\Psi\wedge b_1\langle 1\rangle\wedge b_2\langle 2\rangle\\ (f_1,\ldots,f_n)\text{ is }\gamma\text{-weakly-composable and monotonic}\\ \delta_n \stackrel{\text{def}}{=} n\delta+R_n\ \text{ where }R_n\stackrel{\text{def}}{=}\gamma\delta^2\textstyle\sum_{i=0}^{n-2}(i+1)(1+\gamma\delta)^{n-2-i}\\ \Theta\implies b_1\langle 1\rangle=b_2\langle 2\rangle\qquad \Theta\wedge e\langle 1\rangle\geq n\implies\neg b_1\langle 1\rangle\\ \models^{\blacklozenge} c_1\sim_{f_1,\delta} c_2:\Theta\wedge b_1\langle 1\rangle\wedge e\langle 1\rangle=k\Rightarrow\Theta\wedge e\langle 1\rangle>k\end{array}}{\models^{\blacklozenge}\ \mathsf{while}\ b_1\ \mathsf{do}\ c_1\sim_{f_n,\delta_n}\mathsf{while}\ b_2\ \mathsf{do}\ c_2:\Theta\wedge e\langle 1\rangle\geq 0\Rightarrow\Theta\wedge\neg b_1\langle 1\rangle}\ [\mathsf{w\text{-}while}]$$

$$\frac{\begin{array}{c}(c_1,c_2)\text{ eq-weighted w.r.t.}\Psi\qquad (c_1',c_2')\text{ eq-weighted w.r.t.}\Phi'\\ (f_1,f_2,\gamma)\text{ weakly-composes into }f_3\qquad \delta_3\stackrel{\text{def}}{=}\delta_1+\delta_2+\gamma\delta_1\delta_2\\ \models^{\blacklozenge} c_1\sim_{f_1,\delta_1} c_2:\Psi\Rightarrow\Phi'\qquad \models^{\blacklozenge} c_1'\sim_{f_2,\delta_2} c_2':\Phi'\Rightarrow\Phi\end{array}}{\models^{\blacklozenge} c_1;c_1'\sim_{f_3,\delta_3} c_2;c_2':\Psi\Rightarrow\Phi}\ [\mathsf{w\text{-}seq}]$$

Figure 5.4: Proof rules of f-pRHL$^{\blacklozenge}$ for weakly-composable f-divergences.

We now analyze the soundness of our proof system of Figures 5.3 and 5.4. The soundness proof follows the same line as that of the α-pRHL proof system, which was discussed in Section 4.2 and formally verified in Coq. The proof rests basically on properties of the (f,δ)-lifting. For instance, the crux of the soundness proofs of rules [rand] and [s-seq] is an application of Lemmas 5.5 and 5.8, respectively. The soundness of rule [weak] relays on the monotonicity property of the (f,δ)-lifting from Lemma 5.7 while the soundness of rules [assn] and [skip] depends on a sufficient condition to build exact versions of the (f,δ)-lifting (see Proposition 5.20 in Appendix 5.A).

5.2.3 Symmetric Logic

We can also define a symmetric version of f-pRHL$^{\blacklozenge}$ by slightly modifying the definition of the (f,δ)-lifting to include the additional requirement $\Delta_f^{\blacklozenge}(\mu_R,\mu_L)\leq\delta$ on the witness distributions μ_L and μ_R. Rigorously speaking, α-pRHL is instance of this symmetric logic (and α-pRHL$^{\blacklozenge}$ is an instance of f-pRHL$^{\blacklozenge}$). All rules in the proof system of Figures 5.3 and 5.4 remain unchanged, except for the rule for random assignments [rand] whose premise now requires the additional inequality $\Delta_f^{\blacklozenge}(\llbracket d_2\rrbracket_{\mathcal{DE}}\ m_2,\llbracket d_1\rrbracket_{\mathcal{DE}}\ m_1)\leq\delta$ to be checked. Theorem 5.10 and Lemmas 5.11 and 5.12 are also adapted accordingly; for instance Theorem 5.10 now allows deriving claims of the form $\Delta_f^{\blacklozenge}(\llbracket c_1\rrbracket\ m_1,\llbracket c_2\rrbracket\ m_2)\leq\delta\ \wedge\ \Delta_f^{\blacklozenge}(\llbracket c_2\rrbracket\ m_2,\llbracket c_1\rrbracket\ m_1)\leq\delta$.

The symmetric version of the logic can be understood in a more uniform manner as being constructed using a symmetric notion of closeness between distributions, namely by requiring that $\max\{\Delta_f^{\blacklozenge}(\mu_1,\mu_2),\Delta_f^{\blacklozenge}(\mu_2,\mu_1)\}\leq\delta$ for two distributions μ_1 and μ_2 to be (f,δ)-close (cf. Equation (4.2)), rather than just requiring $\Delta_f^{\blacklozenge}(\mu_1,\mu_2)\leq\delta$, as stated in Definition 5.1, and consistently employed in the development of f-pRHL$^{\blacklozenge}$.

5.A Appendix

In this section we outline proofs of all the results in the present chapter. We first introduce some auxiliary lemmas and then turn to the proofs.

5.A.1 Auxiliary Lemmas

Proposition 5.13 (Generalized Log-Sum Inequality)**.** *Let $f : I \to \mathbb{R}$ be a convex function. For any two sequences of non-negative numbers $a_1, \dots, a_n$ and $b_1, \dots, b_n$ such that a_i/b_i belongs to I for $i = 1, \dots, n$ we have*

$$\sum_{1 \le i \le n} b_i \, f\left(\frac{a_i}{b_i}\right) \ge b \, f\left(\frac{a}{b}\right),$$

where $a \stackrel{\text{def}}{=} \sum_{1 \le i \le n} a_i$ and $b \stackrel{\text{def}}{=} \sum_{1 \le i \le n} b_i$. The result remains valid for $n = \infty$ as long as $\sum_{1 \le i \le \infty} a_i$, $\sum_{1 \le i \le \infty} b_i$ and $\sum_{1 \le i \le \infty} b_i \, f(a_i/b_i)$ exist and are finite.

Proof. The generalized log-sum inequality is derived from the Jensen's inequality (see e.g. [Rockafellar, 1997, Th. 4.3]) as follows:

$$\sum_{1 \le i \le n} b_i \, f\left(\frac{a_i}{b_i}\right) = b \sum_{1 \le i \le n} \frac{b_i}{b} \, f\left(\frac{a_i}{b_i}\right) \ge b \, f\left(\sum_{1 \le i \le n} \frac{b_i}{b} \frac{a_i}{b_i}\right) = b \, f\left(\frac{a}{b}\right). \qquad \blacksquare$$

Proposition 5.14. *Let $f : I \to \mathbb{R}$ be a convex function, differentiable at $x_0 \in I$. Then*

$$f(x) \ge f(x_0) + f'(x_0)(x - x_0) \qquad \forall x \in I.$$

When f is not differentiable at x_0, the above inequality remains valid if we replace $f'(x_0)$ with any value in the interval $[f'_-(x_0), f'_+(x_0)]$.

Proof. See e.g. [van Tiel, 1984, Th. 1.6]. $\blacksquare$

Proposition 5.15. *For every $f, g \in \mathcal{F}$, $c \in \mathbb{R}$, $\mu_1, \mu_2 \in \mathcal{D}(A)$ and equivalence relation $R \subseteq A \times A$,*

i) $g(t) = f(t) - c\,(t-1) \implies \Delta_g^{\blacklozenge}(\mu_1, \mu_2) = \Delta_f^{\blacklozenge}(\mu_1, \mu_2) - c\,(w(\mu_1) - w(\mu_2))$;

ii) $f \le g \implies \Delta_f^{\blacklozenge}(\mu_1, \mu_2) \le \Delta_g^{\blacklozenge}(\mu_1, \mu_2)$;

iii) $\Delta_f^{\blacklozenge}(\mu_1, \mu_2) \ge \Delta_f^{\blacklozenge}(\mu_1/R, \mu_2/R)$.

Proof. i) follows from a straightforward computation; ii) is immediate from the definition of f-divergences and the non-negativity of f and g; finally iii) is a corollary of Proposition 5.13. $\blacksquare$

Proposition 5.16. *For every $\alpha \ge 1$ and μ_1, μ_2 in $\mathcal{D}(A)$,*

$$\Delta_\alpha^{\blacklozenge}(\mu_1, \mu_2) = \mu_1(A_0) - \alpha \, \mu_2(A_0),$$

where $A_0 = \{a \in A \mid \mu_1(a) \ge \alpha \, \mu_2(a)\}$.

Proof. It follows the same lines as that of Lemma 4.16. ■

Proposition 5.17. *Let $\mu \in \mathcal{D}(A)$ satisfy predicate* range $P\,\mu$. *Then, for any $M : A \to \mathcal{D}(B)$, any predicate Q over B and any pair of functions $f, g : A \to [0,1]$,*

i) $(\forall a \bullet P(a) \implies f(a) = g(a)) \implies \mu(f) = \mu(g)$;

ii) $(\forall a \bullet P(a) \implies \mathsf{range}\, Q\, M(a)) \implies \mathsf{range}\, Q\, (\mathsf{bind}\, \mu\, M)$.

