# Measuring the Loss of Privacy from Statistics

Michael Carl Tschantz[*]
Computer Science Department
Carnegie Mellon University
mtschant@cs.cmu.edu

Aditya V. Nori
Rigorous Software Engineering
Microsoft Research India
adityan@microsoft.com

**Abstract**

We present a specialization of quantitative information flow to programs that compute statistics. We provide an approach for estimating the information flows present in such programs based on Monte Carlo simulation and argue that it is more accurate than previous approaches in this domain.

## 1 Introduction

Organizations often collect sensitive information about survey respondents. To protect the privacy of the respondents, they only publish aggregate statistics about the responses rather than the responses themselves. These statistics are designed to provide information about the responses as a whole without providing a detailed view of any one response. However, under some circumstances, these statistics may reveal sensitive information about a particular respondent. We would like to quantify how much information about a single respondent can be learned from a given statistic.

For example, a trivially unsafe program might just report the responses themselves including the name of the person who provided each response. Likewise, a trivially safe program might always report "access denied" providing no information.

For a less trivial example, consider a program that takes two non-negative integer salaries and returns their sum:

```
return (salary1 + salary2)
```

Such a program provides an upper bound on each respondent's salary since neither can be greater than twice the average. Furthermore, if the sum is zero, the sum also provides the exact salary of each respondent. If, on the other hand, the sum is one, then two possibilities for each respondent's salary remain: zero and one. As the sum goes up, the number of possibilities goes up. Thus, unlike the trivial cases above in which the program could be analyzed independently of the response it produces, in this case, the value of the produced statistic influences the amount of privacy maintained.

Our goal is to provide an automated method for determining the amount of information that flows through a program that computes a statistic. We further desire that our analysis is accurate enough to provide reasonable results for common statistics. For example, Clark et al. provide an analysis for measuring the mutual information flow from sensitive inputs to public outputs [1]. While their approach produces results accurate enough for their problem domain, confidentiality, it is not accurate enough for use on statistics.

[*]This work was primarily done while the first author was an intern at Microsoft Research India.

For example, it cannot distinguish between a program that simply lists the responses and a program that provides the sum of all the responses.

To meet this goal, we use Monte Carlo simulation. This simple approach has many advantages. By treating the program as a black box, it can work on any program written in any language and is fully automated. We need not create any models or ensure that the program obeys a typing discipline. Running the actual implementation rather than analyzing a specification of what a statistic should calculate catches the effects of bugs. Despite not having a soundness guarantee, with enough samples, our approach will approach the exact values, whereas sound analyses often provide very loose bounds.

Rather than simply provide one number that measures this loss, we provide both the probability distribution over the sensitive attribute for a respondent before and after learning the value of the statistic. From these distributions, many measures of information flow (privacy loss) used in other works can be easily calculated including mutual information [1] and the change in distribution accuracy [2].

First, we present an formal model of programs that produce statistics from a list of survey responses. Second, we formalize the problem and discuss a related problem that is more practical in many settings. Third, we discuss our analysis. Fourth, evaluate our analysis on simple statistics. Lastly, we discuss related work and conclude. While an intuitive understanding of probability suffices for understanding this work, the appendix formalizes our models with measure theory.

## 2 Model

We model a program that computes a statistic as a function $f$ that accepts as input a finite list of survey responses and produces as output the value of the statistic. Let each survey response be an element of the countable set $\mathscr{X}$ and let the value of the statistic range over the countable set $\mathscr{Y}$. Thus, the program is treated as a function $f$ from $\mathscr{X}^*$ to $\mathscr{Y}$. While the restriction to countable sets $\mathscr{X}$ and $\mathscr{Y}$ might seem unnatural given a survey of continuous values such as weights or heights, this is not a limitation in practice since respondents only ever provide this information to a fixed accuracy (such as to the nearest kilogram for weight). Also, we can model probabilistic programs by having $f$ accept a second argument that determines the probabilistic choices. In our implementation, it's irrelevant since it treats programs as black boxes.

The program operates in an environment from which its input comes. Let the set $\Omega$ represent the set of possible worlds and $P$ be a probability measure over these worlds. The survey is conducted and program ran in one of these worlds $\underline{\omega}$, the actual world.

Let $\mathbf{X}$ be a random variable from $\Omega$ to $\mathscr{X}^*$ that provides the inputs to the program. This models the process of conducting the survey, which provides the program some information about the actual world $\underline{\omega}$. We use $\underline{\mathbf{x}} = \langle \underline{x}_1, \ldots, \underline{x}_n \rangle$ to denote the actual survey responses provided to the program: that is, $\underline{\mathbf{x}} = \mathbf{X}(\underline{\omega})$.

