# Proving that Programs are Differentially Private

Annabelle McIver[1] and Carroll Morgan[2] *

[1] Dept. Computing, Macquarie University
[2] University of New South Wales and Data61

**Abstract.** We extend recent work in Quantitative Information Flow (QIF) to provide tools for the analysis of programs that aim to implement differentially private mechanisms. We demonstrate how differential privacy can be expressed using loss functions, and how to use this idea in conjunction with a QIF-enabled program semantics to verify differentially private guarantees. Finally we describe how to use this approach experimentally using Kuifje, a recently developed tool for analysing information-flow properties of programs.

**Keywords**: Quantitative Information Flow, probabilistic program semantics, verification, privacy, differential privacy.

## 1 Introduction

This paper concerns the verification of privacy properties of programs. Our particular focus is *differential privacy* and how to prove that implementations of privacy-style mechanisms satisfy a differentially private property. Whilst differential privacy has been much studied as a mathematical theory there is further work to be done in ensuring that its properties are faithfully implemented in a programming language. This is particularly important because implementations are rarely transcriptions of mathematical functions, and program code could present very differently from the mathematical function it purports to compute. There are several important approaches for tackling this problem [**?**,**?**] from a programming languages perspective. In this paper we study it as a verification exercise using a programming semantics based on the *Quantitative Information Flow* (QIF) paradigm.

Quantitative information flow is designed to model the severity of risks associated with any information leaks in systems that process confidential information. QIF consists of three components. The first is a model for confidential information (or "secrets") based on probability distributions over possible values that a secret could take. The second is a model of a mechanism (or program) which describes how the mechanism's external outputs affect the flow of information about the secret. The third is a model of an adversary (as a gain or loss function)

* This research was supported by the Australian Research Council Grant DP140101119.

operating within a specific scenario describing how an adversary could exploit any flow that has occurred. A QIF analysis takes these elements and then interprets the severity of the information leak relative to the given adversarial scenario.

Elsewhere [**?**] it has been shown how to model differential privacy as a QIF mechanism and here we extend that idea by introducing an adverarial scenario (or loss function) that can be used to verify whether a mechanism satisfies a differential privacy guarantee. We also show how it can be applied directly to a QIF-enabled semantics of a sequential programming language [**?**,**?**] to verify differential privacy. Finally we describe how a QIF-enabled interpreter Kuifje [**?**] is able to verify experimentally that sequential programs satisfy differentially private properties.

In §**??** we review the fundamentals of QIF, and in §**??** we show how to express a the notion of differential privacy as an adversarial scenario. In §**??** we recall how to use the basic QIF ideas in a QIF-enabled program semantics and illustrate how to use it to verify a small example based on the familiar random-response protocol of Warner [**?**].

## 2 Review of Quantitative Information Flow

The informal idea of a secret is that it is something about which there is some uncertainty, and the greater the uncertainty the more difficult it is to discover exactly what the secret is. For example, the name of one's first primary school teacher might not be generally known, but if the gender of the teacher is leaked, then it might rule out some possible names and make others more likely. Similarly, when some information about a secret becomes available to an observer (often referred to as an adversary) the uncertainty is reduced, and it becomes easier to guess its value. If that happens, we say that information (about the secret) has leaked, or equivalently that information flow has occurred.

Quantitative Information Flow (QIF) makes this intuition mathematically precise. Given a range of possible secret values of (finite) type $\mathcal{X}$, let $\mathbb{D}\mathcal{X}$ be the space of probability distributions over $\mathcal{X}$. We model a secret as a probability distribution of type $\mathbb{D}\mathcal{X}$, because it ascribes "probabilistic uncertainty" to the secret's exact value. Given $\pi\colon \mathbb{D}\mathcal{X}$ we write $\pi_x$ for the probability that $\pi$ assigns to $x\colon \mathcal{X}$ with the idea that the more likely it is that the real value is some specific $x$ then the closer $\pi_x$ will be to 1. Normally the uniform distribution over $\mathcal{X}$ models a secret which with equal likelihood could take any one of the possible values drawn from its type and we might say that, beyond the existence of the secret, nothing else is known. There could, of course, be many reasons for using some other distribution, for example if the secret was the height of an individual then a normal distribution might be more realistic. In any case, once we have a secret, we are interested in analysing whether an algorithm, or protocol, that uses it might leak some information about it. To do this we define a measure for uncertainty, and use it to compare the uncertainty of the secret before and after

executing the algorithm. If we find that the two measurements are different then we can say that there has been an information leak.

The original QIF analyses of information leaks in computer systems used Shannon entropy [?] to measure uncertainty because it captures the idea that more uncertainty implies "more secrecy", and indeed the uniform distribution corresponds to maximum Shannon entropy (corresponding to maximum "Shannon uncertainty"). More recent treatments have shown however that Shannon entropy is not the best way to measure uncertainty in security contexts because it does not necessarily model scenarios relevant to the goals of the adversary. In particular there are some circumstances where a Shannon analysis gives a more favourable assessment of security than is actually warranted if the adversary's motivation is taken into account [?].

Alvim et al. [?] proposed a more general notion of uncertainty based on "gain functions". Here we shall use its dual formulation namely *loss functions*. A *loss function* measures a secret's uncertainty according to how it affects an adversary's actions within a given scenario. We write $\mathcal{W}$ for a (usually finite) set of actions available to an adversary corresponding to an "attack scenario" where the adversary tries to guess something (e.g. some property) about the secret. For a given secret $x\colon \mathcal{X}$ an adversary's choice of $w\colon \mathcal{W}$ results in the adversary losing something beneficial to his objective [3]. This loss can vary depending on the adversary's choice ($w$) and the exact value of the secret ($x$). The more effective is the adversary's choice in how to act, the more he is able to overcome any uncertainty concerning the secret's value thereby losing less compared to his losses in the same scenario but without the benefit of the leaked information.