Proof.

i) From the definition of range we have

$$\mu(f) = \sum_{a \in A} \mu(a)\, f(a) = \sum_{\substack{a \in A \\ P(a)}} \mu(a)\, f(a) = \sum_{\substack{a \in A \\ P(a)}} \mu(a)\, g(a) = \sum_{a \in A} \mu(a)\, g(a) = \mu(g).$$

ii) Let $b \in B$ such that

$$(\mathsf{bind}\, \mu\, M)(b) = \sum_{a \in A} \mu(a)\, M(a)(b) = \sum_{\substack{a \in A \\ P(a)}} \mu(a)\, M(a)(b) > 0.$$

Then, there must exists $a' \in A$ such that $P(a')$ and $M(a')(b) > 0$. By hypothesis $\forall a \bullet P(a) \implies \mathsf{range}\, Q\, M(a)$, formula $\mathsf{range}\, Q\, M(a')$ holds, which implies $Q(b)$. ■

Proposition 5.18. *Let (f_1, f_2, γ) strongly-compose into f_3. Then for all predicate $R \subseteq A$, all $\mu_1, \mu_2 \in \mathcal{D}(A)$ such that* range $R\,\mu_1$ *and* range $R\,\mu_2$, *and all $M_1, M_2 : A \to \mathcal{D}(B)$,*

$$\begin{aligned}\Delta^{\blacklozenge}_{f_3}(\mathsf{bind}\, \mu_1\, M_1, \mathsf{bind}\, \mu_2\, M_2) \leq \Delta^{\blacklozenge}_{f_1}(\mu_1, \mu_2) + \sup_{a|R(a)} \Delta^{\blacklozenge}_{f_2}(M_1(a), M_2(a)) \\ + \gamma\, \Delta^{\blacklozenge}_{f_1}(\mu_1, \mu_2) \sup_{a|R(a)} \Delta^{\blacklozenge}_{f_2}(M_1(a), M_2(a))\,.\end{aligned}$$

The result remains valid when f_1, f_2 and f_3 satisfy only weak-composability, provided $w(\mu_1) = w(\mu_2)$ and $w(M_1(a)) = w(M_2(b))$ for all pair of R-related a, b.

Proof. Consider the mappings $M_1', M_2' : A \to \mathcal{D}(B)$ such that $M_1'(a) = M_1(a)$ if $R(a)$ and $M_1'(a) = \mu_0$ otherwise. Then we have

$$\Delta^{\blacklozenge}_{f_3}(\mathsf{bind}\, \mu_1\, M_1, \mathsf{bind}\, \mu_2\, M_2) \leq \Delta^{\blacklozenge}_{f_3}\big(\mathsf{bind}\, \mu_1\, M_1', \mathsf{bind}\, \mu_2\, M_2'\big)$$

since $\mathsf{bind}\, \mu_i\, M_i = \mathsf{bind}\, \mu_i\, M_i'$ for $i = 1, 2$. The proof concludes by applying the definition of f-divergence composability in the RHS of the above equation and observing that

$$\sup_a \Delta^{\blacklozenge}_{f_2}\big(M_1'(a), M_2'(a)\big) = \sup_{a|R(a)} \Delta^{\blacklozenge}_{f_2}(M_1(a), M_2(a))\,.$$ ■

Proposition 5.19. *Given $R \subseteq A \times B$, we let $\mathcal{I}_f^\delta(R)$ be the relation over $\mathcal{D}(A) \times \mathcal{D}(B)$ that satisfies the following rules:*

$$\frac{a\, R\, b \qquad k'\, f\left(\frac{k}{k'}\right) \leq \delta}{\lbag a \rbag^k\, \mathcal{I}_f^\delta(R)\, \lbag b \rbag^{k'}} \qquad\qquad \frac{\mu_i\, \mathcal{I}_f^{\delta_i}(R)\, \mu_i' \;\; \forall i \in I \qquad \sum_{i\in I} \delta_i \leq \delta}{\left(\sum_{i\in I} \mu_i\right)\, \mathcal{I}_f^\delta(R)\, \left(\sum_{i\in I} \mu_i'\right)}$$

(In the RHS rule we assume that distributions $\sum_{i\in I} \mu_i$ and $\sum_{i\in I} \mu_i'$ are well defined, i.e. $\sum_{i\in I} w(\mu_i) \leq 1$ and $\sum_{i\in I} w(\mu_i') \leq 1$). Then given $\mu \in \mathcal{D}(A)$ and $\mu' \in \mathcal{D}(B)$,

$$\mu\, \mathcal{I}_f^\delta(R)\, \mu'$$

if and only if μ and μ' can be decomposed as

$$\mu = \sum_{i\in I} \lbag a_i \rbag^{k_i} \qquad \text{and} \qquad \mu' = \sum_{i\in I} \lbag b_i \rbag^{k_i'}$$

for some index set I, a_i in A, b_i in B and k_i, k_i' in $[0,1]$, where

$$a_i\, R\, b_i \;\; \text{for all } i \in I \quad \text{and} \quad \sum_{i\in I} k_i'\, f\left(\frac{k_i}{k_i'}\right) \leq \delta.$$

Proof. The "if" direction is immediate. The "only if" direction can be proved by induction on the height of the derivation trees. Formally, the induction argument relays on the notion of rank of a lifting. The *rank* of a lifting $\mu\, \mathcal{I}_f^\delta(R)\, \mu'$ is defined as the height of the shortest derivation tree that relates μ and μ'. Observe that if the rank of $\mu\, \mathcal{I}_f^\delta(R)\, \mu'$ is n, then $\mu\, \mathcal{I}_f^\delta(R)\, \mu'$ was derived either with an application of the LHS rule, and the purported decomposition of μ and μ' is immediate, or with an application of the RHS rule, and each of the liftings $\mu_i\, \mathcal{I}_f^{\delta_i}(R)\, \mu_i'$ in its premise has rank strictly less than n. We can thus use well-founded induction on the rank of the liftings to prove the "only if" direction of the proposition. ■

Proposition 5.20. *For any relation $R \subseteq A \times B$ and $k \in [0,1]$,*

$$a\, R\, b \implies \lbag a \rbag^k\, \mathcal{L}_{f,0}^{\blacklozenge}(R)\, \lbag b \rbag^k.$$

Proof. The proof is straightforward by considering distribution $k^{-1}(\lbag a \rbag^k \times \lbag b \rbag^k)$ as the left and right witness of the lifting $\lbag a \rbag^k\, \mathcal{L}_{f,0}^{\blacklozenge}(R)\, \lbag b \rbag^k$ if $k > 0$ and distribution μ_0 if $k = 0$. ■

5.A.2 Proofs

Proof of Proposition 5.1. First, we show that g does belong to $\mathcal{F}$. The convexity of g follows from the fact that convex functions are closed under addition and linear functions are convex. The continuity of g at 0 and the fact that it vanishes at 1 follow from f satisfying the same properties. The non-negativity of g can be derived using Proposition 5.14. A

direct application of this proposition says that $g(t) \geq g(1) + g'(1)\,(t-1)$; the result follows from $g(1) = g'(1) = 0$. To conclude observe that the purported relationship between $\Delta_f^{\blacklozenge}(\mu_1, \mu_2)$ and $\Delta_g^{\blacklozenge}(\mu_1, \mu_2)$ can be proved with an application of Proposition 5.15.i). ■

Proof of Lemma 5.2. The result follows from the reasoning below.

$$
\begin{aligned}
&\Delta_f^{\blacklozenge}(\mathsf{bind}\,\mu_1\,M, \mathsf{bind}\,\mu_2\,M) \\
&\qquad = \sum_{b\in B} (\mathsf{bind}\,\mu_2\,M)(b)\; f\left(\frac{(\mathsf{bind}\,\mu_1\,M)(b)}{(\mathsf{bind}\,\mu_2\,M)(b)}\right) \\
&\qquad = \sum_{b\in B} \left(\sum_{a\in A} \mu_2(a)\,M(a)(b)\right) f\left(\frac{\sum_{a\in A}\mu_1(a)\,M(a)(b)}{\sum_{a\in A}\mu_2(a)\,M(a)(b)}\right) \\
&\qquad \overset{(1)}{\leq} \sum_{b\in B}\sum_{a\in A} \mu_2(a)\,M(a)(b)\; f\left(\frac{\mu_1(a)\,M(a)(b)}{\mu_2(a)\,M(a)(b)}\right) \\
&\qquad \overset{(2)}{=} \sum_{a\in A} \mu_2(a)\; f\left(\frac{\mu_1(a)}{\mu_2(a)}\right) w(M(a)) \leq \Delta_f^{\blacklozenge}(\mu_1, \mu_2)\,.
\end{aligned}
$$

Here inequality (1) is an instance of Proposition 5.13 and step (2) holds by a simple reordering of the terms in the series. ■

Proof of Lemma 5.3. Verifying that AD_α belongs to $\mathcal{F}$ whenever $\alpha \geq 1$ is a simple matter. For proving the main claim we rely on Proposition 5.16.

$$
\Delta_{\mathsf{AD}_\alpha}^{\blacklozenge}(\mu_1, \mu_2) = \sum_{a\in A} \mu_2(a) \max\left\{\frac{\mu_1(a)}{\mu_2(a)} - \alpha, 0\right\} = \sum_{\substack{a\in A \\ \mu_1(a)\geq \alpha\,\mu_2(a)}} \mu_1(a) - \alpha\,\mu_2(a) = \Delta_\alpha^{\blacklozenge}(\mu_1, \mu_2)\,.
$$

■

Proof of Theorem 5.4.

α-distance: see Lemma 4.17.