The program $f$ computes the value of a statistic of the provided survey responses. This defines a new random variable $Y = f \circ \mathbf{X}$ from $\Omega$ to $\mathscr{Y}$. We use $\underline{y}$ to denote the actual value of the statistic: $\underline{y} = f(\underline{\mathbf{x}}) = f(\mathbf{X}(\underline{\omega}))$.

For example, $X_i$ could be random variable that relates the weight of the $i$th surveyed person. That is, $X_i(\omega)$ represents the weight of the $i$th surveyed person in the possible world $\omega$. Since $\underline{\omega}$ is the actual world, $\underline{x}_i = X_i(\underline{\omega})$ is the actual weight of the $i$th surveyed person. The program $f$ could accept a list of such weights and compute their mean. Then $Y$ would be a random variable that provides the mean of the respondents given a possible world with the actual mean being $\underline{y} = f(\underline{x}_1, \ldots, \underline{x}_n)$.

We model an adversary as attempting to determine the value taken by some random variable $Z$ where $Z$ ranges over $\mathscr{Z}$. That is, the adversary, would like to determine $\underline{z} = Z(\underline{\omega})$. For example, $Z$ might be *The weight of Bob* or *Bob has AIDS*. The surveyor must determine for which random variables $Z$ the adversary should not be able to determine the value taken. These random variables will vary from survey to survey depending on the information collected by the survey and privacy expectations of the respondents.

The adversary has some prior beliefs about $\underline{z}$. We assume that the adversary knows what worlds are possible, how the survey was conducted, and what statistic was computed (that is, he knows $\Omega$, $\mathbf{X}$, and $f$). However, we assume that the adversary does not know the actual world $\underline{\omega}$ or the actual responses $\underline{x} = \mathbf{X}(\underline{\omega})$. Rather than knowing the actual probability measure $P$, which is impossible to know exactly in many realistic environments, the adversary has beliefs about the world represented as a probability measure $Q$.

## 3 Problem Formalization

Before formalizing the problem, we provide some notation. Given a random variable $Z$ and probability measure $Q$, we use $Q_Z$ to denote the distribution $D$ over $\mathscr{Z}$ such that $D(z) = Q[Z = z]$ for all $z$ in $\mathscr{Z}$. Similarly, $(Q|Y = y)_Z$ represents the distribution $D$ such that $D(z) = Q[Z = z|Y = y]$ for $y \in \mathscr{Y}$ such that $Q[Y = y] \neq 0$.

Our goal is to provide an analysis that computes a comparison of the adversary's knowledge before and after seeing the statistic $\underline{y}$. That is, a comparison of the distribution of $Q_Z$ and the distribution $(Q|Y = \underline{y})_Z$. Since many such comparisons exist, our analysis will provide both $Q_Z$ and $(Q|Y = \underline{y})_Z$ and allow the analysis user to perform any selected comparison upon them.

While a comparison of $Q_Z$ and $(Q|Y = \underline{y})_Z$ is ideal, it seems unreasonable that the surveyor would know the adversary's prior beliefs $Q$. Furthermore, the surveyor cannot do a worse case analysis over all possible values for $Q_Z$ since it could be arbitrarily bad as an adversary could be arbitrarily ignorant before seeing the program output. Thus, we must make some assumptions about the adversary to produce a problem that the surveyor can practically solve given reasonably accessible information.

First, we assume that the adversary bases his prior distribution $Q_Z$ on the actual probability measure $P$. That is, we assume that $Q_Z$ is $P_Z$. This assumption, as pointed out by Clarkson et al. [2], is made implicitly by most works on quantitative information flow (e.g., the work of Clark et al. [1]). This first assumption might appear to not help us since we have traded one unknown, $Q$, for another unknown, $P$. However, unlike $Q_Z$, the surveyor can estimate $P_Z$ using the next three assumptions.

Second, we assume that $Z$ is determined by $\mathbf{X}$. That is, we assume that the surveyor can decompose $Z$ using some function $g$ such that $Z = g \circ \mathbf{X}$. For example, if $Z$ is the response of the first respondent, then $g$ is a function that returns the first response from the sequence of actual responses $\mathbf{X}(\underline{\omega})$. This assumption is reasonable since such random variables are the most vulnerable to attack. (If $Z$ is not completely determined by $\mathbf{X}$, then the surveyor would have to also provide an estimation of the other factors that determine $Z$. It would still be possible to use our approach, but we wish to avoid this complication.)

Third, we assume that the adversary knows the number of responses in the actual responses $\underline{\mathbf{x}} = \langle \underline{x}_1, \dots, \underline{x}_n \rangle = \mathbf{X}(\underline{\omega})$. That is, he knows $n$. Since most surveys publish the number of responses examined, this assumption is not too limiting. Fixing $n$, we can treat $\mathbf{X}$ as consisting of $n$ random variables $X_1$ to $X_n$ with each $X_i$ producing one response $x_i$.