**Definition 1.** *Given a type $\mathcal{X}$ of secrets, a* loss function $\ell\colon \mathcal{W}\times\mathcal{X} \to \mathbb{R}$ *is a real-valued function such that $\ell(w, x)$ determines the loss to an adversary if he chooses $w$ and the secret is $x$.*

A simple example of a loss function is given by $\mathsf{br}$, where $\mathcal{W} := \mathcal{X}$, and

$$\mathsf{br}(x, x') \quad := \quad 0 \;\; \textit{if} \;\; x = x' \;\; \textit{else} \;\; 1 \; . \tag{1}$$

For this scenario, the cost to the adversary if he correctly guesses the value of a secret is 0, but 1 if he guesses incorrectly. Elsewhere the utility and expressivity of loss functions for measuring various attack scenarios relevant to security have been explored in more detail [?,?]. Given a loss function we define the *uncertainty* of a secret in $\mathbb{D}\mathcal{X}$ relative to the scenario it describes: it is the minimum average loss to an adversary.

**Definition 2.** *Let $\ell\colon \mathcal{W}\times\mathcal{X} \to \mathbb{R}$ be a loss function, and $\pi\colon \mathbb{D}\mathcal{X}$ be a secret. The* uncertainty $U_\ell[\pi]$ *of the secret wrt. $\ell$ is:*

$$U_\ell[\pi] \quad := \quad \min_{w\in\mathcal{W}} \sum_{x\in\mathcal{X}} \ell(w, x)\times\pi_x \; .$$

[3] Alvim et al. explained this as a gain to benefit the adversary; couching the interpretation as losses is mathematically equivalent but the formulation as losses turns out to be more convenient for reasoning about programs [?].

For a secret $\pi: \mathbb{D}\mathcal{X}$, the uncertainty wrt. $\mathsf{br}$ in particular is $U_{\mathsf{br}}[\pi] := 1 - \max_{x:\mathcal{X}} \pi_x$, i.e. the complement of the maximum probability assigned by $\pi$ to possible values of $x$. The adversary's best strategy for minimising his loss would therefore be to choose the value $x$ that corresponds to the maximum probability under $\pi$.

A *mechanism* is an abstract model of a protocol or algorithm that uses secrets. As the mechanism executes we assume that there are a number of observables that can depend on the actual value of the secret. We define $\mathcal{Y}$ to be the type for observables. The model of a mechanism now assigns a probability that $y: \mathcal{Y}$ can be observed given that the secret is $x$. Such observables could be sample timings in a timing analysis in cryptography, for example.

**Definition 3.** *A mechanism is a* stochastic matrix[4] $C: \mathcal{X} \times \mathcal{Y} \to [0,1]$. *The value $C_{xy}$ is the probability that $y$ is observed given that the secret is $x$. Given a (prior) secret $\pi: \mathbb{D}\mathcal{X}$ we write $\pi \rangle C$ for the* joint distribution *over $\mathcal{X} \times \mathcal{Y}$ defined*

$$(\pi \rangle C)_{xy} \quad := \quad \pi_x \times C_{xy} \ .$$

*For each $y: \mathcal{Y}$, the* marginal probability *that $y$ is observed is $p_y := \sum_{x:\mathcal{X}} (\pi \rangle C)_{xy}$. And for each observable $y$ the corresponding* posterior probability *of the secret is the conditional $\pi|^y: \mathbb{D}\mathcal{X}$ defined $(\pi|^y)_x := (\pi \rangle C)_{xy} / p_y$ .* [5] *(It is undefined if $p_y$ is zero.)*

Now consider the secret space with only two values, $\mathcal{X} := \{\mathtt{c}, \neg\mathtt{c}\}$ and the channel `inc` given below which produces two observations `A` and `B`. If the secret is `c` then `A` will be observed with probability $1/4$ and `B` will be observed with probability $3/4$. Alternatively if the secret is $\neg$`c` then `A` will be observed with probability $3/4$ and `B` will be observed with probability $1/4$.

$$\mathtt{inc} := \begin{array}{c} \\ \mathtt{c} \\ \neg\mathtt{c} \end{array} \begin{array}{c} \begin{array}{cc} \mathtt{A} & \mathtt{B} \end{array} \\ \begin{pmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{pmatrix} \end{array} \tag{2}$$

Intuitively, given a prior secret $\pi$, the entry $\pi_x \times C_{xy}$ of the joint distribution $\pi \rangle C$ is the probability that the actual secret value is $x$ and the observation is $y$. This joint distribution contains two pieces of information: the probability $p_y$ of observing $y$ and the corresponding posterior $\pi|^y$ which would then represent the adversary's updated view about the uncertainty of the secret's value. In our example above, if $\pi$ is the uniform distribution over $\{\mathtt{c}, \neg\mathtt{c}\}$ then $p_A = p_B = 1/2$. On the other hand the posterior $\pi|^A$ assigns a probability of $1/4$ that the secret is `c`, an event to which $\pi|^B$ assigns a probability of $3/4$. This implies that if `A` is observed the adversary is likely to decide that the secret is $\neg$`c`, whereas if he observes `B` then he will most likely determine that the secret is `c`. The extent to which the adversary can use the leaked information can be understood by comparing the uncertainties of the posteriors with the uncertainty of the prior.

[4] Stochastic means that the rows sum to 1.

[5] We use $p_y$ and $\pi|^y$ for typographical convenience. Notation suited for calculation would need to incorporate $C$ and $\pi$.

For example if $U_\ell[\pi|^A]$ is strictly less than $U_\ell[\pi]$ then the adversary is indeed able to use the information leaked by `inc` to benefit himself within the scenario defined by the loss function $\ell$.