Statistical Distance: composability follows from the reasoning below:

$$
\begin{aligned}
&\Delta_{\mathsf{SD}}^{\blacklozenge}(\mathsf{bind}\,\mu_1\,M_1, \mathsf{bind}\,\mu_2\,M_2) \\
&\qquad = \sum_{b\in B} \frac{1}{2} \left| \sum_{a\in A} \mu_1(a)\,M_1(a)(b) - \sum_{a\in A} \mu_2(a)\,M_2(a)(b) \right| \\
&\qquad = \sum_{b\in B} \frac{1}{2} \left| \sum_{a\in A} (\mu_1(a) - \mu_2(a))\,M_1(a)(b) + \sum_{a\in A} \mu_2(a)\,(M_1(a)(b) - M_2(a)(b)) \right|
\end{aligned}
$$

$$
\begin{aligned}
&\leq \sum_{b\in B} \frac{1}{2} \left(\sum_{a\in A} |\mu_1(a) - \mu_2(a)|\ M_1(a)(b) + \sum_{a\in A} \mu_2(a)\ |M_1(a)(b) - M_2(a)(b)| \right) \\
&= \sum_{a\in A} \frac{1}{2} |\mu_1(a) - \mu_2(a)|\ w(M_1(a)) + \sum_{a\in A} \mu_2(a) \sum_{b\in B} \frac{1}{2} |M_1(a)(b) - M_2(a)(b)| \\
&\leq \Delta^{\blacklozenge}_{\mathsf{SD}}(\mu_1, \mu_2) + \sup_a\ \Delta^{\blacklozenge}_{\mathsf{SD}}(M_1(a), M_2(a))\,.
\end{aligned}
$$

Kullback-Leibler: We use the simplified form $\sum_{a\in A} \nu_1(a)\ \ln\left(\frac{\nu_1(a)}{\nu_2(a)}\right)$ from Figure 5.1 to compute the Kullback-Leibler divergence between any pair of distributions ν_1 and ν_2. (Observe that the term $w(\nu_2) - w(\nu_1)$ vanishes since the notion of weak-composability considers only distributions ν_1 and ν_2 with the same mass.) Then we have

$$
\begin{aligned}
&\Delta^{\blacklozenge}_{\mathsf{KL}}(\mathsf{bind}\ \mu_1\ M_1, \mathsf{bind}\ \mu_2\ M_2) \\
&\quad= \sum_{b\in B} \left(\sum_{a\in A} \mu_1(a)\, M_1(a)(b) \right) \ln\left(\frac{\sum_{a\in A} \mu_1(a)\, M_1(a)(b)}{\sum_{a\in A} \mu_2(a)\, M_2(a)(b)} \right) \\
&\quad\overset{(1)}{\leq} \sum_{b\in B} \sum_{a\in A} \mu_1(a)\, M_1(a)(b)\ \ln\left(\frac{\mu_1(a)\, M_1(a)(b)}{\mu_2(a)\, M_2(a)(b)} \right) \\
&\quad= \sum_{a\in A} \mu_1(a)\ \ln\left(\frac{\mu_1(a)}{\mu_2(a)}\right) w(M_1(a)) + \sum_{a\in A} \mu_1(a) \sum_{b\in B} M_1(a)(b)\ \ln\left(\frac{M_1(a)(b)}{M_2(a)(b)}\right) \\
&\quad\leq \Delta^{\blacklozenge}_{\mathsf{KL}}(\mu_1, \mu_2) + \sup_a\ \Delta^{\blacklozenge}_{\mathsf{KL}}(M_1(a), M_2(a))\,.
\end{aligned}
$$

Here, the inequality step (1) holds by Proposition 5.13.

Hellinger distance: the proof of composability rests on two auxiliary properties. The first is an alternative characterization of the Hellinger distance, namely, $\Delta^{\blacklozenge}_{\mathsf{HD}}(\nu, \nu') = \frac{1}{2}(w(\nu_1) + w(\nu_2)) - \sum_a \sqrt{\nu(a)\,\nu'(a)}$. The second is identity $\sqrt{\mu_1(a)\,\mu_2(a)} = \mu_3(a) - \mu_2(a)\,\mathsf{HD}\left(\frac{\mu_1(a)}{\mu_2(a)}\right)$, where $\mu_3(a) \overset{\text{def}}{=} \frac{1}{2}(\mu_1(a) + \mu_2(a))$. Furthermore, let us define $w_\mu \overset{\text{def}}{=} w(\mu_1) = w(\mu_2)$ and $w_M \overset{\text{def}}{=} w(M_1(a)) = w(M_2(a))$ for every $a \in A$. Then

$$
\begin{aligned}
&\Delta^{\blacklozenge}_{\mathsf{HD}}(\mathsf{bind}\ \mu_1\ M_1, \mathsf{bind}\ \mu_2\ M_2) \\
&\quad= w_\mu w_M - \sum_{b\in B} \sqrt{\left(\sum_{a\in A} \mu_1(a) M_1(a)(b) \right) \left(\sum_{a\in A} \mu_2(a) M_2(a)(b) \right)} \\
&\quad\overset{(1)}{\leq} w_\mu w_M - \sum_{b\in B} \sum_{a\in A} \sqrt{\mu_1(a) M_1(a)(b) \mu_2(a) M_2(a)(b)} \\
&\quad= w_\mu w_M - \sum_{a\in A} \sqrt{\mu_1(a)\mu_2(a)} \sum_b \sqrt{M_1(a)(b) M_2(a)(b)}
\end{aligned}
$$

$$
\begin{aligned}
&= w_\mu w_M - \sum_{a\in A}\left(\mu_3(a) - \mu_2(a)\,\mathsf{HD}\left(\frac{\mu_1(a)}{\mu_2(a)}\right)\right)\left(w_M - \Delta^{\blacklozenge}_{\mathsf{HD}}(M_1(a),M_2(a))\right)\\
&\leq w_M\,\Delta^{\blacklozenge}_{\mathsf{HD}}(\mu_1,\mu_2) + \sum_{a\in A}\mu_3(a)\,\Delta^{\blacklozenge}_{\mathsf{HD}}(M_1(a),M_2(a))\\
&\leq \Delta^{\blacklozenge}_{\mathsf{HD}}(\mu_1,\mu_2) + \sup_a\,\Delta^{\blacklozenge}_{\mathsf{HD}}(M_1(a),M_2(a))\,.
\end{aligned}
$$

In the above reasoning, step (1) is justified by the Cauchy-Schwarz inequality.

χ^2-*distance:* Using the same argument as for the proof of the α-distance composability (see Lemma 4.17), we have

$$\Delta^{\blacklozenge}_{\chi^2}(\mathsf{bind}\,\mu_1\,M_1,\mathsf{bind}\,\mu_2\,M_2) \leq \Delta^{\blacklozenge}_{\chi^2}(\theta_1,\theta_2)\,,$$

where distributions $\theta_1,\theta_2 \in \mathcal{D}(A\times B)$ are defined as $\theta_1(a,b) \stackrel{\text{def}}{=} \mu_1(a)\,M_1(a)(b)$ and $\theta_2(a,b) \stackrel{\text{def}}{=} \mu_2(a)\,M_2(a)(b)$. Now let us bound $\Delta^{\blacklozenge}_{\chi^2}(\theta_1,\theta_2)$. To this end, we rely on the characterization of the χ^2-distance $\Delta^{\blacklozenge}_{\chi^2}(\nu_1,\nu_2) = \sum_{a\in A}\frac{\nu_1^2(a)}{\nu_2(a)} - w(\nu_1)$ when $w(\nu_1) = w(\nu_2)$. Moreover, we define $w_\mu \stackrel{\text{def}}{=} w(\mu_1) = w(\mu_2)$ and $w_M \stackrel{\text{def}}{=} w(M_1(a)) = w(M_2(a))$ for every $a\in A$.

$$
\begin{aligned}
\Delta^{\blacklozenge}_{\chi^2}(\theta_1,\theta_2) &= \sum_{(a,b)\in A\times B}\frac{\mu_1^2(a)\,M_1^2(a)(b)}{\mu_2(a)\,M_2(a)(b)} - w_\mu w_M\\
&= \sum_{a\in A}\frac{\mu_1^2(a)}{\mu_2(a)}\sum_{b\in B}\frac{M_1^2(a)(b)}{M_2(a)(b)} - w_\mu w_M\\
&= \sum_{a\in A}\frac{\mu_1^2(a)}{\mu_2(a)}\left(\Delta^{\blacklozenge}_{\chi^2}(M_1(a),M_2(a)) + w_M\right) - w_\mu w_M\\
&\leq \left(\sum_{a\in A}\frac{\mu_1^2(a)}{\mu_2(a)}\right)\left(\sup_a\,\Delta^{\blacklozenge}_{\chi^2}(M_1(a),M_2(a)) + w_M\right) - w_\mu w_M\\
&= \left(\Delta^{\blacklozenge}_{\chi^2}(\mu_1,\mu_2) + w_\mu\right)\left(\sup_a\,\Delta^{\blacklozenge}_{\chi^2}(M_1(a),M_2(a)) + w_M\right) - w_\mu w_M\\
&\leq \Delta^{\blacklozenge}_{\chi^2}(\mu_1,\mu_2) + \sup_a\,\Delta^{\blacklozenge}_{\chi^2}(M_1(a),M_2(a))\,.
\end{aligned}
$$