Fourth, we assume that $X_1$ to $X_n$ are independent and identically distributed. Statistically accurate surveys will meet this assumption by design. Under this assumption, $x_1$ to $x_n$ are $n$ samples from a single distribution $P_X$. Given the $n$ samples $\mathbf{x}$, the surveyor can approximate $P_X$. Let $\hat{P}_X$ be one such approximation selected by surveyor. This estimates $P_{\mathbf{X}}$ as $\hat{P}_{X^n}$ (i.e., the distribution resulting from $n$ independent and identically distributed copies of $X$).

These assumptions combine to allow the surveyor to estimate $Q_Z$ as $\hat{P}_{g \circ X^n}$. The problem then becomes to compute a comparison of $\hat{P}_{g \circ X^n}$ and $(\hat{P}|Y = \underline{y})_{g \circ X^n}$ from the following inputs:

- the program $f$ where $Y = f \circ \mathbf{X}$,
- the actual value of the responses $\underline{\mathbf{x}} = \mathbf{X}(\underline{\omega})$,
- a function $g$ where adversary is attempting to learn $Z(\underline{\omega}) = g(\mathbf{X}(\underline{\omega}))$, and
- an approximation $\hat{P}_X$ of the distribution $P_X$ that generated the responses and determines $Z$.

Note the problem depends not just on the statistic $f$, but also on the actual value of the statistic, the information that the adversary would like to learn, and the estimation of the distribution $P_X$. This requires that the survivor solve this problem each time the statistic is to be applied to different responses or with a different adversary. However, as argued in the introduction, the amount of information flow is sensitive to these changes.

## 4 Analysis

We now present a simple analysis for providing an approximate answer to the practical version of the problem above. We also discuss our implementation of this analysis.

We use Monte Carlo simulation to estimate $(\hat{P}|Y = y)_Z$ as follows. We repetitively use $\hat{P}_X$ to generate a sample $\mathbf{x}'$ from $\hat{P}_{X^n}$, we run $f$ on $\mathbf{x}'$ to produce $y'$, and we run $g$ on $\mathbf{x}'$ to produce $z'$. By keeping track of the value $z'$ takes on each time $y'$ is equal to $\underline{y}$, we can construct estimations of $\hat{P}_Z$ and $(\hat{P}|Y = y)_Z$ in the usual way: we estimate $\hat{P}_Z(z)$ as the number of samples that result in $Z = z$ divided by the number of samples and we estimate $(\hat{P}|Y = y)_Z(z)$ as the number of samples that resulted in both $Z = z$ and $Y = y$ divided by the number of samples that resulted in $Y = y$.

An advantage of this method is it works for any $f$ and $g$ that are functions. (The method also works for randomized functions provided that the surveyor can model their sources of randomness.) The method runs on large, complex programs even without source code.

Since constructing $(\hat{P}|Y = y)_Z$ takes memory linear in $\mathscr{Z}$ (not counting any memory used by $f$ or $g$), this approach will not work for large $\mathscr{Z}$. However, one may choose to focus on a subset of $\mathscr{Z}$ that indicate sensitive outcomes to reduce memory usage to the size of this subset. For example, one might focus only on $\underline{z}$, the actual value that $Z$ takes on, and calculate $\hat{P}(Z = \underline{z}|Y = y)$ for comparison to $\hat{P}(Z = \underline{z})$.

Several factors can slow down gaining an accurate estimation. If $f$ or $g$ is a time intensive computation, our dynamic analysis will be slow. A large size of $\mathscr{X}$ or $n$, or a low value for $\hat{P}(Y = \underline{y})$ can each result in needing a large number of samples for constructing an accurate estimation of $(\hat{P}|Y = y)_Z$. While surveys that ask for exact answers can have a large $\mathscr{X}$, many only ask multiple choice questions yielding a more manageable $\mathscr{X}$.

In general a large $n$ can be problematic, but in the following special case, we can optimize our analysis to not depend upon $n$. Some statistics strips sensitive information (such as name) from each $X_i$ and lists the sanitized form. Such statistics $f$ have the form $f([X_1, X_2, \ldots, X_n]) = [f'(X_1), f'(X_2), \ldots, f'(X_n)]$ for some function $f'$. If $Z$ is independent of all $X_i$ except one of them, say $X_i$, then

$$\begin{aligned}\hat{P}(Z = z|Y = y) &= \hat{P}[Z = z|f([X_1, \ldots, X_i, \ldots, X_n]) = [y_1, \ldots, y_i, \ldots, y_n]] \\ &= \hat{P}[Z = z|f'(X_1) = y_1, \ldots, f'(X_i) = y_i, \ldots, f'(X_n) = y_n] \\ &= \hat{P}[Z = z|f'(X_i) = y_i]\end{aligned}$$

where the last equality follows from $Z$ being independent of all $X_j$ other than $X_i$. Thus, we can ignore all $X_j$ other than $X_i$. This greatly speeds up the approximation.