More generally if the risk of the posterior decreases wrt. a scenario defined by $\ell$, then information about the secret has leaked and the adversary can use it to decrease his loss by changing how he chooses to act. The adversary's average overall loss, taking the observations into account, is defined to be the average posterior uncertainty (i.e. the posterior distribution, weighted according to their respective marginals):

$$U_\ell[\pi\rangle C] := \sum_{y\in\mathcal{Y}} p_y \times U_\ell[\pi|^y] \ , \qquad \textit{where } p_y,\ \pi|^y \textit{ are defined at Def. ??.} \tag{3}$$

Now that we have Def. **??** and Def. **??** we can start to investigate whether the information leaked through observations $\mathcal{Y}$ actually have an impact in terms of whether it is useful to an adversary. It is easy to see that for any loss function $\ell$, prior $\pi$ and mechanism $C$ we have that $U_\ell[\pi] \geq U_\ell[\pi\rangle C]$. In fact the greater the difference between the prior and posterior vulnerability, the more the adversary is able to use the leaked information within the scenario defined by $\ell$. In a mechanism that leaks no information at all, its prior and posterior vulnerabilities are the same under any scenario.

In our example above, can compare the overall losses without the benefit of `inc`'s leaks and with them as follows. Since $U_{\mathsf{br}}[\pi] = 1/2$ and $U_{\mathsf{br}}[\pi\rangle \mathtt{inc}] = 1-3/4 = 1/4$, we can see that with the benefit of `inc`'s leaked information the adversary is able to halve his losses (in the Bayes' Risk scenario).

Using the idea of quantifying losses, we can define a robust qualitative comparison between mechanisms. We say that one mechanism $M'$ is more secure than another $M$ exactly when the adversary's losses under $M'$ are always at least those under $M$ in every possible scenario defined by a prior and a loss function.

**Definition 4.** *Given mechanisms $M, M'$ we say that $M'$ is more secure than $M$, or $M \sqsubseteq M'$ if for all loss functions $\ell$ and priors $\pi$ we have that*

$$U_\ell[\pi\rangle M] \quad \leq \quad U_\ell[\pi\rangle M'] \ .$$

Given the observation above that $U_\ell[\pi] \geq U_\ell[\pi\rangle C]$ we can say (now formally) that the mechanism that releases no information is the most secure amongst all mechanisms.

In the next section we will review how to specialise the above ideas to obtain a QIF formulation of differential privacy which can then be applied directly to verify programs.

## 3 Differential privacy as a problem in QIF

Dwork's original definition of differential privacy [**?**] relates to databases and protections for individuals whose data might or might not be contained in the database. We generalise the definition for the context of secrets described above.

Given a (privacy) mechanism $\mathcal{M} : \mathcal{X} \to \mathbb{D}\mathcal{Y}$ and $\epsilon > 0$, we say that $\mathcal{M}$ is $\epsilon$-differentially private with respect to secrets $x, x'$ if:

$$\int_{\mathcal{M}.x} \zeta \quad \leq \quad e^{\epsilon} \times \int_{\mathcal{M}.x'} \zeta \ , \tag{4}$$

where $\zeta : \mathcal{Y} \to \mathbb{R}$ is any (measurable) function, and we write $\int_\gamma \zeta$ for the weighted average of $\zeta$ with respect to the distribution $\gamma \in \mathbb{D}\mathcal{Y}$. More general definitions of differential privacy [**?**] include a metric between secrets which are incorporated in constraints such as (**??**), so that:

$$\int_{\mathcal{M}.x} \zeta \quad \leq \quad e^{\epsilon \times d(x,x')} \times \int_{\mathcal{M}.x'} \zeta \tag{5}$$

must hold, where $d(\cdot,\cdot)$ is a metric on values. Here the metric provides a measure of similarity between alternative values, with a greater divergence between values implying a possible greater variation in the distributions over outputs. In our definitions below we use the simpler (**??**), but note here that they can easily be extended to include a metric as in (**??**). [6]

Alvim et al. [**?**] show that when a mechanism is modelled as a channel the above definition (**??**) is equivalent to comparing rows of channels relating to $x, x'$. In particular (**??**) applied to a channel $M$ says that $M$ is $\epsilon$-differentially private for $x, x'$ if

$$M_{xy} \leq e^{\epsilon} M_{x'y} \quad \textit{and} \quad M_{x'y} \leq e^{\epsilon} M_{xy} \tag{6}$$

for all $y \in \mathcal{Y}$. Notice that (**??**) compares two channel rows corresponding to secret values $x, x'$. It turns out that we can do the same thing using loss functions. Given a pair $v := (v_1, v_2) \in \mathcal{X} \times \mathcal{X}$ we write $\overleftarrow{v}$ for the left component $v_1$ and $\overrightarrow{v}$ for the right component $v_2$. We say that a subset of pairs $\mathcal{V} \subseteq \mathcal{X} \times \mathcal{X}$ is *symmetric* and *irreflexive* if whenever $(x, x') \in \mathcal{V}$ then also $(x', x) \in \mathcal{V}$ and $(x, x) \notin \mathcal{V}$ for any $x$.

**Definition 5.** *Given are $\epsilon > 0$, and $\mathcal{W} := \mathcal{V} \cup \{\star\}$, where $\mathcal{V} \subseteq \mathcal{X} \times \mathcal{X}$ is symmetric and irreflexive. We define $dp_\epsilon$, the $\epsilon$-differentially private loss function relative to $\mathcal{W}$:*

$$\begin{array}{lcll} dp_\epsilon(w,x) & = & -1 \ , & \textit{if} \ \ w \neq \star \quad \wedge \quad \overleftarrow{w} = x \\ dp_\epsilon(w,x) & = & e^{\epsilon} \ , & \textit{if} \ \ w \neq \star \quad \wedge \quad \overrightarrow{w} = x \\ dp_\epsilon(w,x) & = & 0 \ , & \textit{if} \ \ w \neq \star \quad \wedge \quad \overleftarrow{w} \neq x \neq \overrightarrow{w} \\ dp_\epsilon(\star,x) & = & 0 \ . & \end{array}$$