■

Proof of Lemma 5.5.
"Only If" direction: Let μ_L and μ_R be a pair of left and right witnesses of the lifting $\mu_1\,\mathcal{L}^{\blacklozenge}_{f,\delta}(R)\,\mu_2$. Then we have $\mathsf{range}\,R\,\mu_L$, $\mathsf{range}\,R\,\mu_R$, $\pi_1(\mu_L) = \mu_1$, $\pi_2(\mu_R) = \mu_2$ and $\Delta^{\blacklozenge}_f(\mu_L,\mu_R) \leq \delta$. Now

$$\Delta^{\blacklozenge}_f(\mu_1/R,\mu_2/R)$$

$$
\begin{aligned}
&= \sum_{A_0 \in A/R} \mu_2(A_0)\, f\left(\frac{\mu_1(A_0)}{\mu_2(A_0)}\right) = \sum_{A_0 \in A/R} \pi_2(\mu_R)(A_0)\, f\left(\frac{\pi_1(\mu_L)(A_0)}{\pi_2(\mu_R)(A_0)}\right) \\
&\overset{(1)}{=} \sum_{A_0 \in A/R} \pi_1(\mu_R)(A_0)\, f\left(\frac{\pi_1(\mu_L)(A_0)}{\pi_1(\mu_R)(A_0)}\right) = \Delta_f^{\blacklozenge}\left(\pi_1(\mu_L)^{A/R}, \pi_1(\mu_R)^{A/R}\right) \\
&\overset{(2)}{\leq} \Delta_f^{\blacklozenge}(\pi_1(\mu_L), \pi_1(\mu_R)) \overset{(3)}{\leq} \Delta_f^{\blacklozenge}(\mu_L, \mu_R) \leq \delta.
\end{aligned}
$$

To justify equality $\pi_2(\mu_R)(A_0) = \pi_1(\mu_R)(A_0)$ in step (1) we apply Proposition 5.17.i) with hypothesis $\mathsf{range}\, R\, \mu_R$. Steps (2) and (3) are direct applications of Proposition 5.15.iii) and Lemma 5.2 respectively.
"If" direction: We propose

$$
\mu_L(a_1, a_2) = \begin{cases} \dfrac{\mu_1(a_1)\,\mu_2(a_2)}{\mu_2([a_1])} & \text{if } a_1\, R\, a_2 \ \wedge\ \mu_2([a_1]) \neq 0 \\ 0 & \text{otherwise} \end{cases}
$$

$$
\mu_R(a_1, a_2) = \begin{cases} \dfrac{\mu_1(a_1)\,\mu_2(a_2)}{\mu_1([a_2])} & \text{if } a_1\, R\, a_2 \ \wedge\ \mu_1([a_2]) \neq 0 \\ 0 & \text{otherwise} \end{cases}
$$

as witnesses for the lifting $\mu_1\, \mathcal{L}_{f,\delta}^{\blacklozenge}(R)\, \mu_2$. We now verify the properties that μ_L and μ_R must meet. Property $\mathsf{range}\, R\, \mu_L \wedge \mathsf{range}\, R\, \mu_R$ follows immediately from the definitions of μ_L and μ_R. Some simple computations yield $\pi_1(\mu_L) = \mu_1$ and $\pi_2(\mu_R) = \mu_2$. Finally, we bound $\Delta_f^{\blacklozenge}(\mu_L, \mu_R)$ as follows:

$$
\begin{aligned}
\Delta_f^{\blacklozenge}(\mu_L, \mu_R) &= \sum_{(a_1,a_2) \in A \times A} \mu_R(a_1, a_2)\, f\left(\frac{\mu_L(a_1, a_2)}{\mu_R(a_1, a_2)}\right) \\
&\overset{(1)}{=} \sum_{(a_1,a_2) \in R} \mu_R(a_1, a_2)\, f\left(\frac{\mu_L(a_1, a_2)}{\mu_R(a_1, a_2)}\right) \\
&\overset{(2)}{=} \sum_{A_0 \in A/R} \sum_{a_1 \in A_0} \sum_{a_2 \in A_0} \mu_R(a_1, a_2)\, f\left(\frac{\mu_L(a_1, a_2)}{\mu_R(a_1, a_2)}\right) \\
&\overset{(3)}{=} \sum_{A_0 \in A/R} \sum_{\substack{a_1 \in A_0 \\ \mu_2([a_1]) \neq 0}} \sum_{\substack{a_2 \in A_0 \\ \mu_1([a_2]) \neq 0}} \frac{\mu_1(a_1)\,\mu_2(a_2)}{\mu_1([a_2])}\, f\left(\frac{\mu_1([a_2])}{\mu_2([a_1])}\right) \\
&= \sum_{A_0 \in A/R} \frac{1}{\mu_1(A_0)}\, f\left(\frac{\mu_1(A_0)}{\mu_2(A_0)}\right) \sum_{\substack{a_1 \in A_0 \\ \mu_2([a_1]) \neq 0}} \mu_1(a_1) \sum_{\substack{a_2 \in A_0 \\ \mu_1([a_2]) \neq 0}} \mu_2(a_2) \\
&\leq \sum_{A_0 \in A/R} \mu_2(A_0)\, f\left(\frac{\mu_1(A_0)}{\mu_2(A_0)}\right) = \Delta_f^{\blacklozenge}(\mu_1/R, \mu_2/R) \leq \delta.
\end{aligned}
$$

Here (1) holds since $\mathsf{range}\, R\, \mu_R$; (2) is a sum reordering while (3) holds because for every equivalence class $A_0 \in A/R$, $\mu_1(A_0) = 0$ implies $\mu_L(a_1, a_2) = \mu_R(a_1, a_2) = 0$ for every $a_1, a_2 \in A_0$, and similarly for μ_2. ■

Proof of Lemma 5.6. Let μ_L and μ_R be a pair of left and right witnesses of the lifting $\mu_1\, \mathcal{L}^{\blacklozenge}_{f,\delta}(R)\, \mu_2$. Then we have $\mathsf{range}\, R\, \mu_L \wedge \mathsf{range}\, R\, \mu_R$, $\pi_1(\mu_L) = \mu_1 \wedge \pi_2(\mu_R) = \mu_2$ and $\Delta^{\blacklozenge}_f(\mu_L, \mu_R) \leq \delta$. Now

$$
\begin{aligned}
&\mu_2(g_2)\, f\left(\frac{\mu_1(g_1)}{\mu_2(g_2)}\right) + \mu_2(\overline{g_2})\, f\left(\frac{\mu_1(\overline{g_1})}{\mu_2(\overline{g_2})}\right) \\
&\qquad = \pi_2(\mu_R)(g_2)\, f\left(\frac{\pi_1(\mu_L)(g_1)}{\pi_2(\mu_R)(g_2)}\right) + \pi_2(\mu_R)(\overline{g_2})\, f\left(\frac{\pi_1(\mu_L)(\overline{g_1})}{\pi_2(\mu_R)(\overline{g_2})}\right) \\
&\qquad \overset{(1)}{=} \pi_1(\mu_R)(g_1)\, f\left(\frac{\pi_1(\mu_L)(g_1)}{\pi_1(\mu_R)(g_1)}\right) + \pi_1(\mu_R)(\overline{g_1})\, f\left(\frac{\pi_1(\mu_L)(\overline{g_1})}{\pi_1(\mu_R)(\overline{g_1})}\right) \\
&\qquad = \left(\sum_{a\in A} \pi_1(\mu_R)(a)\, g_1(a)\right)\, f\left(\frac{\sum_{a\in A} \pi_1(\mu_L)(a)\, g_1(a)}{\sum_{a\in A} \pi_1(\mu_R)(a)\, g_1(a)}\right) + \\
&\qquad\quad \left(\sum_{a\in A} \pi_1(\mu_R)(a)\, \overline{g_1}(a)\right)\, f\left(\frac{\sum_{a\in A} \pi_1(\mu_L)(a)\, \overline{g_1}(a)}{\sum_{a\in A} \pi_1(\mu_R)(a)\, \overline{g_1}(a)}\right) \\
&\qquad \overset{(2)}{\leq} \sum_{a\in A} \pi_1(\mu_R)(a)\, f\left(\frac{\pi_1(\mu_L)(a)}{\pi_1(\mu_R)(a)}\right) = \Delta^{\blacklozenge}_f(\pi_1(\mu_L), \pi_1(\mu_R)) \\
&\qquad \overset{(3)}{\leq} \Delta^{\blacklozenge}_f(\mu_L, \mu_R) \leq \delta.
\end{aligned}
$$

Step (1) amounts to showing that $\pi_1(\mu_R)(g_1) = \pi_2(\mu_R)(g_2)$ and $\pi_1(\mu_R)(\overline{g_1}) = \pi_2(\mu_R)(\overline{g_2})$. We prove each equality applying Proposition 5.17.i) and the premises of each application are are discharged using hypotheses $\mathsf{range}\, R\, \mu_R$ and $g_1 =_R g_2$. Steps (2) and (3) are direct applications of Proposition 5.13 and Lemma 5.2. ■

Proof of Lemma 5.7. The proof proceeds by taking the witnes distributions from $\mu_1\, \mathcal{L}^{\blacklozenge}_{f,\delta}(R)\, \mu_2$ to relate μ_1 and μ_2 by $\mathcal{L}^{\blacklozenge}_{f',\delta'}(R')$, and relies on the monotoniticy of the f-divergences w.r.t. f (see Proposition 5.15.ii)) and the monotonicity of operator $\mathsf{range}\, R$ w.r.t. predicate R. ■

Proof of Lemma 5.8. Let $\mu_L \in \mathcal{D}(A \times B)$ (resp. μ_R) be a left (resp. right) witness for the lifting $\mu_1\, \mathcal{L}^{\blacklozenge}_{f_1,\delta_1}(R)\, \mu_2$ and let $M_L : A \times B \to \mathcal{D}(A' \times B')$ (resp. M_R) map R_1-related values a, b to a right (resp. left) witness distribution for the lifting $M_1(a)\, \mathcal{L}^{\blacklozenge}_{f_2,\delta_2}(R_2)\, M_2(b)$. (The action of M_L and M_R over pairs outside R_1 is irrelevant for our purposes.) Then we have

i) $\mathsf{range}\, R_1\, \mu_L \wedge \mathsf{range}\, R_1\, \mu_R$;
ii) $\pi_1(\mu_L) = \mu_1 \wedge \pi_2(\mu_R) = \mu_2$;

iii) $\Delta^{\blacklozenge}_{f_1}(\mu_L, \mu_R) \leq \delta_1$;
iv) $a\, R_1\, b \implies \mathsf{range}\, R_2\, M_L(a,b) \wedge \mathsf{range}\, R_2\, M_R(a,b)$;
v) $a\, R_1\, b \implies \pi_1(M_L(a,b)) = M_1(a) \wedge \pi_2(M_R(a,b)) = M_2(b)$;
vi) $a\, R_1\, b \implies \Delta^{\blacklozenge}_{f_2}(M_L(a,b), M_R(a,b)) \leq \delta_2$.