## 5 Evaluation

To evaluate our approach, we fix a method of comparing $\hat{P}_Z$ and $(\hat{P}|Y = \underline{y})_Z$. The method we choose uses *entropy*, an information theoretic measure of the amount of uncertainty associated with a distribution. The entropy of the distribution $\hat{P}_Z$ is

$$\mathcal{H}(\hat{P}_Z) = -\sum_{z \in \mathscr{Z}} \hat{P}[Z = z] \log_2 \hat{P}[Z = z]$$

and the entropy of the distribution $(\hat{P}|Y = \underline{y})_Z$ is

$$\mathcal{H}((\hat{P}|Y = \underline{y})_Z) = -\sum_{z \in \mathscr{Z}} \hat{P}[Z = z|Y = y] \log_2 \hat{P}[Z = z|Y = y]$$

(One usually speaks of the entropy of a random variable with the underlying probability measure $P$ being understood. Since we are dealing with two probability measures, $\hat{P}$ and $\hat{P}|Y = y$, we choose to make them explicit.)

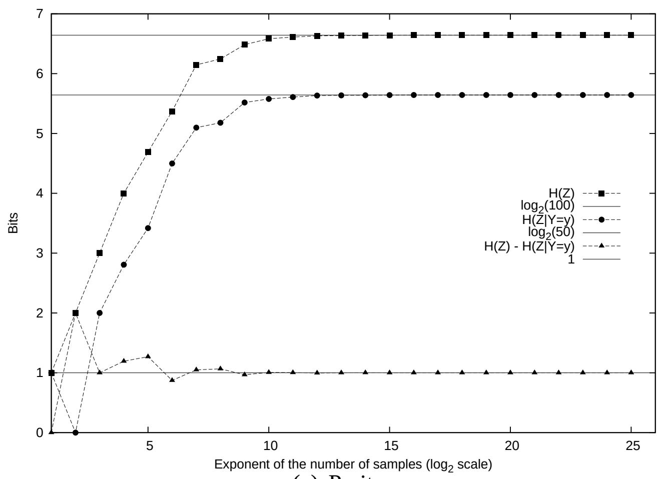
The comparison of the distributions $\hat{P}_Z$ and $(\hat{P}|Y = \underline{y})_Z$ we use is the difference of their entropies: $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}|Y = \underline{y})_Z)$. Clark et al. [1] argues that this difference measures the amount of information that flows from $Y = \underline{y}$ to the adversary about $Z$ since it is the decrease in the uncertainty of $Z$ after learning that $Y$ is equal to $\underline{y}$. Indeed, this difference is related to mutual information, an information theoretic measure of how much information one random variable provides about another. Ignoring that $Y = \underline{y}$ is a condition and not a random variable, $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ may be seen as providing the mutual information $\mathcal{I}(Z; Y = \underline{y})$ between $Z$ and $Y = y$ for a deterministic program.

Using entropy, we computed the difference between $\hat{P}_Z$ and $(\hat{P}|Y = \underline{y})_Z$ for various statistics. In all cases we used the uniform distribution over $0$ to $99$ for each $X_i$. We selected the uniform distribution since by having a high variance, we expected it to be a challenging distribution for the analysis in the sense of requiring a large number of samples. For $Z$, we used the value of the first input $X_1$.

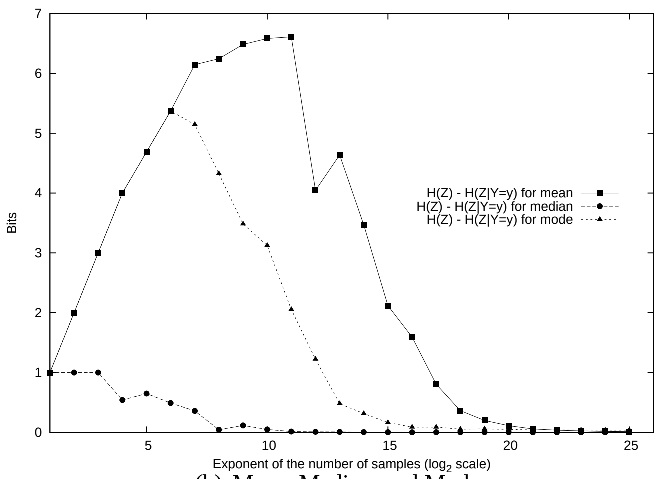
The first statistic we consider is the parity of $X_1$. This is not a particularly interesting statistic, but we can exactly calculate $\mathcal{H}(\hat{P}_Z)$ to be $\log_2(100)$ and $\mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ to be $\log_2(50)$ allowing us to see the accuracy of our analysis. To study convergence and show that our analysis can provide accurate estimations, we show the estimations produced using various numbers of samples in Figure 1(a). The y-axis shows the estimated values for the entropies and mutual information while the x-axis shows the number of samples performed, which ranges from $2^1$ to $2^{25}$. This table shows that the estimations of the values of $\mathcal{H}(\hat{P}_Z)$ and $\mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ approach their real values as the number of samples increases. Thus, the estimation of $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ approaches its real value as well. By $2^{25}$ samples, the mutual information is less then $0.0000003$ bits away from the exact value of $1$.