[6] The revised definition would change the second line of Def. **??** to be

$$dp_\epsilon(w,x) \quad = \quad e^{\epsilon \times d(\overleftarrow{w}, \overrightarrow{w})} \ , \quad \textit{if} \ \ w \neq \star \quad \wedge \quad \overrightarrow{w} = x \ .$$

Note that for $\pi \in \mathbb{D}\mathcal{X}$ we see that $U_{dp_\epsilon}[\pi] = \min_{x \neq x'}(\pi_x \times e^\epsilon - \pi_{x'}) \min 0$, where the minimisation is over the relevant pairs $(x, x')$ defined by $\mathcal{V}$. This means that if $U_{dp_\epsilon}[\pi] \geq 0$ then for each $x \neq x'$ we must have that $\pi_x \times e^\epsilon - \pi_{x'} \geq 0$, which is reminiscent of (**??**). It turns out that this idea can indeed be applied to privacy, as follows.

**Theorem 1.** *Given are $\epsilon > 0$, and $M$ a mechanism interpreted as a channel, and let $\upsilon$ be the uniform distribution in $\mathbb{D}\mathcal{X}$. Then $M$ satisfies $\epsilon$-differential privacy if and only if*

$$U_{dp_\epsilon}[\upsilon \rangle M] = 0 \ .$$

*Proof. It is clear that $U_{dp_\epsilon}[\upsilon \rangle M] \leq 0$, since the adversary is always able to choose action $\star$ for a loss of zero to the adversary. If it turns out that $U_{dp_\epsilon}[\upsilon \rangle M] < 0$, it implies by (**??**) that there is some observation $y$ such that the adversary can choose some $w \in \mathcal{V}$ that gives an average negative loss, i.e. that $U_{dp_\epsilon}[\upsilon|^y] < 0$ for some observation $y$. Let $(x, x') = w$ be the action that produces that negative minimum loss for this $y$. We reason as follows:*

$$\begin{array}{lll} & U_{dp_\epsilon}[\upsilon|^y] < 0 & \\ \Rightarrow & \sum_{x'' \in \mathcal{X}} dp_\epsilon(w, x'') \times M_{x''y}/|\mathcal{X}| < 0 & \text{“Def. ?? for } \upsilon|^y \text{ and choice of } w \text{ Def. ??”} \\ \Rightarrow & e^\epsilon M_{x'y}/|\mathcal{X}| - M_{xy}/|\mathcal{X}| < 0 \ , & \text{“Def. ??, with } w = (x, x')\text{”} \end{array}$$

*implying from (**??**) that $M$ is not differentially private.*

*On the other hand if $U_{dp_\epsilon}[\upsilon \rangle M] \geq 0$, it must mean that $e^\epsilon M_{x'y}|/\mathcal{X}| - M_{xy}/|\mathcal{X}| \geq 0$ for all choices of $x, x', y$ and so $M$ is $\epsilon$-differentially private.*

A simple corollary is that any mechanisms that are more secure than some $\epsilon$-differentially private mechanism $M$, must also be $\epsilon$-differentially private.

**Lemma 1.** *If $M, M'$ are two mechanisms and $M \sqsubseteq M'$, then if $M$ is $\epsilon$-differentially private, so is $M'$.*

*Proof. We reason as follows:*

$$\begin{array}{lll} & U_{dp_\epsilon}[\upsilon \rangle M'] & \\ \geq & U_{dp_\epsilon}[\upsilon \rangle M] & \text{“}M \sqsubseteq M'\text{, Def. ??”} \\ \geq & 0 \ , & \text{“Thm. ?? for } M\text{”} \end{array}$$

*implying that $M'$ is $\epsilon$-differentially private, also by Thm. **??**.*

Observe that when $M$ fails to be $\epsilon$-differentially private it is because the adversary is able to reason that the secret is more likely to be one value rather than another by an amount distinguishable by $e^\epsilon$.

Recall the mechanism `inc` from (**??**). Now from Def. **??** and Thm. **??** we see that `inc` is $\log 3$ differentially private [7] but not $\log 2$ differentially private, since

$$U_{dp_{\log 2}}[\upsilon \rangle \texttt{inc}] \quad < \quad 0 \quad \leq \quad U_{dp_{\log 3}}[\upsilon \rangle \texttt{inc}] \ .$$

[7] We use logs base $e$ throughout.

Next we prove the familiar additive law for differentially-private mechanisms. Recall that the additive law determines that if the secret is accessed first by $M$ and then by $M'$ then the combined access represents a mechanism with (differential privacy) parameter the sum of those of $M$ and $M'$. From (**??**) we can see that $U_\ell[\upsilon\rangle M]$ is determined by the posteriors, and Thm. **??** teaches us that whether or not a mechanism satisfies a differentially private property is therefore determined by the "unpredictability" of its posteriors, defined next.

**Definition 6.** *Let $\pi$ be a (prior/posterior) distribution in $\mathbb{D}\mathcal{X}$, and let $\epsilon{>}0$. We say that $\pi$ is $dp_\epsilon$-unpredictable if $U_{dp_\epsilon}[\pi] \geq 0$.*

The uniform distribution $\upsilon$ of the whole type is $dp_0$-unpredictable, and a consequence of Thm. **??** is that a mechanism is $\epsilon$-differentially private if and only if all posteriors in $u\rangle M$ are $dp_\epsilon$-unpredictable. In cases where the prior $\pi$ is known, and it is not uniform, we can see that unpredictability of the posteriors $\pi\rangle M$ are bounded by $\pi$'s unpredictability and the $\epsilon$-privacy of $M$.