We claim that distributions $\mathsf{bind}\, \mu_L\, M_L$ and $\mathsf{bind}\, \mu_R\, M_R$ are valid left and right witnesses for the lifting $(\mathsf{bind}\, \mu_1\, M_1)\, \mathcal{L}^{\blacklozenge}_{f_3,\delta_3}(R_2)\, (\mathsf{bind}\, \mu_2\, M_2)$. First, properties $\mathsf{range}\, R_2\, (\mathsf{bind}\, \mu_L\, M_L)$ and $\mathsf{range}\, R_2\, (\mathsf{bind}\, \mu_R\, M_R)$ follow from Proposition 5.17.ii) and hypotheses i) and iv). Second, in view of hypothesis ii), equality $\pi_1(\mathsf{bind}\, \mu_L\, M_L) = \mathsf{bind}\, \mu_1\, M_1$ can be restated as $\pi_1(\mathsf{bind}\, \mu_L\, M_L) = \mathsf{bind}\, \pi_1(\mu_L)\, M_1$ and follows from an application of Proposition 5.17.i) with hypotheses i) and v); equality $\pi_2(\mathsf{bind}\, \mu_R\, M_R) = \mathsf{bind}\, \mu_2\, M_2$ is shown with a similar reasoning. Finally, the upper bound $\Delta^{\blacklozenge}_{f_3}(\mathsf{bind}\, \mu_L\, M_L, \mathsf{bind}\, \mu_R\, M_R) \leq \delta_3$ is derived from Proposition 5.18 using hypotheses i), iii) and vi). ■

Proof of Lemma 5.9. Lemma 5.9 can be paraphrased by saying that $\mu\, \mathcal{L}^{\blacklozenge}_{f,\delta}(R)\, \mu'$ if and only if $\mu\, \mathcal{I}^{\delta}_{f}(R)\, \mu'$, where $\mathcal{I}^{\delta}_{f}(R)$ is defined as in Proposition 5.19. To prove this equivalence we adopt the alternative characterization of $\mathcal{I}^{\delta}_{f}(R)$ given in Proposition 5.19.
"Only if" direction: Let $\mu_L, \mu_R \in \mathcal{D}(A \times B)$ be a pair of right and left witnesses for the lifting $\mu\, \mathcal{L}^{\blacklozenge}_{f,\delta}(R)\, \mu'$; then we have

i) $\mathsf{range}\, R\, \mu_L \wedge \mathsf{range}\, R\, \mu_R$;
ii) $\pi_1(\mu_L) = \mu \wedge \pi_2(\mu_R) = \mu'$;
iii) $\Delta^{\blacklozenge}_{f}(\mu_L, \mu_R) \leq \delta$.

The proof proceeds by taking the index set $I = R$ and for each $(\alpha, \beta) \in I$,

$$a_{(\alpha,\beta)} = \alpha \qquad b_{(\alpha,\beta)} = \beta \qquad k_{(\alpha,\beta)} = \mu_L(\alpha,\beta) \qquad k'_{(\alpha,\beta)} = \mu_R(\alpha,\beta)$$

and showing that they satisfy properties $\mu = \sum_{i \in I} \lbag a_i \rbag^{k_i}$, $\mu' = \sum_{i \in I} \lbag b_i \rbag^{k'_i}$, $a_i\, R\, b_i$ for all $i \in I$ and $\sum_{i \in I} k'_i\, f\left(\frac{k_i}{k'_i}\right) \leq \delta$. The first property about the decomposition of μ and μ' is derived from the reasoning below (and from a similar reasoning for the decomposition of μ'):

$$\sum_{i \in I} \lbag a_i \rbag^{k_i}(a) \;=\; \sum_{(\alpha,\beta) \in R} \lbag \alpha \rbag^{\mu_L(\alpha,\beta)}(a) \;=\; \sum_{(a,\beta) \in R} \mu_L(a,\beta) \overset{(1)}{=} \pi_1(\mu_L)(a) \overset{(2)}{=} \mu(a).$$

Here equalities (1) and (2) holds in view of hypotheses i) and ii), respectively. The second property relating a_i and b_i is immediate from the definition of the index set I. The last property $\sum_{i \in I} k'_i\, f\left(\frac{k_i}{k'_i}\right) \leq \delta$ follows from hypotheses i) and iii).

"If" direction: Assume that μ and μ' can be decomposed as $\mu = \sum_{i \in I} \lbag a_i \rbag^{k_i}$ and $\mu' = \sum_{i \in I} \lbag b_i \rbag^{k'_i}$ for some index set I, a_i in A, b_i in B and k_i, k'_i in $[0,1]$, where $a_i\, R\, b_i$ for all

$i \in I$ and $\sum_{i\in I} k'_i\, f\left(\frac{k_i}{k'_i}\right) \leq \delta$. We conclude that $\mu\, \mathcal{L}^{\blacklozenge}_{f,\delta}(R)\, \mu'$ by showing that distributions $\mu_L, \mu_R \in \mathcal{D}(A\times B)$ defined as

$$\mu_L(a,b) \stackrel{\text{def}}{=} \sum_{\substack{i\in I\\(a_i,b_i)=(a,b)}} k_i \qquad \text{and} \qquad \mu_R(a,b) \stackrel{\text{def}}{=} \sum_{\substack{i\in I\\(a_i,b_i)=(a,b)}} k'_i$$

are valid left and right witnesses of the lifting. To prove that $\mathsf{range}\, R\, \mu_L$, consider a pair (a,b) such that $\mu_L(a,b) > 0$. There must exists an index $i \in I$ such that $k_i > 0$ and $(a,b) = (a_i,b_i)$. From hypothesis $a_i\, R\, b_i$ for all $i \in I$ we conclude that $a\, R\, b$, which entails formula $\mathsf{range}\, R\, \mu_L$. In a similar way we show that $\mathsf{range}\, R\, \mu_R$. The equivalence between $\pi_1(\mu_L)$ and μ is justified as follows:

$$\pi_1(\mu_L)(a) = \sum_{b\in B} \sum_{\substack{i\in I\\(a_i,b_i)=(a,b)}} k_i = \sum_{\substack{i\in I\\a_i=a}} k_i = \sum_{i\in I} \lbag a_i \rbag^{k_i}(a) = \mu(a).$$

We prove the counterpart equivalence $\pi_2(\mu_R) = \mu'$ following a similar reasoning. Finally the closeness condition $\Delta^{\blacklozenge}_f(\mu_L,\mu_R) \leq \delta$ is derived from the computations below.

$$\begin{aligned}
\Delta^{\blacklozenge}_f(\mu_L,\mu_R) &= \sum_{(a,b)\in A\times B} \mu_R(a,b)\, f\left(\frac{\mu_L(a,b)}{\mu_R(a,b)}\right)\\
&= \sum_{(a,b)\in A\times B} \left(\sum_{\substack{i\in I\\(a_i,b_i)=(a,b)}} k'_i\right) f\left(\frac{\sum_{\substack{i\in I\\(a_i,b_i)=(a,b)}} k_i}{\sum_{\substack{i\in I\\(a_i,b_i)=(a,b)}} k'_i}\right)\\
&\stackrel{(1)}{\leq} \sum_{(a,b)\in A\times B} \sum_{\substack{i\in I\\(a_i,b_i)=(a,b)}} k'_i\, f\left(\frac{k_i}{k'_i}\right) = \sum_{i\in I} k'_i\, f\left(\frac{k_i}{k'_i}\right) \leq \delta.
\end{aligned}$$

In the reasoning above, inequality (1) holds in view of Proposition 5.13. ■

Proof of Theorem 5.10. The proof follows immediately from the definition of judgement validity and an application of Lemma 5.5, taking as equivalence relation the equality over program memories. ■

Proof of Lemma 5.11. Assume that (f_1, f_2, γ) strongly-composes into f_3 and let $\mu_1, \mu_2 \in \mathcal{D}(\mathcal{M})$ be a pair of distributions of initial memories such that $\Delta^{\blacklozenge}_{f_1}(\mu_1,\mu_2) \leq \delta_1$. Then, by Lemma 5.5, we have

$$\mu_1\, \mathcal{L}^{\blacklozenge}_{f_1,\delta_1}(\equiv)\, \mu_2.$$

On the other hand, the validity of judgment $c \sim_{f_2,\delta_2} c : \equiv \Rightarrow \equiv$ states that for every pair of memories $m_1, m_2 \in \mathcal{M}$,

$$m_1 \equiv m_2 \implies (\llbracket c \rrbracket\, m_1)\, \mathcal{L}^{\blacklozenge}_{f_2,\delta_2}(\equiv)\, (\llbracket c \rrbracket\, m_2).$$

A direct application of Lemma 5.8 yields

$$(\mathsf{bind}\ \mu_1\ [\![c_1]\!])\ \mathcal{L}^{\blacklozenge}_{f_2,\delta_2}(\equiv)\ (\mathsf{bind}\ \mu_1\ [\![c_2]\!]),$$

which, on account of Lemma 5.5, gives the desired (f_3, δ_3)-closeness condition $\Delta^{\blacklozenge}_{f_3}\big([\![c]\!]^{\#}\,\mu_1, [\![c]\!]^{\#}\,\mu_2\big) \leq \delta_3$. ■

Proof of Lemma 5.12. The proof is straightforward from the definition of judgement validity and Proposition 5.6. ■

6
Related Work and Conclusions

6.1 Related Work

Formal Verification of Hash Functions and Elliptic Curve Cryptography. Despite their widespread use, hash functions and elliptic curves have received little attention from the formal verification community. To our best knowledge, our work from Chapter 3 provides the first machine-checked proof of security for a cryptographic primitive based on elliptic curves, and the first proof of security for a cryptographic hash function.