Note that the estimations of $\mathcal{H}(\hat{P}_Z)$ and $\mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ tend to approach from below. Indeed, our estimator is a biased one. While others have created less unbiased estimators ([5] provides a recent overview), we simply opt to use more samples instead.

The results for more realistic statistics (mean, median, and mode) are shown in Figure 1(b). Note that the value of the estimations for all three statistics stabilized by $2^{23}$ samples. The raise and fall of the estimations is due to both the estimations of $\mathcal{H}(\hat{P}_Z)$ and $\mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ approaching their real values from below with $\mathcal{H}(\hat{P}_Z)$ approaching it's real value more quickly than $\mathcal{H}((\hat{P}|Y = \underline{y})_Z)$. This creates a period where $\mathcal{H}(\hat{P}_Z)$ is a reasonable estimation and $\mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ is a radical underestimation resulting in $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ being a radical overestimation.



(a) Parity



(b) Mean, Median, and Mode

Figure 1: Estimations for Various Statistics

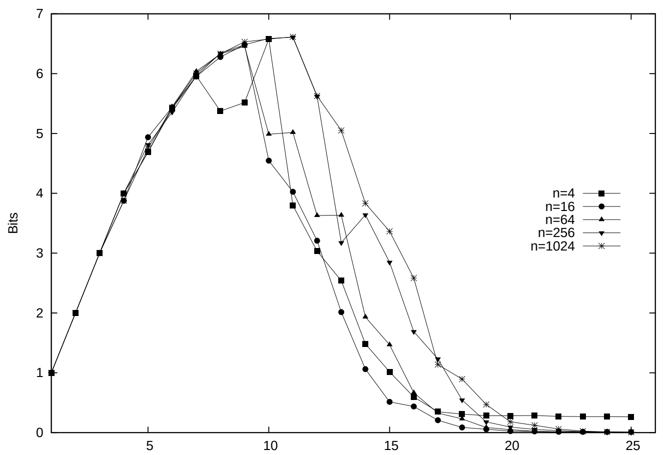
| Statistic | $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}\|Y = \underline{y})_Z)$ | Run Time (secs) |
|---|---|---|
| Parity | 0.999999797131 | 639 |
| Mean | 0.0125233560025 | 684 |
| Median | 0.00205987477602 | 1498 |
| Mode | 0.0376910036281 | 2444 |

Table 1: Summery of Analysis Results for Four Statistics

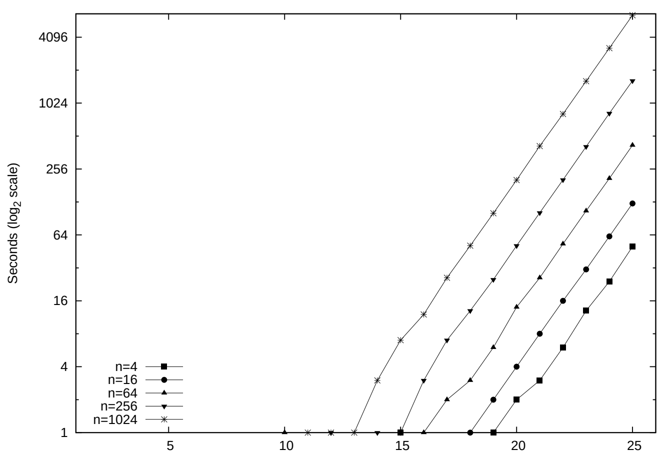
Table 1 summarizes the estimations for $2^{25}$ samples and shows the amount of time taken to compute these results for running on a $3.2$ GHz, $64$-bit processor. Note that the estimations of $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ for the mean, median, and mode are all lower than for parity. This conforms our suspicion that aggregate statistics tend to reveal little about their respondents. The time for estimating these values grow linearly with the number of samples as expected. The slowest was mode, which took $41$ minutes for $2^{25}$ samples. However, an estimation that differs by less than $0.021$ bits ($0.32\%$) is available in under a minute using $2^{19}$ samples.