**Lemma 2.** *Let $\pi \in \mathbb{D}\mathcal{X}$ be $dp_\epsilon$-unpredictable, and let $M$ be $\epsilon'$-differentially private. Then $U_{dp_{\epsilon+\epsilon'}}[\pi\rangle M] = 0$.*

*Proof. (Sketch.) We observe first that by assumption we have that for (relevant) $x, x' \in \mathcal{X}$ and any $y \in \mathcal{Y}$, we know that $\pi_x - e^\epsilon \pi_{x'} \geq 0$ and $M_{xy} - e^{\epsilon'} M_{x'y} \geq 0$. Rearranging, we have:*

$$\begin{array}{ll} & \pi_x \geq e^\epsilon \pi_{x'} \ \wedge \ M_{xy} \geq e^{\epsilon'} M_{x'y} \\ \Rightarrow & \pi_x \times M_{xy} \geq e^{\epsilon+\epsilon'} \pi_{x'} \times M_{x'y} \ , \qquad \textit{"arithmetic"} \end{array}$$

*from which we deduce that $\pi_x \times M_{xy} - e^{\epsilon+\epsilon'} \pi_{x'} \times M_{x'y} \geq 0$. This implies that $U_{dp_{\epsilon+\epsilon'}}[\pi|^y] = 0$ for the posterior $\pi|^y$. Hence by (**??**) we must have $U_{dp_{\epsilon+\epsilon'}}[\pi\rangle M] = 0$ as well.*

For the composition of two differentially private mechanisms, if the posteriors of the composition written $(\upsilon\rangle(M; M'))$ are formed from the posteriors of $(\upsilon|^y\rangle M')$, where $\upsilon|^y$ is any posterior of $(\upsilon\rangle M)$, then it follows that the unpredictability of all the posteriors $(\upsilon\rangle(M; M'))$ are determined by Lem. **??**. In fact we shall see in our semantics for programming language this is case (see §**??**), thus for such a composition we have the following additive law.

**Corollary 1.** *Let $M$ be $\epsilon$-differentially private, and $M'$ be $\epsilon'$-differentially private. The composition $M; M'$ is $\epsilon{+}\epsilon'$-differentially private.*

*Proof. (Sketch.) This follows if all posteriors of $(\upsilon\rangle(M; M'))$ are $dp_{\epsilon+\epsilon'}$-unpredictable. But each such posterior is exactly one of the posteriors of $\upsilon|^y\rangle M'$ for some posterior $\upsilon|^y$ of $(\upsilon\rangle M)$ (see discussion above). Since $M$ is $\epsilon$-differentially private we have that $\upsilon|^y$ must be $dp_\epsilon$-unpredictable; therefore by Lem. **??** it must be that all posteriors of $(\upsilon|^y\rangle M')$ are $dp_{\epsilon+\epsilon'}$-unpredictable since $M'$ is $\epsilon'$-differentially private.*

In the remainder of the paper we show how to apply these ideas to the verification of programs that implement differential privacy.

## 4 QIF in programming languages

Elsewhere [**?**] we introduced a probabilistic semantics applicable to a small sequential programming language. It embeds QIF ideas within a probabilistic semantics based on the well known probability monad [**?**].

### 4.1 The probabilistic Monad for information flow

Standard models of (sequential) probabilistic programs are normally based on Markov Processes with type $\mathcal{A} \to \mathbb{D}\mathcal{A}$. In this sense programs can be thought of as mapping a base type $\mathcal{A}$ to a probability distribution (also) over type $\mathcal{A}$. In QIF however, as has been noted, the mathematical essentials for understanding information flows are priors, posteriors and marginals. Setting $\mathcal{A}$ to $\mathbb{D}\mathcal{X}$ that gives the type of a QIF-enabled model for programs as $\mathbb{D}\mathcal{X} \to \mathbb{D}(\mathbb{D}\mathcal{X})$, or $\mathbb{D}\mathcal{X} \to \mathbb{D}^2\mathcal{X}$.

We call an object of type $\mathbb{D}^2\mathcal{X}$ a *hyper-distribution* over $\mathcal{X}$. It turns out that hyper-distributions exactly match the structure of posteriors and marginals discussed above. Recall the mechanism `inc` described at (**??**) and that the observations labelled `A` and `B` both occurred with probability $1/2$ (in the given scenario of a uniform prior) with corresponding posteriors $\pi|^A$ and $\pi|^B$. Formatted as a hyper-distribution, this scenario can be presented as:

$$\frac{1}{2}(\pi|^A) \quad \oplus \quad \frac{1}{2}(\pi|^B) \ , \tag{7}$$

where we use the operator $\oplus$ to indicate addition at the level of $\mathbb{D}\mathcal{X}$ considered as a "vector space", so that a hyper-distribution is a weighted $\oplus$-sum of posteriors considered as individual (1-summing) vectors.

In (**??**) the outer distribution corresponds to the marginal and the inner distributions corresponds to posteriors. Moreover for a hyper-distribution $\Delta \in \mathbb{D}^2\mathcal{X}$, we write $\Delta_\delta$ for the outer probability corresponding to inner $\delta$; we can therefore define the average uncertainty relative to $\Delta$ as:

$$U_\ell(\Delta) \quad := \quad \sum_\delta U_\ell[\delta] \times \Delta_\delta \ .$$

If we let $[\pi\rangle M]$ be formatted as a hyper-distribution as sketched above, we can see clearly that $U_\ell[\pi\rangle M]$ returns exactly the same value as (**??**) for $\pi\rangle M$ as a joint distribution, showing that the average posterior uncertainty does not depend on the names of the observations, but only on how a mechanism determines marginals and posteriors [**?**,**?**]. With this in mind we can define a QIF-enabled semantic space.