Previous works on the formalization of elliptic curves include [Hurd *et al.*, 2006] and [Théry & Hanrot, 2007]. Hurd *et al.* report on the verification in HOL of the group axioms and an application to the functional correctness of ElGamal encryption. Théry & Hanrot use Coq to formalize the group axioms, and show how the formalization of elliptic curves can be used to build efficient reflective tactics for testing primality.

Other works on the formalization of hash functions include [Backes *et al.*, 2012] and [Toma & Borrione, 2005]. Toma & Borrione use ACL2 to reason about functional properties of SHA-1, while Backes *et al.* use EasyCrypt [Barthe *et al.*, 2011b] to prove that, under certain assumptions, the Merkle-Damgård construction is collision-resistant and indifferentiable from a random oracle.

Recently, Bacelar Almeida *et al.* [2012] has reported on a certified compiler of Zero-Knowledge protocols. The verification is done in CertiCrypt and relies on the fragment of α-pRHL used to reason about statistical distance.

Differential Privacy. There is a vast body of work on differential privacy; we refer, e.g., to [Dwork, 2008, 2011] for an account of some of the latest developments in the field. In particular, there have been several proposals of methods for reasoning about differential privacy on the basis of different languages and models of computation; these methods are based on runtime verification, such as Pinq [McSherry, 2009] or Airavat [Roy *et al.*, 2010], type systems [Gaboardi *et al.*, 2013; Reed & Pierce, 2010], or deductive verification [Chaudhuri *et al.*, 2011]. We briefly overview each of them and refer the reader

to [Pierce, 2012] for a survey thereof.

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 & 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. In [Gaboardi *et al.*, 2013] the type system is extended with dependent types to reason about computations whose sensitivity depends on run-time information. As in Pinq, the soundness of the type systems 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 systems can handle functional data structures, they do 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.

In contrast to the above techniques, our framework supports reasoning about differential privacy guarantees from first principles. In particular, our extension of CertiCrypt 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 α-pRHL in general cannot be fully automated.

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 α-pRHL, and their unwinding-based verification method can be regarded as an abstract, language-independent, equivalent of α-pRHL. However, their method, like the rest of the reviewed techniques, is currently limited to reason about standard differential privacy; our logic α-pRHL enables reasoning about both standard and approximate differential privacy.

Distance between Probabilistic Computations. The problem of computing the distance between two probabilistic computations has been addressed in different areas of computer science, including machine learning, stochastic systems, and security.

Methods for computing the distance between probabilistic automata (PA) have been

studied by Cortes *et al.* [2007, 2008]. Their work is motivated by machine learning applications and shows that it is possible to compute certain distance measures between PA by solving a generalized version of the single-source shortest-path problem over weighted graphs. This reduction is proved valid for the Kullback-Leibler divergence, the Hellinger distance and the L_p distances, for even values of p; algorithms for computing exact values and approximations of these distance measures are provided.

Quantitative versions of approximate (bi)simulation relations can also be used to endow PA with a notion of distance. This approach has been investigated, among others, by Segala & Turrini [2007] and by Tracol *et al.* [2011]. As our work, their (bi)simulation relations rest on a notion of approximate relation lifting. The simulation of Segala & Turrini is carefully tailored to reason about cryptographic protocols and satisfies transitivity and good compositionality properties. The (bi)simulation of Tracol *et al.* is less sophisticated and admits a logical characterization. The liftings underlying both (bi)simulation relations can be obtained as slight variants of our (α, δ)-lifting for the case $\alpha = 1$.

Approximate probabilistic bisimulation has also been studied by Pierro *et al.* [2005] in the context of quantitative information flow. In their work, probabilistic transition systems are interpreted as bounded linear maps between Hilbert spaces and approximate bisimulation is defined in terms of the norm of such maps; an statistical interpretation of their notion of approximate bisimulation is also discussed. Recently, the connections between quantitative information flow and differential privacy have been explored by Barthe & Köpf [2011] and by Alvim *et al.* [2011].

The sequential composition operator of our probabilistic language expands errors in a "controlled" manner (see f-pRHL rule [s-seq] in Figure 5.3). This behaviour of language operators is connected with the notion of *non-expansiveness* from [Tini, 2010]. In his work, Tini studies conditions that guarantee good compositionality properties of approximate bisimulations w.r.t. the operators of probabilistic process algebras. These conditions are given as syntactical constrains on the set of admissible rules for defining the semantics of the operators.

Relational Program Verification. Program logics have a long tradition and have been used effectively to reason about functional correctness of programs. In contrast, the quantitative security notions that we consider in this dissertation are usually defined as 2-safety properties [Clarkson & Schneider, 2010; Terauchi & Aiken, 2005], that is, (universally quantified) properties 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 [Barthe *et al.*, 2011a; Zaks & Pnueli, 2008] 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 programs.

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 & Banerjee [2004] and Amtoft *et al.* [2006] develop specialized relational logics for information flow. Backes *et al.* [2009] compute relational weakest preconditions as a basis for quantifying information leaks. Further applications of relational program verification include determinism [Burnim & Sen, 2009] and robustness [Chaudhuri *et al.*, 2011].

Barthe *et al.* [2009b] extend Benton's logic to a probabilistic setting (pRHL) and use this logic for verifying game-based cryptographic proofs in the interactive proof-assistant Coq (see Chapter 2). In order to increase the automation level of these proofs, Barthe *et al.* [2011b] develop EasyCrypt, a tool that verifies automatically pRHL judgments using SMT solvers and a verification condition generator. In a follow-up work, Barthe *et al.* [2013a] extend EasyCrypt with support for reasoning about α-pRHL judgments and probabilistic operators. This extension is used to build automated proofs of differential privacy for some of the case studies reported in Chapter 4 and interactive game-based proofs of computational differential privacy for 2-party computations [Mironov *et al.*, 2009].

Probabilistic Program Verification. Reif [1980], Kozen [1985] and Feldman & Harel [1984] were among the first to develop axiomatic logics for probabilistic computations. This line of work was further developed by Jones [1993], Morgan *et al.* [1996], den Hartog [1999], and more recently by 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. It should be possible to develop relational variants of these logics or to use self-composition for reasoning about (quantitative) 2-safety properties.

Hurd was among the first to develop a machine-checked framework to reason about probabilistic programs [Hurd, 2003; Hurd *et al.*, 2005]. His formalization is based on the standard notion of σ-algebra, and partly follows earlier formalizations in Mizar. Building on Hurd's work, Coble [2010] and Mhamdi *et al.* [2010, 2011] formalized integration theory in the HOL proof assistant. 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 & 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 also been used to reason about the security of watermarking algorithms.

6.2 Conclusion and Future Work

From a technological perspective, the last few years have been marked by a steady increase in the sensitivity of the tasks that we entrust to computer systems. As a result, providing trustworthiness guarantees for these systems has become a primary concern. Designed

to ensure the highest level of assurance, the *verified security* methodology has proved an effective approach in this regard. However, it has been limited to the verification of a reduced number of security properties in the area of cryptography. In this dissertation, we have pushed the methodology's boundaries to allow for the formal, computer-aided analysis of an important class of quantitative security properties that remained out of its scope.

More specifically, we have focused on quantitative 2-safety properties, which require reasoning simultaneously about two different runs of a system and establishing a (quantitative) similarity condition between them. In order to verify this kind of properties, we have adopted a deductive approach: we model systems as probabilistic imperative programs and we follow Benton's approach of using relational Hoare logics to reason about 2-safety properties; we use the Coq proof assistant to discharge the generated proof obligations.

Our main technical contribution is f-pRHL, a quantitative relational Hoare logic for reasoning about f-divergences between probabilistic programs. f-divergences are a well-studied family of distance measures between probability distributions and serve as our basis for comparing program executions; the comparison is specified as an upper bound for the distance between the outputs generated by the executions of two programs, modulo relational pre and post-condition.

This program logic provides a uniform approach for reasoning about an important class of quantitative security notions such as differential privacy, approximate probabilistic non-interference and indifferentiability. Nevertheless, in order to make the presentation more incremental, we have introduced different quantitative logics aimed at reasoning about specific program properties and then we have shown how they can be recovered from our full-fledged relation Hoare logic f-pRHL.

We have confirmed the effectiveness of our approach through several case studies that have been fully formalized in Coq, using an extension of CertiCrypt. These case studies concisely illustrate some remarkable features of our approach. The case study of Chapter 3 demonstrates that our framework admits security analyses that involve specialized and complex reasonings from specific fields and blends well with large mathematical libraries. The case studies of Chapter 4 show that our technique for reasoning about differential privacy improves on the scope of all previous techniques. These techniques are resticted to standard differential privacy and algorithms that achieve privacy by combining basic mechanisms thorough sequential or parallel composition; instead, our program logic f-pRHL (or its framgment α-pRHL) can handle both standard and approximate differential privacy as well as algorithms that achieve privacy in an ad-hoc manner.