To explore how the number samples $n$ affects the value of $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ and the rate of convergence to it, Figure 2(a) shows the estimations of $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}|Y = \underline{y})_Z)$ for the mean for varying sizes $n$. Using more respondents decreased the difference between $\mathcal{H}(\hat{P}_Z)$ and $\mathcal{H}((\hat{P}|Y = \underline{y})_Z)$. However, it increased the number of samples needed for convergence since convergence requires seeing many samples such that $Y = \underline{y}$, which becomes a less common event as $n$ increases. Furthermore, it increased the amount of time needed to compute the value of the statistic keeping the number of samples constant since calculating the mean over more respondents takes longer. In the worse case, the mean over $1024$ respondents, it took $109$ minutes for $2^{25}$ samples with convergence still not reached. Figure 2(b) plots these run times.

Our implementation may be downloaded from `http://www.cs.cmu.edu/~mtschant/mcqif/`



(a) Estimations of $\mathcal{H}(\hat{P}_Z) - \mathcal{H}((\hat{P}|Y = \underline{y})_Z)$



(b) Run Times for Analysis

Figure 2: Results for Mean with Various Numbers of Respondents

## 6 Related Work

**Quantitative Information Flow.** Much work has been done on information flow analysis. We will only discuss those works that deal with quantifying the flow of information. These works concern themselves with either confidentiality or integrity. In both cases, the tool user partitions the inputs and outputs of the program into high-level and low-level classes. Quantitative information flow for confidentiality measures how much the high-level inputs affect the low-level outputs. Quantitative information flow for integrity, on the other hand, measures how much the low-level inputs affect the high-level outputs. The two problems are dual and an analysis for one will apply to the other. Since our work fits under the confidentiality problem, we will discuss all related works from this angle even if they were created with integrity in mind.

The work of Clark, Hunt, and Malacaria presents a formal model of programs for quantifying information flows and a static analysis that provides lower and upper bounds on the amount of information that flows [1]. They measure information flow as the mutual information between the high-level inputs and low-level outputs given that the adversary has control over the low-level inputs. That is, they measure $\mathcal{I}(L^{\mathsf{out}}; H^{\mathsf{in}}|L^{\mathsf{in}})$ where $L^{\mathsf{out}}$ is a random variable representing the low-level outputs, $H^{\mathsf{in}}$ is one representing the high-level inputs, and $L^{\mathsf{in}}$ is one representing low-level inputs. Unlike our work that measures the information flow in a program given a particular input, their analysis provides upper and lower bounds on the size of the information flow in a given program regardless of the actual inputs or the distributions that generate them. Since the upper bound holds for all input distributions, it is an upper bound on the channel capacity of the program.

Their analysis, if implemented, could be used for our problem by treating the inputs $\mathbf{X}$ as $H^{\mathsf{in}}$, using $Y$ as $L^{\mathsf{out}}$, and assuming that $Z = \mathbf{X}$. ($L^{\mathsf{in}}$ is unused since we do not allow the adversary to control any inputs to the statistic.) However, their analysis produces bounds that are too loose for our purposes. For example, no matter how many independent and identically distributed samples goes into a mean, their analysis will state that all the information about the first sample is provided as output despite the fact that it would be hidden amongst other samples.

McCamant and Ernst provide a dynamic analysis for quantitative information flow using the mutual information formalization [6]. Their analysis provides an upper bound on the flow of information of a single path of execution in a program. Their analysis converts a path of execution into a flow network. They then find the max cut of the network to bound the information flow. Unlike us, they provide a sound upper bound for that path of execution instead of an estimate. However, like the work of Clark et al., their analysis does not account for information hiding in the calculations like a sum making the bound too loose to be useful for our purposes.

Newsome and Song also provide a dynamic analysis for quantitative information flow using the mutual information formalization [7]. Their analysis converts a single path of execution into a logical formula that characterizes the path. Each solution to this formula corresponds to a value that the output $Y$ can take on while taking that path of execution. If all such solutions are found, this provides the channel capacity between $\mathbf{X}$ and $Y$ provided only the analyzed path of execution is ever used. In practice, a theorem prover can rarely find all such solutions, and thus, their analysis only provides a lower bound on the channel capacity. Whether or not this bound is tight enough for our uses depends on the theorem prover and the formula.