**Definition 7 ([?,?]).** *Let $\mathcal{X}$ be a (finite) state space. The space of programs is defined to be the set of functions from priors to hyper-distributions $\mathbb{D}\mathcal{X} \to \mathbb{D}^2\mathcal{X}$. If $P, P' : \mathbb{D}\mathcal{X} \to \mathbb{D}^2\mathcal{X}$ are programs then we say that $P \sqsubseteq P'$ if $U_\ell(P.\pi) \leq U_\ell(P'.\pi)$ for all loss functions $\ell$ and priors $\pi$.*

Once a program is modelled as a function $\mathbb{D}\mathcal{X} \to \mathbb{D}^2\mathcal{X}$, it turns out that a standard Giry Monadic setting provides the basic functionality for sequencing and assignments. We summarise the semantics for three important operators here, and refer elsewhere for full details [**?**]. Recall the Giry Monad defined by the triple $(\mathbb{D}, \eta, \mathsf{avg})$, where the type constructor $\mathbb{D}$ is a functor, $\eta$ maps an object of type $\mathcal{A}$ to a point distribution in $\mathbb{D}\mathcal{A}$ and $\mathsf{avg} : \mathbb{D}^2\mathcal{A} \to \mathbb{D}\mathcal{A}$ takes the weighted average of a hyper-distribution, defined

$$(\mathsf{avg}.\Delta)_a \quad := \quad \sum_{\delta \in \mathbb{D}\mathcal{X}} \Delta_\delta \times \delta_a \ .$$

Here we use $+$ and $\sum$ to mean the normal summation between numbers.

We can interpret a programming language in terms of Def. **??** as follows, where we use $[\![\cdot]\!]$ to map a program fragment to a function $\mathbb{D}\mathcal{X} \to \mathbb{D}^2\mathcal{X}$.

1. **Assignment.** Let $f : \mathcal{X} \to \mathbb{D}\mathcal{X}$ be a function that maps states in $\mathcal{X}$ to distributions over states in $\mathcal{X}$. [8]
$$[\![\mathtt{x}{:=}\,\mathtt{f(x)}]\!].\pi \quad := \quad (\eta \circ \mathsf{avg} \circ \mathbb{D}f).\pi \ .$$

2. **Sequence.** Let $P, Q$ be program fragments.
$$[\![P; Q]\!].\pi \quad := \quad (\mathsf{avg} \circ \mathbb{D}[\![Q]\!] \circ [\![P]\!]).\pi \ .$$

3. **Print statement.** Let $g$ be a function from $\mathcal{X}$ to $\mathcal{Y}$.
$$[\![\mathtt{Print}\ g]\!].\pi \quad := \quad \bigoplus_y p_y(\pi|^y) \ ,$$
where $p_y{:=}\,((\mathbb{D}g).\pi)_y$, and $\pi|^y$ is the posterior probability distribution, given that $y$ is an output of $g$, and $\oplus$ is the summation over 1-summing vectors described above.

The assignment statement is used to assign a value to a variable $x$ according to a distribution, where informally we assume that the value of the variable $x$ is value $\mathtt{x} \in \mathcal{X}$. We use the unit of the Giry Monad to produce a point hyper-distribution. Sequence is defined in the standard monadic manner, by first applying $[\![P]\!]$ to the input and then $\mathbb{D}[\![Q]\!]$ is applied to $[\![P]\!]$'s output hyper-distribution, with a final application of $\mathsf{avg}$ applied to amalgamate equivalent posteriors. The action of $\mathbb{D}[\![Q]\!]$ is to apply $[\![Q]\!]$ to each of the posteriors in the output of $[\![P]\!]$, thus satisfying the condition for Cor. **??**. Finally, notice that the `Print` statement acts like a channel but without creating the joint distribution between the observables and the prior. Instead it formats the (equivalent) result directly as a hyper-distribution. A full description of the QIF-aware program semantics is detailed elsewhere [**?**].

## 5 Example: Implementing plausible deniability

[8] This is essentially a Markov update of the state.

Consider the small program in Fig. **??** which forms the basis for a random-response program. A participant in a survey is asked to input a response *resp* to a yes/no question. If they are concerned about the security of the method of collection, in particular whether their answer will be leaked, they might decline to participate. In order to encourage participation, Warner [**?**], devised a random response protocol which gives participants "plausible deniability" in regards to their responses, if the results of the survey are published.

In Fig. **??** we see the details of the algorithm `SingleRespondent` implemented as a sequential program. The participant's answer is stored in a variable *resp* (1 for "yes" and 0 for "no"). The variable *count* is used to store and then publish the result of the data collection. First a random result is stored in a variable *coin*, where we use "0[1/2]1" to mean that the value is randomised between 0 or 1, using an unbiased "coin toss". Next the variable *count* is updated, and again the update is randomised between either incrementing *count* with the value stored previously in *coin*, or with the participant's choice *resp*. The last act is then to publish the final value of *count*.

A participant worried about the collection procedure might wonder whether the data collected is an accurate recording of their real response *resp*. The answer is "it depends", in the sense that the final value of *count* could be either 1 or 0 whatever the value of *resp*, with the difference between the initial values of *resp* observed through the probabilities ascribed to the possible values of *count* observed. However that difference is bounded by a differentially private guarantee. This fact can be proven by showing that `SingleRespondent` is $\log 3$ differentially private with respect to the two conditions defined by *resp*.

A traditional QIF analysis would construct an explicit channel for the random response protocol to describe how information about the secret (in this case *resp*) can leak. The result of this exercise turns out to be the of the channel `inc` at (**??**). From, this we can initialise *resp* to be either 0 or 1 with probability 1/2 each; finally we can compute $U_{dp_{\log 3}}[\upsilon\rangle\texttt{inc}]$ and observe that it is 0.

An alternative approach is to interpret `SingleRespondent` directly in the QIF semantics above. First we define a mechanism over the secret *resp* as follows. Let $\delta \in \mathbb{D}\{0,1\}$. Define a mechanism $\mathcal{M} : \mathbb{D}\mathcal{X} \to \mathbb{D}^2\mathcal{X}$ [9]

$$\mathcal{M}.\delta \quad := \quad [\![\texttt{SingleRespondent}]\!].\delta \ .$$

Now we examine $U_{dp_{\log 3}}(\mathcal{M}.\upsilon) = 0$, showing similarly that the difference in $resp = 0$ and $resp = 1$, is that for any observation of *count*, the corresponding posteriors differ in probability according to the multiplicative contstraint $e^{\log 3} = 3$.