The scope of our logic f-pRHL is not limited to the verification of security properties, only. It can be used to reason in general about any property that is stated as a closeness condition between the output generated by two executions of a program (or eventually two different programs) modulo relational pre-condition; formally this corresponds to the class of quantitative 2-safety properties captured by formula (5.3). It remains as future work to investigate further applications of the logic outside the realm of security. Due to the extensive use that they make of f-divergences, we beleive that machine learning and

information theory would be fruitful areas to explore.

Another application of the logic that would be beneficial to investigate is the continuity analysis of (probabilistic) programs. The major challenge here is to account for the situation where perturbations in a program input lead to divergent control-flows. A promising starting point would be the work of Chaudhuri *et al.* [2010, 2011], who have studied this problem for the case of deterministic programs.

Besides exploring further applications of our framework, we believe that there is still much room for its improvement. One remarkable property of our framework is its generality and flexibility to reason about relational properties from first principles. Unfortunately, this affects negatively its automation capabilities. We believe that important future research efforts should focus on this direction. A first step has already been taken by Barthe *et al.* [2013a, 2011b] who combine the use of SMT solvers and a verification condition generator to somewhat automate the reasoning about pRHL and α-pRHL judgments; the task of generating invariants for handling loops is, however, fully delegated to the user. There is a vast body of work on invariant inference; it would be interesting to study how we can transpose this work to our (probabilistic relational) setting.

Another limitation of our framework is its confinement to the analysis of programs that sample values from discrete domains. This is because CertiCrypt, the base of our formalization, builds on the representation of distributions provided by the ALEA library [Audebaud & Paulin-Mohring, 2009], which cannot model continuous distributions. The applicability of our framework could be significantly broadened if we extend this library to continuous distributions.

Bibliography

Ali, S. M. & Silvey, S. D. (1966). A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)* **28**(1), 131–142.

Alvim, M., S., Andres, M., E., Chatzikokolakis, K. & Palamidessi, C. (2011). On the relation between Differential Privacy and Quantitative Information Flow. In: *38th International Colloquium on Automata, Languages and Programming - ICALP 2011*, vol. 6756 of *Lecture Notes in Computer Science*. Springer.

Amtoft, T., Bandhakavi, S. & 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*. New York: ACM.

Amtoft, T. & Banerjee, A. (2004). Information flow analysis in logical form. In: *SAS*.

Audebaud, P. & Paulin-Mohring, C. (2009). Proofs of randomized algorithms in Coq. *Sci. Comput. Program.* **74**(8), 568–589.

Bacelar Almeida, J., Barbosa, M., Bangerter, E., Barthe, G., Krenn, S. & Zanella Béguelin, S. (2012). Full proof cryptography: verifiable compilation of efficient zero-knowledge protocols. In: *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12. ACM.

Backes, M., Barthe, G., Berg, M., Grégoire, B., Kunz, C., Skoruppa, M. & Zanella Béguelin, S. (2012). Verified security of Merkle-Damgård. In: *25th IEEE Computer Security Foundations Symposium, CSF 2012*. IEEE Computer Society.

Backes, M., Köpf, B. & Rybalchenko, A. (2009). Automatic discovery and quantification of information leaks. In: *30th IEEE Symposium on Security and Privacy, S&P 2009*. IEEE Computer Society.

Barthe, G., Crespo, J. M. & Kunz, C. (2011a). Relational verification using product programs. In: *Proceedings of the 17th international conference on Formal methods*, FM'11. Berlin, Heidelberg: Springer-Verlag.

Barthe, G., Danezis, G., Grégoire, B., Kunz, C. & Béguelin, S. Z. (2013a). Verified computational differential privacy with applications to smart metering. In: *26th IEEE Computer Security Foundations Symposium, CSF 2013*. IEEE Computer Society.

Barthe, G., D'Argenio, P. & Rezk, T. (2004). Secure information flow by self-composition. In: *17th IEEE Workshop on Computer Security Foundations, CSFW 2004*. Washington: IEEE Computer Society.

Barthe, G., Grégoire, B., Heraud, S. & Zanella Béguelin, S. (2009a). Formal certification of ElGamal encryption. A gentle introduction to CertiCrypt. In: *5th International Workshop on Formal Aspects in Security and Trust, FAST 2008*, vol. 5491 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Barthe, G., Grégoire, B., Heraud, S. & Zanella-Béguelin, S. (2011b). Computer-aided security proofs for the working cryptographer. In: *Advances in Cryptology – CRYPTO 2011*, vol. 6841 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Barthe, G., Grégoire, B., Lakhnech, Y. & Zanella Béguelin, S. (2011c). Beyond provable security. Verifiable IND-CCA security of OAEP. In: *Topics in Cryptology – CT-RSA 2011*, vol. 6558 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Barthe, G., Grégoire, B. & Zanella-Béguelin, S. (2009b). Formal certification of code-based cryptographic proofs. In: *36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009*. New York: ACM.

Barthe, G., Grégoire, B. & Zanella Béguelin, S. (2010). Programming language techniques for cryptographic proofs. In: *1st International Conference on Interactive Theorem Proving, ITP 2010*, vol. 6172 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Barthe, G. & Köpf, B. (2011). Information-theoretic bounds for differentially private mechanisms. In: *24rd IEEE Computer Security Foundations Symposium, CSF 2011*. Los Alamitos: IEEE Computer Society.

Barthe, G., Köpf, B., Olmedo, F. & Zanella-Béguelin, S. (2012). Probabilistic relational reasoning for differential privacy. In: *39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012*. New York: ACM.

Barthe, G., Köpf, B., Olmedo, F. & Zanella-Béguelin, S. (2013b). Probabilistic relational reasoning for differential privacy. *ACM Transactions on Programming Languages and Systems* **35**(3).

Barthe, G., Olmedo, F. & Zanella Béguelin, S. (2011d). Verifiable security of Boneh-Franklin identity-based encryption. In: *5th International Conference on Provable Security, ProvSec 2011*, vol. 6980 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Bartzia, E.-I. (2011). *A Formalization of Ellptic Curves*. Master's thesis, Université de Vincennes-Saint Denis – Paris VIII.

Bellare, M. & Rogaway, P. (1993). Random oracles are practical: a paradigm for designing efficient protocols. In: *1st ACM Conference on Computer and Communications Security, CCS 1993*. New York: ACM.

Bellare, M. & Rogaway, P. (2006). The security of triple encryption and a framework for code-based game-playing proofs. In: *Advances in Cryptology – EUROCRYPT 2006*, vol. 4004 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

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*. New York: ACM.

Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H. & Tibouchi, M. (2010). Efficient indifferentiable hashing into ordinary elliptic curves. In: *Advances in Cryptology – CRYPTO 2010*, vol. 6223 of *Lecture Notes in Computer Science*. Springer.

Burnim, J. & Sen, K. (2009). Asserting and checking determinism for multithreaded programs. In: *ESEC/SIGSOFT FSE*.

Canetti, R., Goldreich, O. & Halevi, S. (2004). The random oracle methodology, revisited. *J. ACM* **51**(4), 557–594.

Chadha, R., Cruz-Filipe, L., Mateus, P. & Sernadas, A. (2007). Reasoning about probabilistic sequential programs. *Theoretical Computer Science* **379**(1-2), 142–165.

Chan, T.-H. H., Shi, E. & Song, D. (2010). Private and continual release of statistics. In: *37th International colloquium on Automata, Languages and Programming, ICALP 2010*, vol. 6199 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Chaudhuri, S., Gulwani, S. & Lublinerman, R. (2010). Continuity analysis of programs. In: *Proceedings of the 37th annual ACM SIGPLAN-SIGACT Symposium on Principles of programming languages*. New York, NY, USA: ACM.

Chaudhuri, S., Gulwani, S., Lublinerman, R. & Navidpour, S. (2011). Proving programs robust. In: *19th ACM SIGSOFT Symposium on the Foundations of Software Engineering and 13rd European Software Engineering Conference, ESEC/FSE 2011*. New York: ACM.

Clarkson, M. R. & 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*.

Coble, A. R. (2010). Anonymity, information, and machine-assisted proof. Tech. Rep. UCAM-CL-TR-785, University of Cambridge, Computer Laboratory. URL http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-785.pdf.

Cortes, C., Mohri, M. & Rastogi, A. (2007). L_p distance and equivalence of probabilistic automata. *Int. J. Found. Comput. Sci.* **18**(4), 761–779.

Cortes, C., Mohri, M., Rastogi, A. & Riley, M. (2008). On the computation of the relative entropy of probabilistic automata. *Int. J. Found. Comput. Sci.* **19**(1), 219–242.

Csiszár, I. (1963). Eine informationstheoretische ungleichung und ihre anwendung auf den beweis der ergodizitat von markoffschen ketten. *Publications of the Mathematical Institute of the Hungarian Academy of Science* **8**, 85–108.

den Hartog, J. (1999). Verifying probabilistic programs using a hoare like logic. In: *ASIAN*.

Deng, Y. & Du, W. (2011). Logical, metric, and algorithmic characterisations of probabilistic bisimulation. Tech. Rep. CMU-CS-11-110, Carnegie Mellon University.

Deng, Y., Glabbeek, R., Hennessy, M. & Morgan, C. (2009). Testing finitary probabilistic processes. In: *CONCUR 2009 - Concurrency Theory*, vol. 5710 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 274–288.

Desharnais, J., Laviolette, F. & 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.