Clarkson, Myers, and Schneider object to the mutual information formulation of quantitative information flow [2]. Instead they proposed a formulation using the beliefs of the adversary. However, such a formulation is often not practical since the surveyor often will not know the adversary's beliefs. After adjusting their definitions for our uses, information flow is defined to be $\mathcal{D}(Q_Z \to \dot{\underline{z}}) - \mathcal{D}((Q|Y = \underline{y})_Z \to \dot{\underline{z}})$ where $Q$ is the adversary's beliefs, $\underline{z} = Z(\omega)$ is the actual value of the random variable the adversary is attempting to learn, $\dot{\underline{z}}$ is a distribution over $\mathscr{Z}$ that assigns 1 to $\underline{z}$ and 0 to every other element of $\mathscr{Z}$, and $\mathcal{D}(Q_Z \to P_Z)$ is the relative entropy:

$$\mathcal{D}(Q_Z \to P_Z) = \sum_{z \in \mathscr{Z}} P_Z(z) \log \frac{P_Z(z)}{Q_Z(z)}$$

For deterministic programs, they prove that $\mathcal{D}(Q_Z \to \dot{z}) - \mathcal{D}((Q|Y = y)_Z \to \dot{z})$ reduces to $-\log Q_Y(y)$. We can calculate this given an approximation of $Q_Y$ directly. We could also calculate this using our sampling approach given an approximation of $Q$ or $Q_X$.

**Preserving Privacy.** Statistical disclosure limitation attempts to preserve privacy despite releasing statistics. (For an overview see [4].) Most of the methods used in this line of work are specialized for a single class of statistics. Most often this is the class of frequency tables, tables that record the number of respondents with various combinations of attributes. Tables of magnitudes and sanitized individual responses (microdata) are also considered. While our approach is more efficient for some statistics than others, it can work on any statistic provided it is calculated by a computer.

Other works in statistical disclosure limitation use Monte Carlo simulation for purposes other than ours. For example, Slavković uses it construct an estimation of probability distributions over outputs ($\hat{P}_X$ in our notation) [8].

Differential privacy is a formalization of what it means for a statistic to maintain the privacy of the respondents about which it is calculated [3]. It requires that the output that the program produces is probably no different from the output it would have produced if one respondent were dropped from or added to the survey. In particular, for a statistic $f$ to have $\epsilon$-differential privacy, it must be the case that for all sets $D_1$ and $D_2$ of responses that differ on at most one response and all subsets $S$ of the range of $f$

$$\Pr[f(D_1) \in S] \leq e^{\epsilon} \Pr[f(D_2) \in S]$$

This ensures that the probability of the statistic's output falling in some set $S$ changes only by a factor $e^{\epsilon}$ as a single respondent's information is either added or removed from the survey. Intuitively, if the statistic probably looks the same regardless of if a person is surveyed or not, an adversary cannot learn much information about the person. While we could consider $\epsilon$ to be measure of information flow, it does not lend itself to the analysis of many standard statistics since they do not have $\epsilon$-differential privacy for any value of $\epsilon$. For example, the mean of respondent incomes would not satisfy $\epsilon$-differential privacy for any $\epsilon$ since it would surely change by at least a small amount with a respondent removed. (A version of the mean statistic that adds random noise to the result could be constructed to satisfy $\epsilon$-differential privacy for an $\epsilon$ that depends upon the amount of noise added.)

## 7 Conclusions and Future Work

We have provided an analysis for determining the amount of information that an adversary learns from a statistic given various assumptions. Future work could ease these assumptions. However, this work and all works on quantitative information flow must make some assumption about the adversary. In most works, including our own, they assume that the adversary's beliefs $Q$ are in line with the actual word $P$ and that adversary has no additional background knowledge. Clarkson et al. instead assume they can model the adversary. Both of these assumptions are troubling. This suggests that methods that do not depend on the adversary, such as differential privacy [3], might provide a better solution to protecting privacy. However, it considers every standard statistic (mean, median, mode, etc.) equally and completely unprivate.

Other future work could combine our method with static approaches for information flow such as the work of Clark et al. [1]. Such a hybrid approach, if possible, might scale to systems too large or slow for our Monte Carlo approach while using our approach to closely examine key components of the program.

## References

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

[2] CLARKSON, M. R., MYERS, A. C., AND SCHNEIDER, F. B. Belief in information flow. In *CSFW '05: Proceedings of the 18th IEEE workshop on Computer Security Foundations* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 31–45.

[3] DWORK, C. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)* (2006), vol. 2, pp. 1–12.

[4] FEDERAL COMMITTEE ON STATISTICAL METHODOLOGY. Statistical disclosure limitation methodology. Statistical Policy Working Paper 22, 2005.

[5] KENNEL, M. B., SHLENS, J., ABARBANEL, H. D. I., AND CHICHILNISKY, E. J. Estimating entropy rates with bayesian confidence intervals. *Neural Computation 17*, 7 (jul 2005), 1531–1576.