### 5.1 Random response protocol

[9] Strictly speaking the state is determined by the values of all the program variables. However the only secret that we worry about for this example is the value of *resp*. These details can all be handled by adjusting the definition of $dp_\epsilon$.

```
// Assume resp is either 0 or 1 initially

count:= 0;
        coin:= 0 [1/2] 1;        // Random response
        count:= (count + coin
                 [1/2]           // Randomly include resp or not
                 count + resp);
Print count;     // Announce the approximate count
```

The value resp is either added to the variable count or not; in the case that it is not included, a random response *coin* for that participant is delivered instead.

**Fig. 1.** Randomised response, `SingleRespondent`

An implementation of a full random response protocol is set out in Fig. **??**. For $N$ participants, each participant executes the single response protocol defined at Fig. **??**.

```
// Assume resp is an array of length N set to
// participants' responses, to a survey question.
i := 0;
count:= 0;
while (i<N) {
        coin:= 0 [1/2] 1;        // Random response
        count:= (count + coin
                 [1/2]           // Randomly include participant i or not
                 count + resp[i]);
        i++;
}
Print count;     // Announce the approximate count
```

On each iteration, the participant i is randomly selected for inclusion in the count or not. In the case that the participant's true response *resp*[*i*] is not included, a random response *coin* for that participant is delivered instead.

**Fig. 2.** Randomised response with `N` participants

The privacy for each individual is whether their specific response is private. For that we can use the results above to show that their individual response is protected through Fig. **??** considered as a $\log 3$ differentially private mechanism. Moreover within the context of the other responses, we are able to show that the other respondents do not affect that privacy level. For example, for the final participant in Fig. **??**, the other participants' responses reveals nothing about the final participant's response, thus the protocol is equivalent to $R; R'$ where $R$ corresponds to the collection of the first $N-1$ participants responses and $R$ to the collection of the final participant's response. We have that $R$ is 0-

differentially private with respect to the final participant's response and that $R'$ is $\log 3$-differentially private. Hence by the additive law Cor. **??** we have that the full random response protocol in Fig. **??** is also $\log 3$-differentially private (for that participant). This argument can be generalised to apply to any participant taking part in the random response.

## 6 Experiment and exploration

The experiments described above were carried out using the tool Kuifje [**?**,**?**] which interprets a small programming language in terms of the QIF semantics alluded to above. Kuifje supports the usual programming constructs (assignment, sequencing, conditionals and loops) but crucially it takes into account information flows consistent with QIF. In particular the `Print` statements used in our examples correspond exactly to the observations that an adversary could make during program execution. This allows a direct model for eg. known side channels that potentially expose partially computation traces during program execution.

The basic assumption built into the semantics of Kuifje is that no variable can be observed unless revealed fully or partially through a `Print` statement. For example `Print x` would print the value of variable `x` and so reveal it completely at that point of execution, but `Print(x>0)` would reveal only whether `x` is strictly positive or not. As usual, we also assume that the adversary knows the program code.

Kuifje is implemented in Haskell and makes extensive use of the Giry monad [**?**] for managing the prior, posterior and marginal probabilities in the form of "hyper-distributions".

In order to use Kuifje to analyse Fig. **??** and Fig. **??**, we assume a uniform input for *resp* (*resp*[$i$]). Kuifje then generates the hyper-distribution output, which can then be evaluated against $dp_\epsilon$ for a chosen $\epsilon{>}0$. Since we are only interested in a specific response, we can assume that the secret is determined by *resp* (*resp*[$i$]). We can then adjust the details of $dp_\epsilon$ by setting $\mathcal{V}$ to be sensitive only to different values of *resp* (*resp*[$i$]).

Finally, we note that since Kuifje computes the output hyper-distribution, other properties of Fig. **??** and Fig. **??** can also be explored, such as the Bayes Risk of the *resp*, and the true average number of "yes" respondents.

## 7 Related work

Differential privacy was proposed by Dwork [**?**] to provide mechanisms that satisfy strong privacy guarantees for individuals. Alvim et al. [**?**] were the first to explain the relationship between information-flow channels and differential privacy, and to investigate leakage properties of differentially-private mechanisms modelled as channels [**?**].

There has been recent interest in verification techniques for proving differential-privacy properties, with the intention of providing programmers with the capability to certify privacy guarantees, and to support reasoning. Wang et al.

have [?] proposed a technique called "Shadow execution" to enable the verification of implementations of differentially private algorithms using traditional program logics. Adaptations of Hoare Logic have been proposed [?] based on reasoning about product programs. Bathe et al. [?] use a technique based on probabilistic couplings to enable differential privacy to be treated as a program-verification problem. More generally Barthe et al. [?] describe three verification and programming-language techniques for certifying that programs satisfy the more general $(\epsilon, \gamma)$ differential privacy guarantees, as in:

$$\int_{\mathcal{M}.x} \zeta \quad \leq \quad e^{\epsilon} \times \int_{\mathcal{M}.x'} \zeta \quad + \quad \gamma \; . \tag{8}$$

All of these techniques are supported by automation. Zhang and Kifer [?] use a relational type system to decompose privacy verification into two parts, one for relational reasoning and the other to compute the "privacy budget". A privacy budget is related to the fact that every query to a database, even one protected by differential privacy, leaks some information about the data. A privacy budget denotes an upper limit on information leakage that is insufficient to identify individuals. Finally, Ebadi and Sands [?] have implemented a system based similarly on reasoning rules to keep track of the privacy budget related to datasets.

## 8 Conclusions

We have shown how to use a loss function combined with a QIF-enabled programming semantics to verify privacy properties for programs. We have used the interpreter Kuifje to enable experimental investigation of differential privacy for small sequential programs.