Dwork, C. (2006). Differential privacy. In: *33rd International Colloquium on Automata, Languages and Programming, ICALP 2006*, vol. 4052 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Dwork, C. (2008). Differential privacy: A survey of results. In: *Theory and Applications of Models of Computation*, vol. 4978 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

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

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I. & Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In: *Advances in Cryptology – EUROCRYPT 2006*, vol. 4004 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Dwork, C., McSherry, F., Nissim, K. & Smith, A. (2006b). Calibrating noise to sensitivity in private data analysis. In: *3rd Theory of Cryptography Conference, TCC 2006*, vol. 3876 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Dwork, C., Rothblum, G. N. & Vadhan, S. P. (2010). Boosting and differential privacy. In: *Symposium on Foundations of Computer Science – FOCS 2010*. IEEE.

Ebanks, B., Sahoo, P. & Sander, W. (1998). *Characterizations of Information Measures*. World Scientific.

Feldman, Y. A. & Harel, D. (1984). A probabilistic dynamic logic. *J. Comput. Syst. Sci.* **28**(2), 193–215.

Fleischmann, E., Gorski, M. & Lucks, S. (2010). Some observations on indifferentiability. In: *Information Security and Privacy*, vol. 6168 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Fouque, P.-A. & Tibouchi, M. (2010). Deterministic encoding and hashing to odd hyperelliptic curves. In: *4th International Conference on Pairing-Based Cryptography, Pairing 2010*, vol. 6487 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Gaboardi, M., Haeberlen, A., Hsu, J., Narayan, A. & Pierce, B. C. (2013). Linear dependent types for differential privacy. In: *40th ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages, POPL 2013*. New York: ACM.

Galindo, D. (2005). Boneh-Franklin identity based encryption revisited. In: *32nd International Colloquium on Automata, Languages and Programming, ICALP 2005,*, vol. 3580 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Goguen, J. A. & Meseguer, J. (1982). Security policies and security models. In: *IEEE Symposium on Security and Privacy*.

Goldreich, O. (2002). Zero-knowledge twenty years after its invention. Tech. Rep. TR02-063, Electronic Colloquium on Computational Complexity.

Goldwasser, S. & Micali, S. (1984). Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299.

Gonthier, G., Mahboubi, A., Rideau, L., Tassi, E. & Théry, L. (2007). A modular formalisation of finite group theory. In: *20th International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2007*, vol. 4732 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Gupta, A., Ligett, K., McSherry, F., Roth, A. & Talwar, K. (2010). Differentially private combinatorial optimization. In: *21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*. SIAM.

Haeberlen, A., Pierce, B. C. & Narayan, A. (2011). Differential privacy under fire. In: *20th USENIX Security Symposium*. Berkeley: USENIX Association.

Hankerson, D., Vanstone, S. & Menezes, A. (2004). *Guide to Elliptic Curve Cryptography*. Springer Professional Computing. Springer.

Hartshorne, R. (1977). *Algebraic Geometry*. Graduate Texts in Mathematics. Springer-Verlag.

Hurd, J. (2003). Formal verification of probabilistic algorithms. Tech. Rep. UCAM-CL-TR-566, University of Cambridge, Computer Laboratory. URL http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-566.pdf.

Hurd, J., Gordon, M. & Fox, A. (2006). Formalized elliptic curve cryptography. In: *High Confidence Software and Systems, HCSS 2006*.

Hurd, J., McIver, A. & Morgan, C. (2005). Probabilistic guarded commands mechanized in HOL. *Theor. Comput. Sci.* **346**(1), 96–112.

Icart, T. (2009). How to hash into elliptic curves. In: *Advances in Cryptology – CRYPTO 2009*, vol. 5677 of *Lecture Notes in Computer Science*. Springer.

Icart, T. (2010). *Algorithms Mapping into Elliptic Curves and Applications*. Ph.D. thesis, Université du Luxembourg.

Jones, C. (1993). *Probabilistic Non-Determinism*. Ph.D. thesis, University of Edinburgh.

Jonsson, B., Yi, W. & Larsen, K. G. (2001). Probabilistic extensions of process algebras. In: *Handbook of Process Algebra* (Bergstra, J., Ponse, A. & Smolka, S., eds.). Amsterdam: Elsevier, pp. 685–710.

Kifer, D. & Lin, B.-R. (2010). Towards an axiomatization of statistical privacy and utility. In: *29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of database systems, PODS '10*. New York: ACM.

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

Maurer, U., Renner, R. & Holenstein, C. (2004). Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: *1st Theory of Cryptography Conference, TCC 2004*, vol. 2951 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

McSherry, F. & Talwar, K. (2007). Mechanism design via differential privacy. In: *48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007*. Washington: IEEE Computer Society.

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*. New York: ACM.

Mhamdi, T., Hasan, O. & Tahar, S. (2010). On the formalization of the lebesgue integration theory in hol. In: *ITP*.

Mhamdi, T., Hasan, O. & Tahar, S. (2011). Formalization of entropy measures in hol. In: *ITP*.

Mironov, I., Pandey, O., Reingold, O. & Vadhan, S. (2009). Computational differential privacy. In: *Advances in Cryptology – CRYPTO 2009*, vol. 5677 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Morgan, C., McIver, A. & Seidel, K. (1996). Probabilistic predicate transformers. *ACM Trans. Program. Lang. Syst.* **18**(3), 325–353.

Nikolov, A., Talwar, K. & Zhang, L. (2012). The geometry of differential privacy: the sparse and approximate cases. *CoRR* **abs/1212.0297**.

Nowak, D. (2009). On formal verification of arithmetic-based cryptographic primitives. In: *11th International Conference on Information Security and Cryptology, ICISC 2008*, vol. 5461 of *Lecture Notes in Computer Science*. Springer.

Pardo, M. & Vajda, I. (1997). About distances of discrete distributions satisfying the data processing theorem of information theory. *Information Theory, IEEE Transactions on* **43**(4), 1288–1293.

Pierce, B. C. (2012). Differential privacy in the programming languages community. Invited tutorial at DIMACS Workshop on Recent Work on Differential Privacy across Computer Science.

Pierro, A. D., Hankin, C. & Wiklicky, H. (2005). Measuring the confinement of probabilistic systems. *Theor. Comput. Sci.* **340**(1), 3–56.

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

Reed, J. & 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*. New York: ACM.

Reif, J. H. (1980). Logics for probabilistic programming (extended abstract). In: *STOC*.

Ristenpart, T., Shacham, H. & Shrimpton, T. (2011). Careful with composition: Limitations of the indifferentiability framework. In: *Advances in Cryptology – EUROCRYPT 2011*, vol. 6632 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Rockafellar, R. (1997). *Convex Analysis*. Princeton mathematical series. Princeton University Press.

Roy, I., Setty, S. T. V., Kilzer, A., Shmatikov, V. & Witchel, E. (2010). Airavat: security and privacy for MapReduce. In: *7th USENIX Conference on Networked Systems Design and Implementation, NSDI 2010*. Berkeley: USENIX Association.

Sahai, A. & Vadhan, S. (1999). Manipulating statistical difference. In: *Randomization Methods in Algorithm Design, DIMACS Workshop, 1997*, vol. 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society.

Segala, R. & Turrini, A. (2007). Approximated computationally bounded simulation relations for probabilistic automata. In: *20th IEEE Computer Security Foundations Symposium, CSF 2007*. IEEE Computer Society.

Shallue, A. & van de Woestijne, C. (2006). Construction of rational points on elliptic curves over finite fields. In: *7th International Symposium on Algorithmic Number Theory, ANTS-VII*, vol. 4076 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Shoup, V. (2001). OAEP reconsidered. In: *Advances in Cryptology – CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Shoup, V. (2004). Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332.

Shoup, V. (2009). *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, second ed.

Silverman, J. H. (2009). *The Arithmetic of Elliptic Curves*, vol. 106 of *Graduate Texts in Mathematics*. Heidelberg: Springer, 2nd ed.

Steinberger, J. (2012). Improved security bounds for key-alternating ciphers via hellinger distance. Cryptology ePrint Archive, Report 2012/481. http://eprint.iacr.org/.

Terauchi, T. & Aiken, A. (2005). Secure information flow as a safety problem. In: *12th International Symposium on Static Analysis, SAS 2005*, vol. 3672 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

The Coq development team (2010). The Coq Proof Assistant Reference Manual Version 8.3. Online – http://coq.inria.fr.

Théry, L. & Hanrot, G. (2007). Primality proving with elliptic curves. In: *20th International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2007*, vol. 4732 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Tini, S. (2010). Non-expansive ϵ-bisimulations for probabilistic processes. *Theoretical Computer Science* **411**(22-24), 2202–2222.

Toma, D. & Borrione, D. (2005). Formal verification of a SHA-1 circuit core using ACL2. In: *18th International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2005*, vol. 3603 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Tracol, M., Desharnais, J. & Zhioua, A. (2011). Computing distances between probabilistic automata. In: *Proceedings of QAPL*, vol. 57 of *EPTCS*.

Tschantz, M. C., Kaynar, D. & Datta, A. (2011). Formal verification of differential privacy for interactive systems. *Electronic Notes in Theoretical Computer Science* **276**, 61–79.

van Tiel, J. (1984). *Convex analysis: an introductory text*. Wiley.

Zaks, A. & Pnueli, A. (2008). Covac: Compiler validation by program analysis of the cross-product. In: *FM*.

Zanella Béguelin, S. (2010). *Formal Certification of Game-Based Cryptographic Proofs*. Ph.D. thesis, Ecole Nationale Supérieure des Mines de Paris – Mines ParisTech.

Zanella Béguelin, S., Grégoire, B., Barthe, G. & Olmedo, F. (2009). Formally certifying the security of digital signature schemes. In: *30th IEEE Symposium on Security and Privacy, S&P 2009*. Los Alamitos: IEEE Computer Society.