[6] MCCAMANT, S., AND ERNST, M. D. A simulation-based proof technique for dynamic information flow. In *PLAS '07: Proceedings of the 2007 workshop on Programming languages and analysis for security* (New York, NY, USA, 2007), ACM, pp. 41–46.

[7] NEWSOME, J., AND SONG, D. Influence: A quantitative approach for data integrity. Tech. Rep. CMU-CyLab-08-005, CyLab, Carnegie Mellon University, 2008.

[8] SLAVKOVIĆ, A. B. *Statistical Disclosure Limitation Beyond the Margins: Characterization of Joint Distributions for Contingency Tables*. PhD thesis, Carnegie Mellon University, 2004.

## A The Model More Formally

In this section, we provide definitions that are more formal then the ones found in Sections 2 and 3.

Formally, we model the environment from which program inputs come as a probability space $\langle \Omega, \mathcal{F}, P \rangle$ with the sample space $\Omega$, events $\mathcal{F}$, and probability measure $P$ that models this environment.

Let $\mathscr{X}^*$ be the set of inputs that the modeled program can consume. We assume that $\mathscr{X}$ is countable, implying that $\mathscr{X}^*$ is countable. This ensures that $\langle \mathscr{X}^*, 2^{\mathscr{X}^*} \rangle$ a measurable space. The random variable $\mathbf{X}$, which models program inputs, is from the probability space $\langle \Omega, \mathcal{F}, P \rangle$ to the measurable space $\langle \mathscr{X}^*, 2^{\mathscr{X}^*} \rangle$.

Let $\mathscr{Y}$ be the set of outputs that the modeled program can produce. We assume that $\mathscr{Y}$ is countable, and thus, $\langle \mathscr{Y}, 2^{\mathscr{Y}} \rangle$ is a measurable space. Let $f : \mathscr{X}^* \to \mathscr{Y}$ be a function that models the program. Let $Y$ be $f \circ \mathbf{X}$, which models the output of the program. $Y$ is from the probability space $\langle \Omega, \mathcal{F}, P \rangle$ to the measurable space $\langle \mathscr{Y}, 2^{\mathscr{Y}} \rangle$. $Y$ is a well-defined random variable since for any $S \in 2^{\mathscr{Y}}$, $f^{-1}(S)$ must be in $2^{\mathscr{X}^*}$ and the state space of $\mathbf{X}$ is $\langle \mathscr{X}^*, 2^{\mathscr{X}^*} \rangle$ ensuring that $\mathbf{X}^{-1}(f^{-1}(S)) = Y^{-1}(S)$ is in $\mathcal{F}$.

We model an adversary as attempting to determine the value took on by some random variable $Z$ from $\langle \Omega, \mathcal{F}, P \rangle$ to some measurable space $\langle \mathscr{Z}, 2^{\mathscr{Z}} \rangle$, again, assuming that $\mathscr{Z}$ is countable.

We model the adversary's beliefs about the world as a probability measure $Q$ on $\langle \Omega, \mathcal{F} \rangle$.

Given a random variable $X$ from $\langle \Omega, \mathcal{F}, P \rangle$ to $\langle \mathscr{X}, \Sigma \rangle$, the *distribution* $P_X$ is the pushforward measure of $P$ by $X$. That is, $P_X(E) = P(X^{-1}(E))$ for $E \in \Sigma$.

Given a probability space $\langle \Omega, \mathcal{F}, P \rangle$ and random variable $X$ from $\langle \Omega, \mathcal{F}, P \rangle$ to $\langle \mathscr{X}, \Sigma \rangle$, we write $P|Y = y$ for the probability measure such that $(P|Y = y)(E) = P(E \cap Y^{-1}(\{y\}))/P(Y^{-1}(\{y\}))$. Note that $\langle \Omega, \mathcal{F}, P|Y = y \rangle$ is a probability space with the same random variables as $\langle \Omega, \mathcal{F}, P \rangle$.

Thus, given probability space $\langle \Omega, \mathcal{F}, P \rangle$, random variable $Y$ from $\langle \Omega, \mathcal{F}, P \rangle$ to $\langle \mathscr{Y}, \Sigma_Y \rangle$, and random variable $Z$ from $\langle \Omega, \mathcal{F}, P \rangle$ to $\langle \mathscr{Z}, \Sigma_Z \rangle$, $(P|Y = y)_Z$ is the distribution $D$ such that $D(z) = P(Z^{-1}(\{z\}) \cap Y^{-1}(\{y\}))/P(Y^{-1}(\{y\}))$ for $y \in \mathscr{Y}$ such that $P(Y^{-1}(\{y\})) \neq 0$.