We described the simplest and most restrictive version of differential privacy (**??**), but note that the weaker $(\epsilon, \gamma)$ notion of differential privacy can also be modelled using loss functions. To see this, we note that if $M$ is not $\epsilon$-differentially private for some particular $x, x' \in \mathcal{X}$, we must have:

$$\sum_{y \in \mathcal{Y}} \; (M_{x'y} e^{\epsilon} - M_{xy}) \min 0 \quad < \quad 0 \; . \tag{9}$$

In fact each individual summand is non-zero exactly when $M_{x'y}/M_{xy}$ can be distinguished by more than the "allowed" $e^{\epsilon}$ multiplier. Observe that the sum of those summands is equal to some value $-\gamma'$, and if it is at least $-\gamma$ in (**??**) then $M$ is $(\epsilon, \gamma)$ differentially private. We can formalise this observation using loss functions as follows.

As in Def. **??** we formulate a loss function $dp^*_{\epsilon}$, this time letting $\mathcal{V}{:=}\{(x, x'), \star\}$. If we let $\upsilon$ be the uniform distribution over $x, x'$, we see that $U_{dp_{\epsilon^*}}[\upsilon\rangle M]$ is equal to half the sum in (**??**). Thus we can conclude that $M$ is $(\epsilon, \gamma)$ differentially private if $U_{dp_{\epsilon^*}}[\upsilon\rangle M] \geq -\gamma/2$ for all such pairs $x, x'$. More investigation is required to determine whether this provides a useful characterisation.

Finally we observe that a QIF model is rich enough to capture many other other kinds of risks related to information flow. For example a participant in the random response survey might be more interested in whether their response can be determined with some likelihood, and the response gatherer might be interested in how the output *count* is related to the real "yes" count. Both of these properties can be analysed using loss functions and the QIF interpretation [?,?].

## Acknowledgements

I thank Tom Schrijvers for having the idea of embedding these ideas in Haskell, based on Carroll Morgan's talk at IFIP WG2.1 in Vermont, and for carrying it out to produce the tool Kuifje. Together with Jeremy Gibbons all four of us wrote the first paper devoted to it [?]. (It was Jeremy who suggested the name "Kuifje", the Dutch name for TinTin — and hence his "QIF".)

## References

1. Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. On the information leakage of differentially-private mechanisms. *Journal of Computer Security*, 23(4):427–469, 2015.
2. Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. On the relation between differential privacy and quantitative information flow. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 60–76, 2011.
3. Mário S. Alvim, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. Differential privacy versus quantitative information flow. *CoRR*, abs/1012.4250, 2010.
4. Mário S. Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. Additive and multiplicative notions of leakage, and their capacities. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 308–322. IEEE, 2014.
5. Mário S. Alvim, Kostas Chatzikokolakis, Catuscia Palamidessi, and Geoffrey Smith. Measuring information leakage using generalized gain functions. In *Proc. 25th IEEE Computer Security Foundations Symposium (CSF 2012)*, pages 265–279, June 2012.
6. Mário S. Alvim, Andre Scedrov, and Fred B. Schneider. When not all bits are equal: Worth-based information flow. In *Proc. 3rd Conference on Principles of Security and Trust (POST 2014)*, pages 120–139, 2014.
7. Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, César Kunz, and Pierre-Yves Strub. Proving differential privacy in hoare logic. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 411–424, 2014.
8. Gilles Barthe, Marco Gaboardi, Benjamin GrÕgoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *Proceedings of the*

*31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 749–758, 2016.
9. Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin Pierce. Programming language techniques for differential privacy. *ACM SIGLOG News*, 3(1):34–53, 2016.
10. Marton Bognar and Tom Schrijvers. Kuifje: A prototype for a quantitative information flow aware programming language. https://github.com/martonbognar/kuifje.
11. K. Chatzikokolakis, M.E. Andrés, N.E. Bordenabe, and C. Palamidessi. Broadening the scope of differential privacy using metrics. In *International Symposium on Privacy Enhancing Technologies Symposium*, volume 7981 of *LNCS*. Springer, 2013.
12. Cynthia Dwork. Differential privacy. In *Proc. 33rd International Colloquium on Automata, Languages, and Programming (ICALP 2006)*, pages 1–12, 2006.
13. Hamid Ebadi and David Sands. Featherweight PINQ. *Journal of Privacy and Security*, 2017.
14. M. Giry. A categorical approach to probability theory. In *Categorical Aspects of Topology and Analysis*, volume 915 of *Lecture Notes in Mathematics*, pages 68–85. Springer, 1981.
15. Carroll Morgan Jeremy Gibbons, Annabelle Mciver and Tom Schrijvers. Quantitative information flow with monads in haskell. In *Foundations of Probabilistic Programming*. CUP, 2019. To appear.
16. Annabelle McIver, Larissa Meinicke, and Carroll Morgan. Compositional closure for Bayes Risk in probabilistic noninterference. In *Proceedings of the 37th international colloquium conference on Automata, languages and programming: Part II*, volume 6199 of *ICALP'10*, pages 223–235, Berlin, Heidelberg, 2010. Springer.
17. Annabelle McIver, Carroll Morgan, and Tahiry Rabehaja. Abstract Hidden Markov Models: a monadic account of quantitative information flow. In *Proc. LiCS 2015*, 2015.
18. C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.
19. Geoffrey Smith. On the foundations of quantitative information flow. In Luca de Alfaro, editor, *Proc. 12th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS '09)*, volume 5504 of *Lecture Notes in Computer Science*, pages 288–302, 2009.
20. Yuxin Wang, Zeyu Ding, Guanhong Wang, Daniel Kifer, and Danfeng Zhang. Proving differential privacy with shadow execution. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, pages 655–669, New York, NY, USA, 2019. ACM.
21. S.L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60:63Ð69, 1965.
22. Danfeng Zhang and Daniel Kifer. Lightdp: Towards automating differential privacy proofs. In *Proceedings of Principles of Programming Languages*, pages 1–17, 2017.